CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DA PRAIA GRANDE
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE
E DESENVOLVIMENTO DE SISTEMAS

RELATÓRIO TÉCNICO DE CONCLUSÃO DE CURSO

EMA - SOFTWARE DE ORGANIZAÇÃO E ADMINISTRAÇÃO DOS NEGÓCIOS PARA OS PROFISSIONAIS AUTÔNOMOS

RENAN DE SOUZA PINTO
THIAGO FERNANDO LIMA DE MORAIS
VINÍCIUS LORETO FERREIRA
VÍTOR JESUS DO CARMO

Orientador:
Prof. RODRIGO LOPES SALGADO

PRAIA GRANDE
DEZEMBRO DE 2018

RENAN DE SOUZA PINTO

THIAGO FERNANDO LIMA DE MORAIS

VINÍCIUS LORETO FERREIRA

VÍTOR JESUS DO CARMO

EMA - SOFTWARE DE ORGANIZAÇÃO E ADMINISTRAÇÃO DOS NEGÓCIOS PARA OS PROFISSIONAIS AUTÔNOMOS

Relatório Técnico-Científico apresentado à Faculdade de Tecnologia de Praia Grande, como parte dos requisitos para a obtenção do título de Tecnólogo em Análise e desenvolvimento de sistemas.

Orientador: Prof. RODRIGO LOPES SALGADO

PRAIA GRANDE

DEZEMBRO DE 2018

AGRADECIMENTOS

Gostaríamos de agradecer a todos que fizeram parte dessa nossa jornada.

Todos os professores que dividiram conosco um pouco de sua sabedoria e que nos deram o exemplo de quão nobre é compartilhar o conhecimento adquirido.

Aos colegas de sala que estiveram conosco em momentos de alegria e momentos de tensão, mas o mais importante, sempre estiveram ao nosso lado tornando essa jornada mais agradável.

Não podemos esquecer de todos aqueles que entenderam nossa ausência, pois essas são pessoas especiais que sabem como foi difícil abrir mão de várias coisas em busca desse sonho. Queremos que saibam o quão importante foi a compreensão e todo o apoio que nos deram dado durante esse período.

RESUMO

A rotina administrativa em um negócio demanda tempo e conhecimento para sua execução. Percebemos, a partir de uma psicóloga, que muitas vezes o fator que mais atrapalha a organização de seu negócio é a falta de tempo, pois o tempo empregado em rotinas administrativas poderiam lhe gerar mais renda em sua atividade principal.

A partir desse cenário se deu a criação da EMA (Electronic Management Assistant), uma aplicação que auxilia na organização das rotinas administrativas de um profissional autônomo.

O projeto apresenta um relatório técnico do desenvolvimento de um software, desde a etapa de planejamento até sua concepção tendo com base algumas boas práticas da engenharia de software como PMBOK 6° edição e da UML.

O resultado deste trabalho é uma ferramenta capaz de auxiliar o profissionais autônomos em suas rotinas administrativas, como: agendamento de seus atendimentos, controle de faturas, cadastro de clientes e gerenciamento de reajuste de preço.

Durante o desenvolvimento são apresentadas ferramentas utilizadas por empresas de desenvolvimento de software e como o projeto fez utilização destas ferramentas.

Por fim o resultado foi satisfatório, pois o software concebido com o projeto cumpriu os requisitos levantados, sendo capaz de auxiliar o profissional autônomo e fazer com que o mesmo economize tempo na rotina administrativa de seus negócios.

Palavras-Chave: Engenharia de software, profissional autônomo, desenvolvimento de software

ABSTRACT

The administrative routine in a business demands time and knowledge for execution. By watching a psychologist, we realized that often the factor that hinders the organization of her business is the lack of time, since the time spent in administrative routines could generate more income in her main activity..

From this scenario, EMA (Electronic Management Assistant) was created, an application that aims to help the organization of administrative routines of an autonomous professional.

This project presents a technical report on the development of a software, from the planning stage to its design, based on some good software engineering practices such as PMBOK 6th edition and UML.

The result of this project is a tool capable of assisting professionals in their administrative routines, such as: scheduling of their attendance, invoice control, customer and price management.

We're presenting tools used by software development industries and how our project made use of these ones.

Finally, the result was satisfactory, since the software designed fulfilled the requirements, being able to help autonomous professional routines and save time in administrative routine of their business.

Keywords: Software Engineering, Autonomous Professional, Software Development

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1. INTRODUÇÃO

No início do segundo trimestre de 2018 o IBGE realizou a Pesquisa Nacional por Amostra de Domicílios Contínua, que mostra que a taxa de desemprego dobrou no Brasil desde 2017, passando de 6,5% para 13,1%.

> É importante destacar ainda a diminuição de postos de trabalho na Indústria (2,7%, ou menos 327 mil pessoas), Construção (5,6%, ou menos 389 mil pessoas) e Comércio (2,2%, ou menos 396 mil pessoas). (AGÊNCIA IBG NOTÍCIAS, 2018)

Porém houve um crescimento no número de trabalhadores autônomos no segmento de prestação de serviços. De acordo com o IBGE, o segundo agrupamento de atividades que mais cresceu no período de 2003 a 2015 foi o de serviços prestados à empresa, aluguéis, atividades imobiliárias e intermediação financeira (32,4%).

Ao contrário de uma empresa, um trabalhador autônomo tem recursos bem limitados e não possui assistência de funcionários para efetuar as tarefas administrativas. Com isso o trabalhador dedica um tempo muito precioso para efetuar essas rotinas administrativas e algumas vezes o autônomo nem possui conhecimento suficiente para isso. Existem algumas ferramentas que facilitam esse dia a dia, porém muitas delas são caras e outras dedicadas a venda de mercadorias.

Pensando em atender um nicho de autônomos prestadores de serviço, será desenvolvida uma ferramenta simples e capaz de auxiliar o profissional nas tarefas administrativas básicas (como por exemplo: gerenciamento de agenda, gestão financeira e cadastro de clientes), de uma forma simples e rápida.

Nosso objetivo é auxiliar profissionais autônomos de forma simples e com baixo custo, para que possa ter maior controle do seu negócio e mais tempo para dedicar-se ao que realmente faz parte do seu *core business*.

1.1. OBJETIVO GERAL

O objetivo é desenvolver uma aplicação que possa proporcionar ao profissional autônomo mais tempo para se dedicar a sua atividade final ao invés de gastar com os controles e burocracias administrativas do negócio.

1.1.1 Objetivos específicos

- Desenvolver um sistema que atenda a demanda de um profissionais autônomos;
- Utilizar plataforma web e responsiva;
- Permitir que o software diminua o tempo gasto na gestão.

2. PLANEJAMENTO DO PROJETO

O planejamento deste projeto teve início na inquietação de uma profissional de psicologia que gastava muito tempo na organização das atividades administrativas de seu consultório, com organização de agenda e finanças.

Levando em conta que sua consulta dura em média uma hora, cada hora gasta em atividade burocrática faz com que a profissional deixe de atender um cliente, logo essa atividade influencia diretamente em sua receita.

Com isso, começamos a analisar as tarefas que poderiam ser automatizadas, trazendo assim praticidade, mobilidade, confiabilidade e diminuindo assim o risco na gestão de seu negócio, já que todo esse controle era feito de forma manual em um caderno. E para nortear o desenvolvimento deste projeto, através das melhores práticas de desenvolvimento de software, utilizamos referências do Guia PMBOK 6ª Edição e UML.

2.1 DEFINIÇÃO DO PROJETO

Em entrevista com a psicóloga conseguimos identificar as tarefas que mais demandam tempo e que a mesma tem maior dificuldade de controlar.

Durante a conversa, a profissional nos relatou a dificuldade de saber quando um cliente está inadimplente ou até mesmo sobre quais sessões se trata um pagamento, já que cada cliente possui um valor e uma data de pagamento diferente. Os pagamentos ocorrem mensalmente com base na quantidade de atendimentos efetuadas, ou seja, o mesmo cliente pode pagar um valor diferente a cada mês, dependendo da quantidade de consultas, o que dificulta ainda mais manter o controle financeiro.

Outra dificuldade é a data de reajuste que deveria ser feita após 6 ou 12 meses de consulta, porém atualmente a profissional não possui nenhum mecanismo que a informe quando o reajuste deve ocorrer, e muitas vezes a data de reajuste passa despercebido, o que traz ônus reais a psicóloga, já que deixa de reajustar o preço dos clientes.

O cadastro de clientes não é um problema hoje, devido a rotatividade de clientes ser pequena, porém não possui uma relação entre os dados de pagamentos, preços, reajustes, o que nos dá uma oportunidade de melhoria, como por exemplo um sistema de recibo, com os

dados cadastrais, o valor de pagamento e a data das consultas, pois hoje esses dados de consultas são pesquisados em sua agenda e os dados do cliente pesquisados em filhas cadastrais físicas, demandando um tempo e esforço desnecessário.

A organização da agenda é algo manual e que não possui backup e nem dá para ser acessada se a agenda física não estiver a mão.

A partir dessa entrevista podemos identificar alguns pontos essenciais do projeto são:

- Controle financeiro;
- Controle de reajuste;
- Gerenciamento de consultas;
- Gerenciamento de clientes.

2.2 REQUISITOS

O levantamento de requisitos é uma etapa muito importante para o bom andamento do projeto e segundo o PMBOK (2017) “O principal benefício deste processo é que o mesmo fornece a base para definição e gerenciamento do escopo do produto e do projeto” (p. 138) .

Para coletar as funcionalidade que a aplicação deve ter utilizamos a técnica entrevista.

> As entrevistas são usadas para obter informações sobre requisitos de alto nível, premissas ou restrições, critérios de aprovação e outras informações de partes interessadas conversando diretamente com elas. (PMBOK, 2017, p. 80).

Os principais pontos identificados na entrevista estão descritos no tópico anterior (2.1) e serão utilizados para montar a matriz de requisitos.

2.2.1 Matriz de requisitos

O PMBOK (2017) define a coleta de requisitos como “(...) o processo de determinar, documentar e gerenciar as necessidades e requisitos das partes interessadas a fim de cumprir os objetivo” e ainda destaca como principal benefício o fornecimento de base para a definição e o gerenciamento de escopo do projeto.

Quadro 1 – Matriz de requisitos

Nome do Projeto:	EMA	**Cliente:**	Psicóloga
Data atualização:	16/09/2018	**Versão:**	1
Responsável:	Thiago Morais		

					Dependências	
ID	**Requisito**	**Descrição do Requisito**	**Tipo Requisito**	**Fornecedor de Requisitos**	**Entre Requisitos**	**Com outros Sistemas**
RE001	Cadastrar usuário	Tela de cadastro usuário. Contendo um formulário com os campos "nome", "email", "senha", "confirmação de senha" e os botões "salvar" e "cancelar".	Funcional	Psicóloga	NA	NA
RE002	Autenticação	Tela de login. Contendo o campo "email" e "senha" e o botão "login".	Funcional	Psicóloga	RE001	NA
RE003	Cadastrar paciente	Tela de cadastro de paciente com os campos "nome", "email", "telefone" e com os botões "salvar" e "cancelar". O cadastro salvará como padrão o "status" do paciente como ativo.	Funcional	Psicóloga	RE002	NA
RE004	Editar de paciente	Tela de edição de cadastro de paciente com os campos "nome", "email", "telefone" e "vencimento" (dia do mês que deverá ocorrer o pagamento da fatura) com os botões "salvar" e "cancelar".	Funcional	Psicóloga	RE003	NA
RE005	Inativar/Ativar paciente	Opção de "inativar" ou "ativar" um paciente.	Funcional	Psicóloga	RE003	NA
RE006	Consultar paciente	Tela que exibe todos os pacientes.	Funcional	Psicóloga	RE003	NA
RE007	Cadastrar preço	Cadastrar preço de um paciente. O cadastro deverá conter o "nome do paciente", a "data" (início da vigência) e o "valor".	Funcional	Psicóloga	RE003	NA
RE008	Consultar preço	Consultar preço de um paciente (vigente e histórico).	Funcional	Psicóloga	RE007	NA
RE009	Gerenciar reajuste de preço	Criar uma regra para avisar quando o preço de um determinado paciente deve sofrer um reajuste.	Funcional	Psicóloga	RE007	NA

Quadro 1 – Matriz de requisitos (Continuação)

					Dependências	
ID	**Requisito**	**Descrição do Requisito**	**Tipo Requisito**	**Fornecedor de Requisitos**	**Entre Requisitos**	**Com outros Sistemas**
RE010	Agendar atendimento	Tela com formulário que cadastre um atendimento, relacionando um paciente e a data/hora do atendimento. Deverá haver uma validação para não ocorrer de dois atendimentos serem agendados para a mesma data/hora. Por padrão o status do atendimento será salvo como "false" e só será alterado para "true" quando o atendimento for efetivada.	Funcional	Psicóloga	RE007	NA
RE011	Editar agendamento	Editar agendamento já cadastrado.	Funcional	Psicóloga	RE010	NA
RE012	Excluir agendamento	Excluir agendamento já cadastrado e com o status "false", não deverá ser possível excluir atendimentos já efetivados (status "true").	Funcional	Psicóloga	RE010	NA
RE013	Efetivar atendimento	Botão que efetiva o atendimento e altera o status do atendimento de "false" para "true". Este botão só deve estar visível quando o atendimento ainda não estiver sido efetivado (status "false"). A efetivação do atendimento gera ou acrescenta valores em uma fatura.	Funcional	Psicóloga	RE010	NA
RE014	Estornar atendimento	Botão que estorna o atendimento e altera o status do atendimento de "true" para "false". Este botão só deve estar visível quando o atendimento ainda já estiver sido efetivado (status "true"). O estorno do atendimento deverá estornar ou diminuir valores na fatura.	Funcional	Psicóloga	RE013	NA
RE015	Consultar atendimento	Tela que consulta todos os atendimentos.	Funcional	Psicóloga	RE010	Google Calendar

Quadro 1 – Matriz de requisitos (Continuação)

					Dependências	
ID	**Requisito**	**Descrição do Requisito**	**Tipo Requisito**	**Fornecedor de Requisitos**	**Entre Requisitos**	**Com outros Sistemas**
RE016	Gerar fatura	A fatura deverá ser cadastrada automaticamente assim que a consulta for efetivada. Uma fatura pode conter vários atendimentos. Os dados para o cadastro da fatura deverão ser: "nome do paciente", "quantidade de consultas", "valor" (preço vigente da consulta x quantidade de consultas), "vencimento".	Funcional	Psicóloga	RE013	NA
RE017	Estornar fatura	A fatura deverá ser estornada automaticamente quando o atendimento for estornado.	Funcional	Psicóloga	RE016	NA
RE018	Consultar fatura	Tela que consulta todas as faturas.	Funcional	Psicóloga	RE016	NA
RE019	Efetuar pagamento	Botão que efetua o pagamento e muda o status da fatura de "false" para "true".	Funcional	Psicóloga	RE016	NA
RE020	Estornar pagamento	Botão deverá estornar o pagamento e muda o status da fatura de "true" para "false".	Funcional	Psicóloga	RE019	NA
RE021	Consultar pagamento	Tela que consulta todas os pagamentos (faturas com status "true").	Funcional	Psicóloga	RE019	NA
RE022	Relatório de faturas em aberto	Apresenta todas as faturas que ainda não foram efetuadas (status "false")	Funcional	Psicóloga	RE016	NA
RE023	Relatório de pagamentos	Apresenta todas os pagamentos (faturas com status "true").	Funcional	Psicóloga	RE019	NA
RE024	Relatório de reajuste	Apresentar a relação de pacientes e a data de reajuste de seus preços.	Funcional	Psicóloga	RE009	NA
RE025	Painel de configurações	O painel deverá possuir um formulário com os dados: "tempo de reajuste", que definirá em quanto tempo deverá ocorrer o próximo reajuste (data de início da vigência de um preço + o "tempo de reajuste").	Funcional	Psicóloga	NA	NA
RE026	Sistema web para dar mobilidade ao usuário	Deverá ser online para que seus dados sejam acessados de qualquer dispositivo com internet.	Não Funcional	desenvolvedor	NA	NA
RE027	Sistema baseado em micro serviço	Para facilitar a integrações futuras.	Não Funcional	desenvolvedor	NA	NA

Fonte: Autor

2.3 DIAGRAMAS

2.3.1 Diagrama de caso de uso

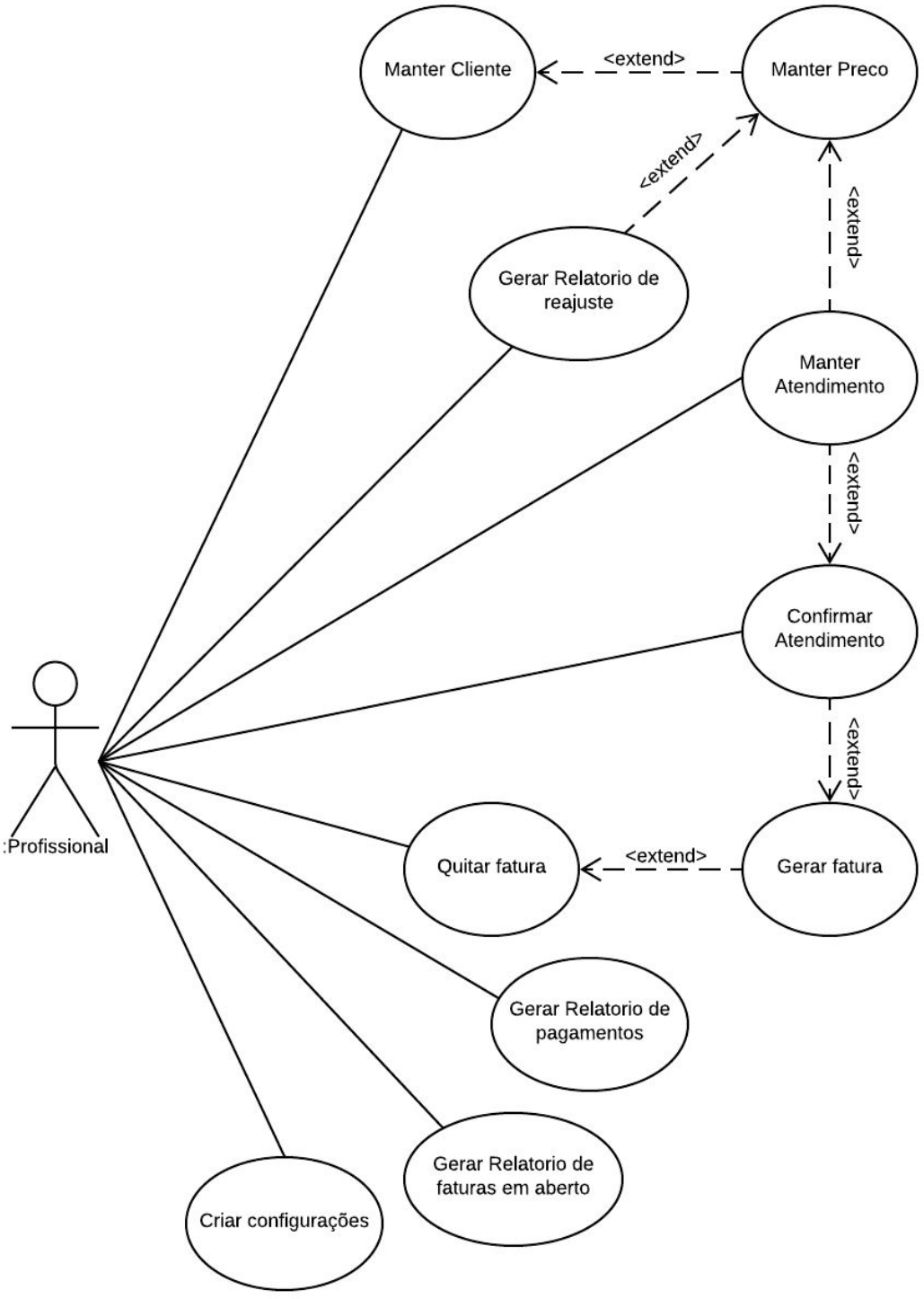
O diagrama de caso de uso tem como objetivo demonstrar as funcionalidade do sistema e como serão suas interações com o usuário.

> Essencialmente, um caso de uso conta uma história estilizada sobre como um usuário final (desempenhando um de uma série de papéis possíveis) interage com o sistema sob um conjunto de circunstâncias específicas. (PRESSMAN, 2011, p. 137).

O diagrama elucida as funcionalidades do sistema de forma simples, dessa forma não é preciso ser técnico para entender o que o digrama quer dizer. Guedes (2011) descreve que o diagrama "Apresenta uma linguagem simples e de fácil compreensão para que os usuários possam ter uma ideia geral de como o sistema irá se comportar".

Para entender melhor esse diagrama devemos saber o que é um ator e qual seu papel nesse contexto. No livro Engenharia de software uma abordagem profissional, o ator é descrito como "qualquer coisa que se comunica com o sistema ou o produto e que é externa ao sistema em si."(PRESSMAN, 2011, p. 138). Em nosso caso de uso teremos apenas 1 ator que será o profissional, que será responsável por todas as interações do sistema.

Figura 1 – Diagrama de caso de uso



Fonte: Autor

2.3.2 Descrição das Regras de Negócio e Validação (RNV)

2.3.2.1 RNV001 - Manter cliente

O cadastro de cliente não possui dependência, dessa forma pode e deve ser feito antes dos demais cadastros**.**

Um cliente possui os atributos nome, telefone e email, sendo todos eles de preenchimento obrigatório. Além do cadastro de novos clientes será permitido a edição dos atributos do cliente, levando em conta a mesma obrigatoriedade de preenchimento de atributos do cadastro.

Nesse projeto optamos pela exclusão lógica, para que o registro do cliente não seja apagado, apenas conste como inativo, dessa forma o profissional terá acesso aos registros de todos os clientes, mesmo que os mesmo não estejam mais sendo atendidos por ele.

2.3.2.2 RNV002 - Manter preço

Para cadastrar um preço é necessário ter um cliente previamente cadastrado (RNV001).

O cadastro de preço deverá ser atrelado a um cliente específico, pois cada cliente terá um preço e uma data de reajuste. Os campos as serem preenchidos para o devido cadastro serão: cliente, valor do atendimento e data de vigência (data que o preço entrará em vigor).

Será permitido excluir um preço cadastrado, isso não afetará os atendimentos, pois o preço será salvo no atendimento.

Também será possível exibir o histórico de preços de um determinado cliente.

2.3.2.3 RNV003 - Manter atendimento

Para cadastrar um atendimento é necessário ter cadastrado o cliente que o profissional pretende atender, além de ter um preço atrelado a esse cliente (RNV001 e RNV 002).

Os dados obrigatórios para o cadastro são: cliente, data e hora do atendimento, por padrão esse atendimento terá o campo status que será preenchido automaticamente como *false* , o que indica que o atendimento não foi efetuado.

O atendimento poderá ser editado e excluído enquanto estiver ainda não estiver sido efetivado (*status=false*).

2.3.2.4 RNV004 - Confirmar atendimento

Após o atendimento ocorrer o profissional deverá efetivá-lo no sistema, a fim de gerar a fatura.

O sistema possuirá um botão "efetivar atendimento" que troca o status de *false* para *true* e só estará visível quando o status do atendimento for *false*.

Já o botão "estornar atendimento" troca o status de *true* para *false* e só estará visível quando o status do atendimento for *true*.

Não será possível estornar um atendimento que tenha sua fatura quitada (*status=true*).

2.3.2.5 RNV005 - Gerar fatura

Será registrada de forma automática, assim que a fatura for efetivada (*status=true*), dessa forma podemos inferir que não haverão faturas sem atendimentos.

A fatura será gerada com seu status *false* por padrão, dessa forma podemos concluir que a fatura recém criada consta em aberto.

2.3.2.6 RNV006 - Quitar fatura

Após o pagamento da fatura o profissional deverá quitá-la para que a mesma não conste mais com fatura em aberto.

O botão "quitar" troca o status de *false* para *true* e só estará visível quando o status da fatura for *false*.

Já o botão "estornar quitação" troca o status de *true* para *false* e só estará visível quando o status da fatura for *true*.

2.3.2.7 RNV007 - Criar configurações

A funcionalidade de configurações foi criada para que o sistema seja flexível a mudanças, dessa forma o próprio profissional.

O usuário poderá configurar o tempo de seus atendimentos, o que facilitará quando o mesmo for agendar um atendimento, já que não irá necessitar preencher o horário de término deste atendimento, pois o parâmetro inserido na configuração irá se encarregar disso.

Já o intervalo entre os reajustes de preço irá se tornar parâmetro para o relatório de reajustes.

2.3.2.8 RNV008 - Gerar relatório de reajuste

Será possível extrair um relatórios de reajuste de preço, para que o profissional tenha uma base da data de reajuste de preço de seus clientes de acordo com um período que será configurável de forma global (RNV007), ou seja, a configuração será válida para todos os clientes. Este relatório irá calcular a data do último reajuste feito ao preço de um cliente mais o período configurado como intervalo entre os reajustes e irá exibir a data que o profissional deverá realizar a próxima alteração de preço para um determinado cliente.

Figura 2 – Relatório de reajuste de preço

Fonte: Autor

2.3.2.9 RNV009 - Gerar relatório de faturas em aberto

Este relatório irá exibir todas as faturas que estão em aberto até o momento. São consideradas faturas em aberto aquelas faturas que ainda não possuem pagamento, ou seja, que seu status é igual a *false*.

Figura 3 – Relatório de faturas em aberto

Fonte: Autor

2.3.2.10 RNV0010 - Gerar relatório de pagamentos

Este relatório irá exibir todas as faturas que estão em quitadas até o momento. São consideradas faturas quitadas aquelas faturas que já possuem pagamento, ou seja, que seu status é igual a *true*.

Figura 4 – Relatório de pagamentos

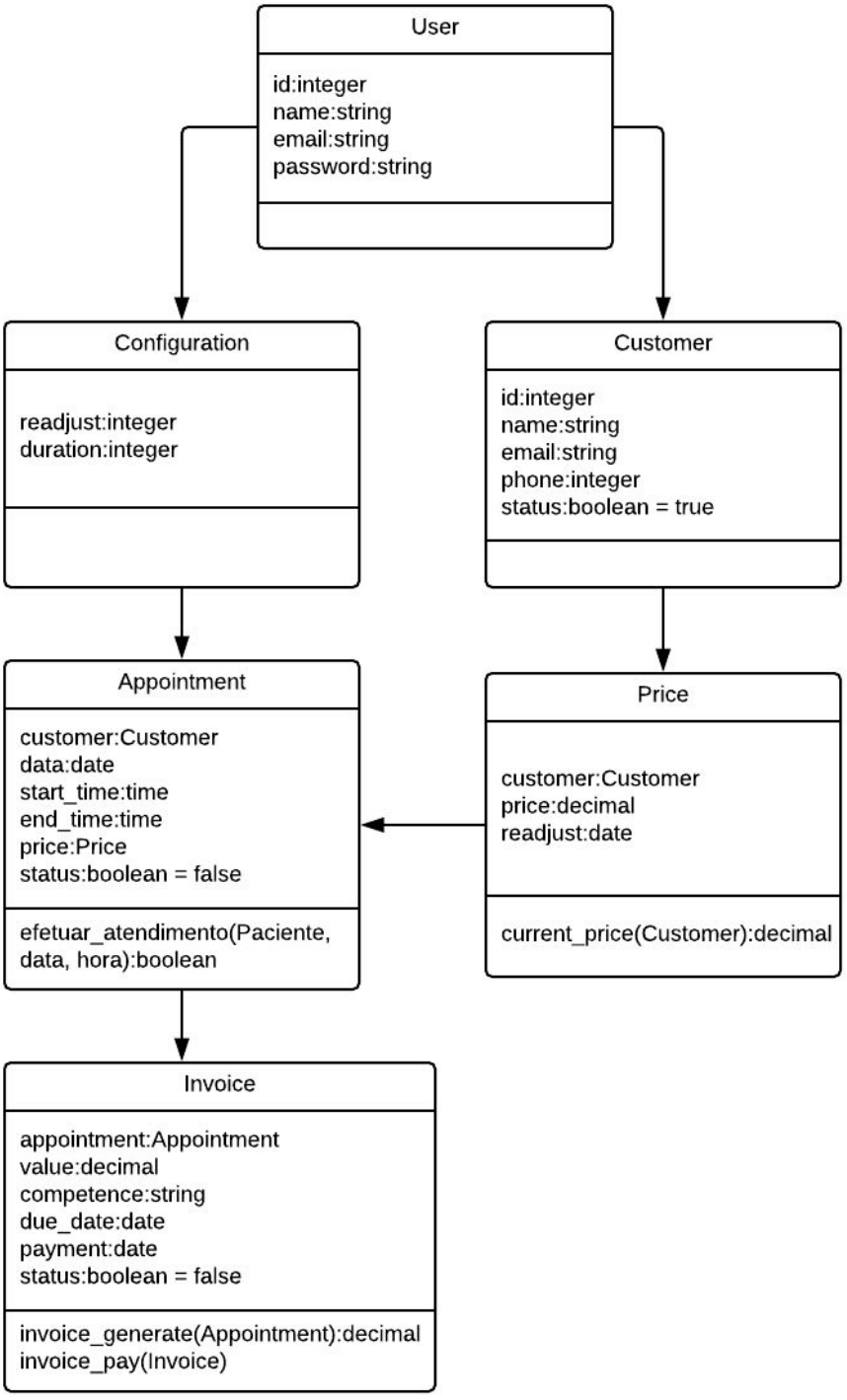
Fonte: Autor

2.3.3 Diagrama de classes

O diagrama de classes mostra a estrutura do sistema a partir de suas classes, atributos e métodos e com elas se relacionam.

> Como o próprio nome diz, define a estrutura das classes utilizadas pelo sistema, determinando os atributos e métodos que cada classe tem, além de estabelecer como as classes se relacionam e trocam informações entre si. (GUEDES, 2011, p. 31)

Figura 5 – Diagrama de classes

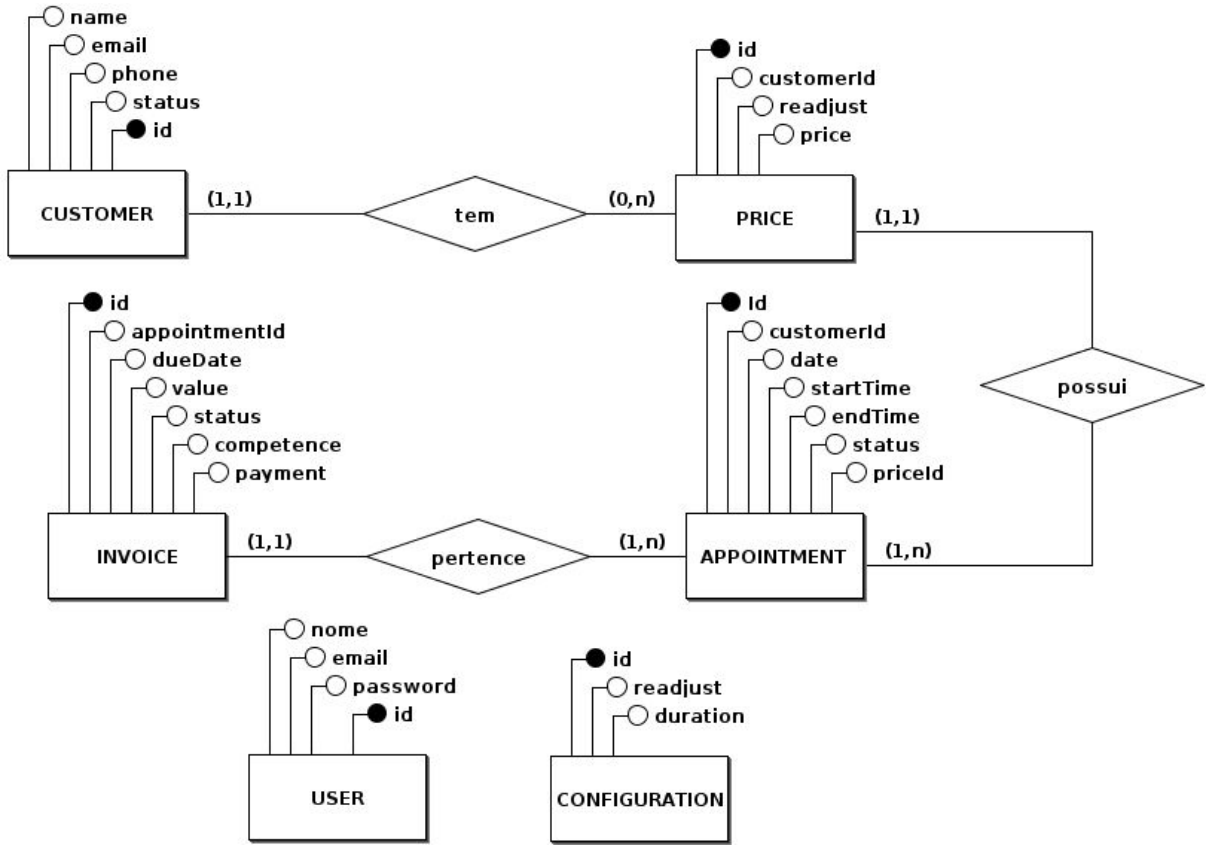


Fonte: Autor

2.3.4 Modelo Entidade Relacionamento

O modelo conceitual do banco de dados demonstra como os dados serão armazenados, além descrever suas tabelas, atributos e seus relacionamentos. “O modelo conceitual registra que dados podem aparecer no banco de dados, mas não registra como estes dados estão armazenados a nível de SGBD.” (HEUSER, 1998, p. 5)

Figura 6 – Modelo Entidade Relacionamento Conceitual

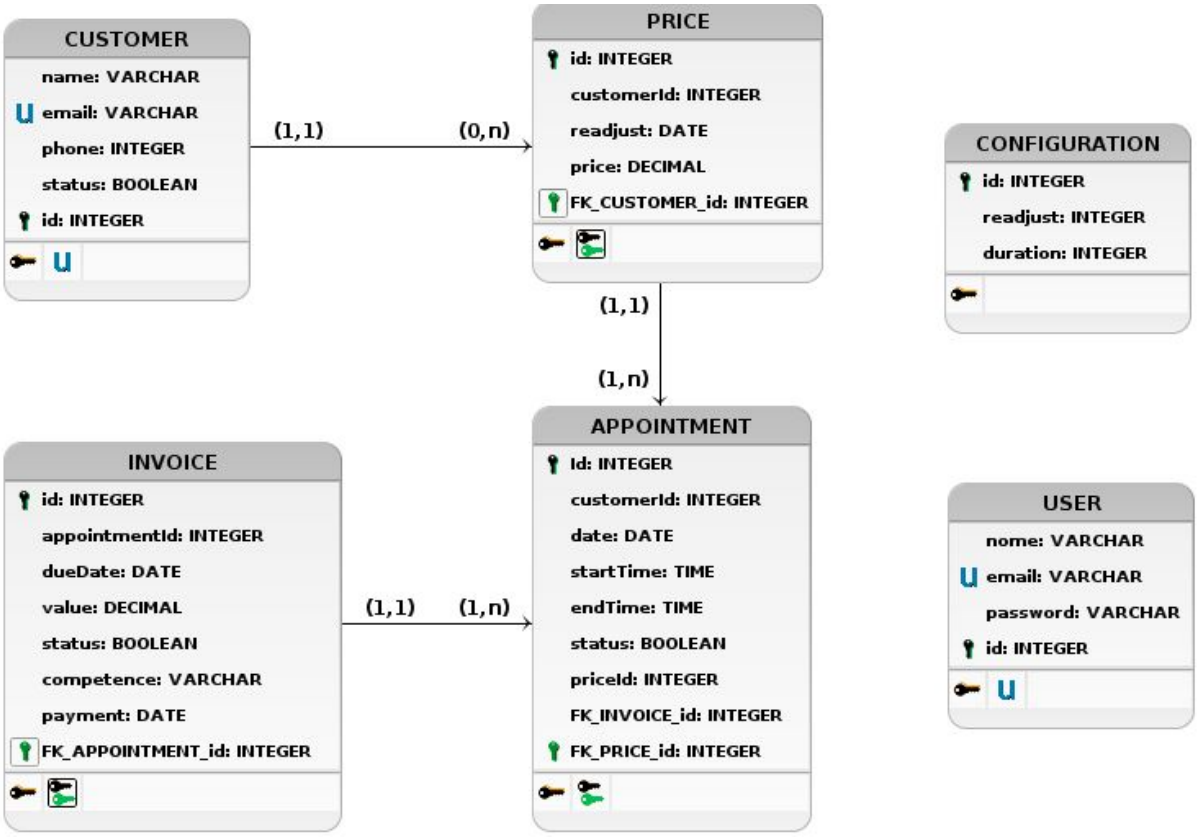


Fonte: Autor

O modelo lógico é baseado no modelo conceitual, porém no modelo lógico o sistema de gerenciamento de banco de dados é levado em conta.

> Um modelo lógico é uma descrição de um banco de dados no nível de abstração visto pelo usuário do SGBD. Assim, o modelo lógico é dependente do tipo particular de SGBD que está sendo usado. (HEUSER, 1998, p. 6)

Figura 7 – Modelo Lógico



Fonte: Autor

3 DESENVOLVIMENTO SISTEMA/SOFTWARE

Neste capítulo abordaremos as ferramentas utilizadas e as boas práticas adotadas para para desenvolvimento do software.

3.1 FERRAMENTAS

3.1.1 Linguagem de programação

A linguagem de programação é uma escrita específica que define um conjunto de instruções para construção de softwares. Esses softwares podem ser executados em computadores, TV, celulares e diversos outro dispositivos. Logo podemos dizer que a linguagem de programação é uma forma de comunicação, de forma que o homem consiga delegar instruções para a máquina.

Neste trabalho faremos utilização de duas linguagens de programação, a fim de concluir o objetivo de criar um programa capaz de tratar as informações inseridas pelo usuário. As linguagens utilizadas serão: Ruby e Javascript.

3.1.1.1 Ruby

Figura 8 – Logotipo Ruby

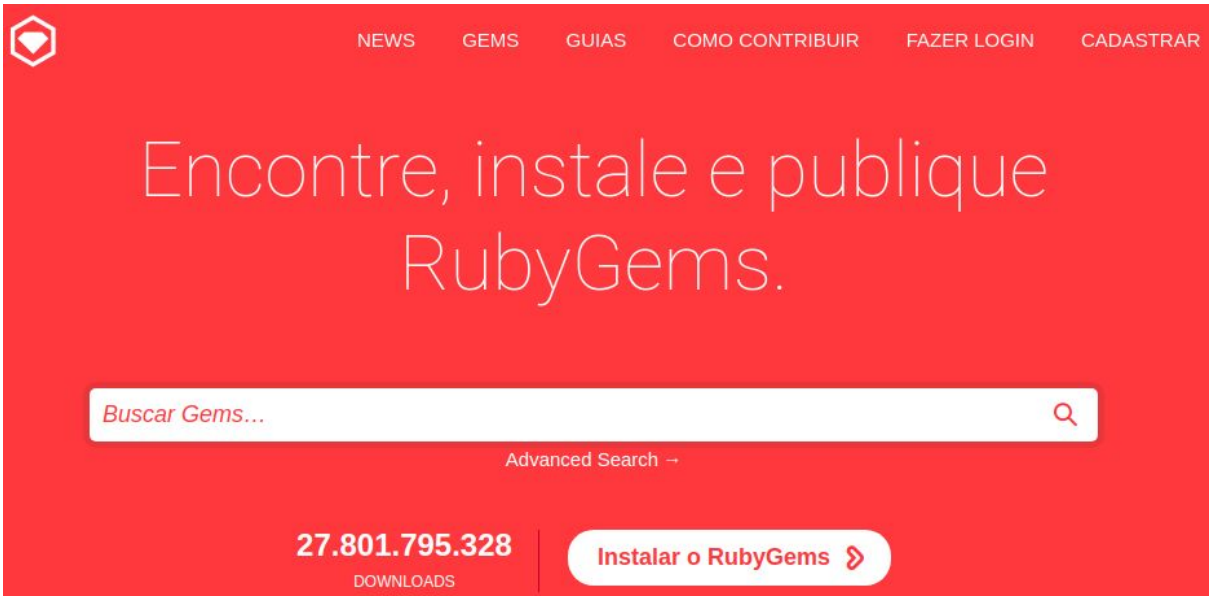
Fonte: https://fuzati.com/technology/ruby-logo/

O Ruby foi apresentado para o público em 1995 e foi criada por Yukihiro “Matz” Matsumoto. A linguagem é uma mistura de algumas linguagens favoritas de Matz (Perl,

Smalltalk, Eiffel, Ada e Lisp) para formar uma nova linguagem que equilibra a programação funcional com a programação imperativa.

A linguagem também conta com um sistema de Gems que funciona como uma espécie de *plugins*, ou incorporação de pequenas funcionalidades em seus projetos. As Gems tem como objetivo agregar pequenas funcionalidade que podem ser acrescentadas ao projeto com poucas linhas e configurações. Um exemplo de funcionalidade que uma Gem pode trazer é o sistema de login, como a Gem chamada Devise, desenvolvida e mantida pela empresa Plataformatec. Com o sistema de Gems podemos nos concentrar nossos esforços nas regras de negócio de nossos clientes. Outra vantagens do sistema de Gems é que a comunidade ou empresa que desenvolve uma Gem costuma mantê-la atualizada, o que facilita a manutenção da aplicação. O catalogo de Gems pode ser encontrado em https://rubygems.org/.

Figura 9 – Print screen Ruby Gem



Fonte: https://rubygems.org/

A escolha desta linguagem se deu pelo fácil aprendizado e por ser uma linguagem simples e bem documentada.

3.1.1.2 Javascript

Figura 10 – Logotipo Javascript

Fonte: http://pluspng.com/png-97955.html

O JavaScript teve sua primeira versão apresentada em 1995. Desenvolvida por Brendan Eich, foi implementada como parte do navegador Netscape 2.0. A linguagem foi originalmente criada para ser parte dos navegadores, para que os scripts pudessem ser executados do lado do cliente e pudessem interagir com o usuário sem a necessidade de passar o processamento para o servidor, podendo assim controlar o conteúdo do navegador, realizar alterações assíncronas e alterar o que for necessário na tela do usuário.

Atualmente essa linguagem é a mais comum nos navegadores web. Porém começa a ser bastante utilizada no lado do servidor através de ambientes como NodeJs.

Sendo esta uma linguagem multiparadigma, possui suporte a orientação a objetos, funcional, imperativa, tipagem dinâmica, o JavaScript vem crescendo no mercado com a possibilidade de ser utilizado fora do navegador. Surgem então frameworks e bibliotecas para auxiliar a utilização da linguagem, dentre eles Angular, Vue.js, NodeJs e ReactJs.

A utilização se deu pelo aumento da popularidade da linguagem no mercado de trabalho bem como pela fácil utilização e aprendizado.

3.1.2 Framework

Para execução do projeto foram utilizados o Rails como frameworks, para adoção de boas práticas, padronização, praticidade e redução de tempo de desenvolvimento.

> O framework é um conjunto de classes que colaboram entre si proporcionando melhores práticas de desenvolvimento e diminuição à repetição de tarefas. Além disso, evita variações de "soluções diferentes para um mesmo tipo de problema. (SANTOS; CARVALHO apud ALVIM , 2010, p. 12)

3.1.2.1 Rails

Figura 11 – Logotipo Rails

Fonte: https://rubyonrails.org/images/rails-logo.svg

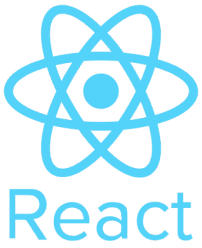
O framework trabalha com a abordagem *Model View Controller* (MVC), que proporciona melhor organização no desenvolvimento, além de se utilizar de várias formas de abstração, como por exemplo a de banco de dados, dessa forma o usuário interage com o banco de dados utilizando uma linguagem de programação, nesse caso Ruby, dessa forma o desenvolvedor não necessita saber instruções de banco de dados, como por exemplo *Structured Query Language* (SQL), o que facilita o desenvolvimento e manutenção do código.

O Rails foi escolhido pela documentação boa documentação, que atende bem às necessidades do projeto e também pela fácil implementação de projetos Ruby.

A documentação oficial do Rails pode ser encontra em https://guides.rubyonrails.org/.

3.1.2.2 React

Figura 12 – Logotipo React



Fonte: https://ubisafe.org/explore/svg-logo-reactjs/

Criado pelo Facebook em 2013, o React é uma biblioteca JavaScript de código aberto que visa auxiliar no desenvolvimento de interfaces com a utilização de componentes.
Por ser de código aberto, outras empresas além do Facebook, como Netflix e Walmart, além de uma comunidade de desenvolvedores ajudam a manter a biblioteca. Por ter uma gama de plugins, componentes e funcionalidades, além de um módulo criado especificamente para desenvolvimento mobile, o React Native, essa biblioteca muitas vezes vem a ser confundida com um framework.

3.1.4 Versionamento

Figura 13 – Logotipo Git

Fonte: https://git-scm.com/downloads/logos

Ao longo do processo de desenvolvimento um projeto passa por várias mudanças, desde novas funcionalidades até correção de pequenos erros. Para reduzir riscos e controlar as alterações feitas, faremos uso de um sistema de controle de versionamento (SCV).

Existem diversas ferramentas de versionamento, entre as mais populares, que possuem versão gratuita, estão as ferramentas baseadas em GIT, como o Github e o Gitlab.

Figura 14 – Logotipo Github

Fonte: https://github.com/logos

Para este projeto foi escolhido o Github como repositório levando em conta a praticidade na integração com a plataforma *cloud* Heroku, uma ferramenta *plataform as a service* (PAAS).

Endereço dos repositórios:

- API : https://github.com/thiagolmorais/ema_api
- Aplicação : https://github.com/Renanpinto/secretariavirtual-frontend

3.2 MICRO SERVIÇO

A aplicação foi desenvolvida com base no conceito de micro serviços.

Segundo Lewis e Fowler (2014) o termo "Microservice Architecture" surgiu para descrever uma maneira de conceber aplicações de software como um conjunto de serviços que podem ser implementadas de forma independente.

Segundo Gambarra (2017), não há uma definição formal de micro serviços, porém existem certas características que guiam essa arquitetura. A principal ideia é construir pequenos serviços, independentes e focados na resolução de um único problema dentro de uma aplicação.

> A arquitetura de micro serviços é um modelo arquitetônico, que aborda o desenvolvimento de uma aplicação composta por vários serviços, onde cada serviço é autônomo, ou seja, independente dos demais, e responsável por um conjunto finito de funcionalidades. (MALIPENSE; ZUCHI, 2018)

Embora não houvesse necessidade dessa abordagem, devido ao tamanho do projeto, decidimos utilizá-la para que o projeto tivesse baixo acoplamento, fazendo com que a modularização e crescimento do mesmo seja fácil no futuro.

3.2.1 Api

API (*Application Programming Interface*), possibilita que uma aplicação interaja com outras através de *endpoints* que fornecem dados em formato Json. A api possibilita que uma única aplicação armazene e até mesmo trate os dados de várias outras aplicações, ou ainda, que uma aplicação receba dados de várias aplicações.

No projeto EMA usaremos um padrão de api chamado RESTfull, que consiste em *endpoints* que podem ter apenas uma rota para efetuar vários tipos de ações, como por exemplo: a rota "/users" pode criar, editar, deletar e consultar um usuário ou vários. A api distingue essas funções a partir dos verbos HTTP.

Quadro 2 – Verbos HTTP

Endpoint	Método	Ação
/users	GET	Retorna a lista de usuários
/users	POST	Insere um novo usuário
/users/{id}	GET	Retorna o usuário com id = {id}
/users/{id}	PUT	Substitui os dados do usuário com id = {id}
/users/{id}	PATCH	Altera items dos dados do usuário com id = {id}
/users/{id}	DELETE	Remove o usuário com id = {id}

Fonte: https://blog.mbeck.com.br/api-rest-e-os-verbos-http-46e189085e21

Em nosso projeto esses endpoints trafegam dados em formato *JavaScript Object Notation* (JSON), que trata dados de forma simples através de parâmetros com chave e valor.

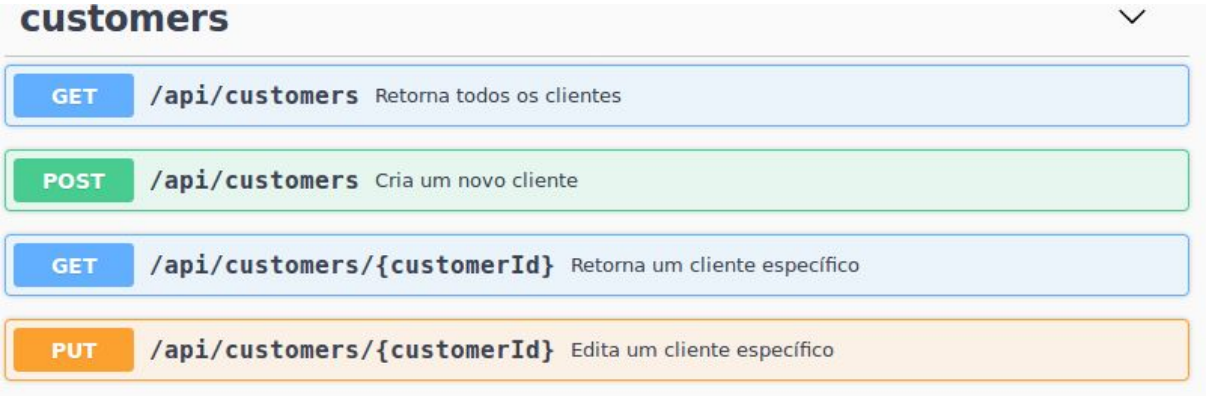
3.2.2 Swagger

Figura 15 – Logotipo Swagger

Fonte: https://swagger.io/swagger/media/assets/images/swagger_logo.svg

Para documentar a api e todos seus *endpoints* e *payloads* utilizamos uma ferramenta chamada Swagger e está disponível em: https://app.swaggerhub.com/apis/Fatec/EMA_api/v1.

Figura 16 – Customers endpoints



Fonte: Autor

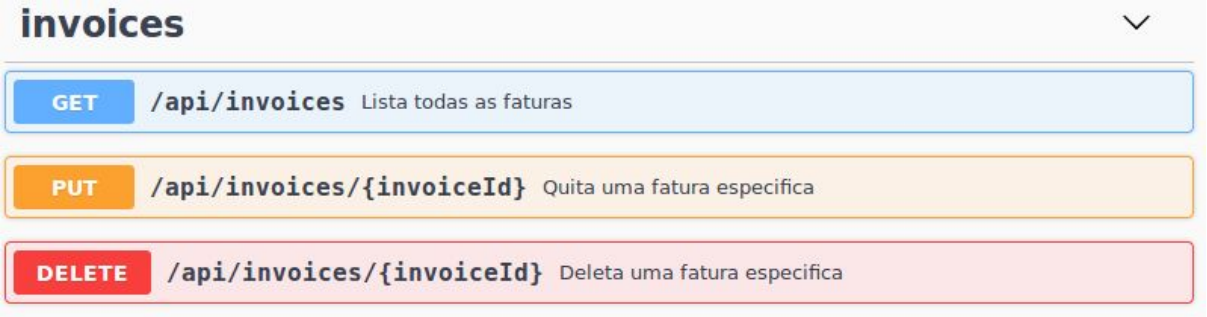
Figura 17 – Prices endpoints

Fonte: Autor

Figura 18 – Appointments endpoints

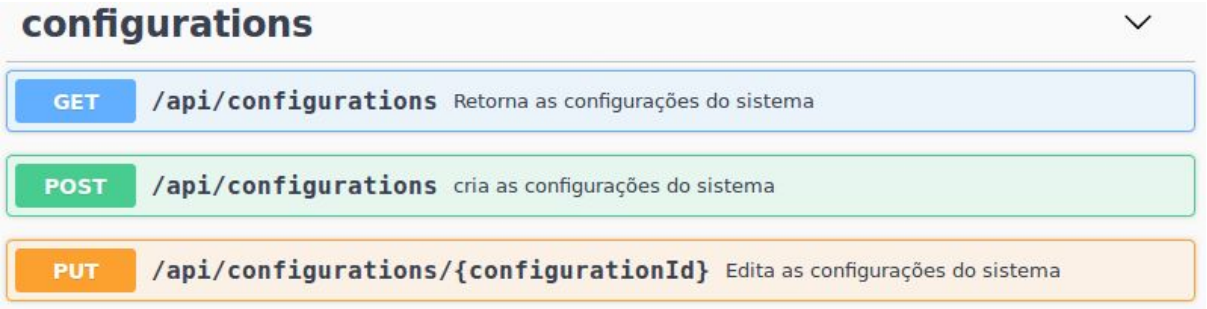
Fonte: Autor

Figura 19 – Invoices endpoints



Fonte: Autor

Figura 20 – Configurations endpoints



Fonte: Autor

3.3 AMBIENTE DE HOMOLOGAÇÃO

Muitas vezes alguns erros que ocorrem em produção não são possíveis serem reproduzidos em ambiente de desenvolvimento, devido diferença nas configurações. Por essa razão o ambiente de homologação é muito importante, pois possui configurações igual ou que se assemelha ao ambiente de produção, sendo utilizado para os testes finais antes de colocar a nova versão para produção.

Neste projeto escolhemos o Heroku, como serviço para homologar nossa aplicação. Um critério para a escolha dessa plataforma foi a praticidade, pois a plataforma possui integração com o Github, facilitando o deploy automático, dessa forma toda vez que a branch escolhida, no nosso caso a "master", é atualizada o projeto é atualizado também no ambiente de homologação. Além da plataforma possuir uma versão gratuita.

Figura 21 – Logotipo Heroku

Fonte: https://www.heroku.com/

- Api: ema-api.herokuapp.com/api/
- Aplicação: http://ema-secretariavirtual.herokuapp.com/

3.3.1 Testes

Para auxiliar na elaboração dos testes foi criada uma matriz de testes, que será demonstrada a seguir.

Quadro 3 – Matriz de teste 01

ID	1
Módulo	Cadastro de usuário
Descrição	Cadastrar um usuário
Roteiro	1) Clique em CADASTRAR. 2) Preencha os campos: Usuário, Email, senha e confirmação de senha. 3) Clique em CADASTRAR.
Resultado esperado	Confirmado o cadastro o usuário será direcionado para a home page
Data	22/05/2018
Analista	Thiago

Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 4 – Matriz de teste 02

ID	2
Módulo	Cadastro de usuário
Descrição	Cadastrar um usuário com senha menor que 6 caracteres
Roteiro	1) Clique em CADASTRAR. 2) Preencha os campos: Usuário, Email, senha e confirmação de senha (com menos de 6 caracteres). 3) Clique em CADASTRAR.
Resultado esperado	Deverá aparecer uma mensagem de erro e continuar na tela de cadastro
Data	22/05/2018
Analista	Thiago
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 5 – Matriz de teste 03

ID	3
Módulo	Login
Descrição	Efetuar o login com a senha correta
Roteiro	1) Preencha os campos: Email e senha. 2) Clique em ENTRAR.
Resultado esperado	O usuário deverá ser redirecionado para home page
Data	22/05/2018
Analista	Thiago
Resultado do teste	Executado com sucesso

Correções / Melhorias	

Fonte: Autor

Quadro 6 – Matriz de teste 04

ID	4
Módulo	Login
Descrição	Efetuar o login com a senha incorreta
Roteiro	1) Preencha os campos: Email e senha (incorreta). 2) Clique em ENTRAR.
Resultado esperado	Deverá aparecer uma mensagem de erro e continuar na tela de login
Data	22/05/2018
Analista	Thiago
Resultado do teste	Erro. O usuário não é autenticado e continua na tela de Login, porém não há mensagem de erro
Correções / Melhorias	

Fonte: Autor

Quadro 7 – Matriz de teste 05

ID	5
Módulo	Cadastro de cliente
Descrição	Cadastrar um cliente com todos os dados preenchidos
Roteiro	1) Se autenticar na aplicação. 2) Clique em CLIENTE e em seguida em CADASTRAR CLIENTE. 3) Preencha os campos: Nome, Email e Telefone. 4) Clique em SALVAR.
Resultado esperado	Os dados deverão ser salvos com sucesso e apresentar uma mensagem de sucesso ao usuário
Data	22/05/2018
Analista	Thiago

Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 8 – Matriz de teste 06

ID	6
Módulo	Cadastro de cliente
Descrição	Cadastrar um cliente com dados em branco
Roteiro	1) Se autenticar na aplicação. 2) Clique em CLIENTE e em seguida em CADASTRAR CLIENTE. 3) Deixe em branco os campos: Nome, Email e Telefone. 4) Clique em SALVAR.
Resultado esperado	Dados não deverão ser salvos e deverá apresentar a descrição dos campos inválidos
Data	22/05/2018
Analista	Thiago
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 9 – Matriz de teste 07

ID	7
Módulo	Consultar cliente
Descrição	Exibir clientes cadastrados
Roteiro	1) Se autenticar na aplicação. 2) Clique em CLIENTE e em seguida em CONSULTAR CLIENTE.
Resultado esperado	Deverá exibir os clientes cadastrados, seu email e seu status
Data	22/05/2018
Analista	Thiago
Resultado do teste	Executado com sucesso

Correções / Melhorias	

Fonte: Autor

Quadro 10 – Matriz de teste 08

ID	8
Módulo	Consultar cliente
Descrição	Consultar os detalhes de um cliente cadastrado
Roteiro	1) Se autenticar na aplicação. 2) Clique em CLIENTE e em seguida em CONSULTAR CLIENTE. 3) Clique em CONSULTAR.
Resultado esperado	Deverá aparecer preenchido todos os dados do cliente selecionado
Data	22/05/2018
Analista	Thiago
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 11 – Matriz de teste 09

ID	9
Módulo	Edição de cliente
Descrição	Editar um cliente com todos os dados preenchidos
Roteiro	1) Se autenticar na aplicação. 2) Clique em CLIENTE e em seguida em CONSULTAR CLIENTE. 3) Clique em CONSULTAR. 4) Clique em EDITAR. 5) Altere alguns dos campos: Nome, Email e Telefone.
Resultado esperado	Os dados deverão ser salvos com sucesso
Data	22/05/2018
Analista	Thiago

Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 12 – Matriz de teste 10

ID	10
Módulo	Edição de cliente
Descrição	Editar um cliente com dados em branco
Roteiro	1) Se autenticar na aplicação. 2) Clique em CLIENTE e em seguida em CONSULTAR CLIENTE. 3) Clique em CONSULTAR. 4) Clique em EDITAR. 5) Apague algum dos campos: Nome, Email ou Telefone.
Resultado esperado	Dados não deverão ser salvos e deverá apresentar a descrição dos campos inválidos
Data	22/05/2018
Analista	Vinicius
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 13 – Matriz de teste 11

ID	11
Módulo	Cadastrar preço
Descrição	Cadastrar preço com todos os dados preenchidos
Roteiro	1) Se autenticar na aplicação. 2) Clique em CLIENTE e em seguida em CONSULTAR CLIENTE. 3) Clique em CONSULTAR. 4) Clique em CADASTRAR PREÇOS. 5) Preencha os campos: Valor e Reajuste. 6) Clique em GRAVAR.
Resultado esperado	Os dados deverão ser salvos com sucesso

Data	22/05/2018
Analista	Vinicius
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 14 – Matriz de teste 12

ID	12
Módulo	Cadastrar preço
Descrição	Cadastrar preço com dados em branco
Roteiro	1) Se autenticar na aplicação. 2) Clique em CLIENTE e em seguida em CONSULTAR CLIENTE. 3) Clique em CONSULTAR. 4) Clique em CADASTRAR PREÇOS. 5) Deixe os dados dos seguintes campos em branco: Valor e Reajuste. 6) Clique em GRAVAR.
Resultado esperado	Dados não deverão ser salvos e deverá apresentar a descrição dos campos inválidos
Data	22/05/2018
Analista	Vinicius
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 15 – Matriz de teste 13

ID	13
Módulo	Consultar preço
Descrição	Consultar preço cadastrado
Roteiro	1) Se autenticar na aplicação. 2) Clique em CLIENTE e em seguida em CONSULTAR CLIENTE. 3) Clique em CONSULTAR. 4) Clique em EXIBIR PREÇOS.
Resultado esperado	Deverá aparecer o histórico de preços e o preço vigente em destaque
Data	22/05/2018

Analista	Vinicius
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 16 – Matriz de teste 14

ID	14
Módulo	Cadastrar atendimentos
Descrição	Cadastrar um atendimento preenchendo todos os dados
Roteiro	1) Se autenticar na aplicação. 2) Clique em ATENDIMENTO e em seguida em CADASTRAR ATENDIMENTO. 3) Preencha os campos: Cliente, Data e Hora. 4) Clique em GRAVAR.
Resultado esperado	Os dados deverão ser salvos com sucesso
Data	22/05/2018
Analista	Vinicius
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 17 – Matriz de teste 15

ID	15
Módulo	Cadastrar atendimentos
Descrição	Cadastrar um atendimento com os dados em branco
Roteiro	1) Se autenticar na aplicação. 2) Clique em ATENDIMENTO e em seguida em CADASTRAR ATENDIMENTO. 3) Deixe os dados dos seguintes campos em branco: Data e Hora. 4) Clique em GRAVAR.
Resultado esperado	Dados não deverão ser salvos e deverá apresentar a descrição dos campos inválidos
Data	22/05/2018

Analista	Vinicius
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 18 – Matriz de teste 16

ID	16
Módulo	Cadastrar atendimentos
Descrição	Cadastrar atendimentos no mesmo intervalo de tempo
Roteiro	1) Se autenticar na aplicação. 2) Clique em ATENDIMENTO e em seguida em CADASTRAR ATENDIMENTO. 3) Preencha os campos: Cliente, Data e Hora. 4) Clique em GRAVAR.
Resultado esperado	Deverá aparecer uma mensagem de erro e o atendimento não deverá ser gravada
Data	22/05/2018
Analista	Vinicius
Resultado do teste	Erro. É possível cadastrar o atendimento durante o intervalo de atendimento de outro cliente
Correções / Melhorias	

Fonte: Autor

Quadro 19 – Matriz de teste 17

ID	17
Módulo	Calendário
Descrição	Exibir as atendimentos de forma mais amigável
Roteiro	1) Se autenticar na aplicação. 2) Clique em ATENDIMENTO e em seguida em CALENDÁRIO.
Resultado esperado	Deverá exibir um calendário com os atendimentos agendadas.
Data	22/05/2018

Analista	Vinicius
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 20 – Matriz de teste 18

ID	18
Módulo	Editar atendimento via calendário
Descrição	Editar um atendimento preenchendo todos os dados
Roteiro	1) Se autenticar na aplicação. 2) Clique em ATENDIMENTO e em seguida em CALENDÁRIO. 3) Clique em um atendimento. 4) Edite os campos: Cliente, Data ou Hora. 5) Clique em GRAVAR.
Resultado esperado	Os dados deverão ser salvos com sucesso
Data	22/05/2018
Analista	Vinicius
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 21 – Matriz de teste 19

ID	19
Módulo	Editar atendimento via calendário
Descrição	Editar um atendimento com os dados em branco
Roteiro	1) Se autenticar na aplicação. 2) Clique em ATENDIMENTO e em seguida em CALENDÁRIO. 3) Clique em um atendimento. 4) Deixe os seguintes campos em branco: Data e Hora. 5) Clique em GRAVAR.
Resultado esperado	Dados não deverão ser salvos e deverá apresentar a descrição dos campos inválidos
Data	22/05/2018

Analista	Renan
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 22 – Matriz de teste 20

ID	20
Módulo	Editar atendimento
Descrição	Editar um atendimento preenchendo todos os dados
Roteiro	1) Se autenticar na aplicação. 2) Clique em ATENDIMENTO e em seguida em EXIBIR ATENDIMENTOS. 3) Clique no campo dia em um atendimento. 4) Clique em EDITAR. 5) Edite os campos: Cliente, Data ou Hora. 6) Clique em GRAVAR.
Resultado esperado	Os dados deverão ser salvos com sucesso
Data	22/05/2018
Analista	Renan
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 23 – Matriz de teste 21

ID	21
Módulo	Editar atendimento
Descrição	Editar um atendimento com os dados em branco
Roteiro	1) Se autenticar na aplicação. 2) Clique em ATENDIMENTO e em seguida em EXIBIR ATENDIMENTOS. 3) Clique no campo dia em um atendimento. 4) Clique em EDITAR. 5) Deixe os seguintes campos em branco: Data e Hora. 6) Clique em GRAVAR.
Resultado esperado	Dados não deverão ser salvos e deverá apresentar a descrição dos campos

	inválidos
Data	22/05/2018
Analista	Renan
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 24 – Matriz de teste 22

ID	22
Módulo	Editar atendimento
Descrição	Editar atendimento com status efetivado
Roteiro	1) Se autenticar na aplicação. 2) Clique em ATENDIMENTO e em seguida em EXIBIR ATENDIMENTOS. 3) Clique no campo dia em um atendimento. 4) Clique em EDITAR. 5) Edite os campos: Cliente, Data ou Hora. 6) Clique em GRAVAR.
Resultado esperado	Não deve ser possível editar o atendimento após mudança para o status "Atendido"
Data	22/05/2018
Analista	Renan
Resultado do teste	Erro. O atendimento é alterado.
Correções / Melhorias	

Fonte: Autor

Quadro 25 – Matriz de teste 23

ID	23
Módulo	Exibir atendimento
Descrição	Exibir todas os atendimentos cadastrados
Roteiro	1) Se autenticar na aplicação. 2) Clique em ATENDIMENTO e em seguida em EXIBIR ATENDIMENTOS.
Resultado esperado	Deverá aparecer todos os atendimentos cadastrados com as informações de data, hora, cliente e status
Data	22/05/2018

Analista	Renan
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 26 – Matriz de teste 24

ID	24
Módulo	Exibir atendimento
Descrição	Exibir detalhes de um atendimento cadastrado
Roteiro	1) Se autenticar na aplicação. 2) Clique em CONSULTA e em seguida em EXIBIR CONSULTAS. 3) Clique no campo dia em um atendimento.
Resultado esperado	Deverá exibir os detalhes de uma determinada consulta de data, hora, cliente e status
Data	22/05/2018
Analista	Renan
Resultado do teste	Erro. Os dados Cliente, Data e Hora são apresentados, porém o Status não
Correções / Melhorias	

Fonte: Autor

Quadro 27 – Matriz de teste 25

ID	25
Módulo	Atendimentos
Descrição	Efetuar consulta em cliente que tenha preço vigente cadastrado na data do atendimento
Roteiro	1) Se autenticar na aplicação. 2) Clique em ATENDIMENTO e em seguida em EXIBIR ATENDIMENTOS. 3) Clique em ATENDER.
Resultado esperado	O status do atendimento em questão deverá mudar de "Agendado" para "Atendido"
Data	22/05/2018

Analista	Renan
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 28 – Matriz de teste 26

ID	26
Módulo	Atendimentos
Descrição	Efetuar atendimento em cliente que não tenha preço vigente cadastrado na data do atendimento
Roteiro	1) Se autenticar na aplicação. 2) Clique em ATENDIMENTO e em seguida em EXIBIR ATENDIMENTOS. 3) Clique em ATENDER.
Resultado esperado	O usuário deverá ser redirecionado para o cadastro de preço do cliente em questão e deverá apresentar uma notificação ao usuário
Data	22/05/2018
Analista	Renan
Resultado do teste	Erro. O usuário é redirecionado para a tela de cadastro de preço, porém não é exibida nenhuma notificação
Correções / Melhorias	Atributo Status inserido nos detalhes do cliente

Fonte: Autor

Quadro 29 – Matriz de teste 27

ID	27
Módulo	Atendimentos
Descrição	Cancelar atendimento
Roteiro	1) Se autenticar na aplicação. 2) Clique em ATENDIMENTO e em seguida em EXIBIR ATENDIMENTOS. 3) Clique em CANCELAR FATURA.
Resultado esperado	O status do atendimento em questão deverá mudar de "Atendido" para "Agendado"
Data	22/05/2018

Analista	Renan
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 30 – Matriz de teste 28

ID	28
Módulo	Atendimentos
Descrição	Cancelar atendimento
Roteiro	1) Se autenticar na aplicação. 2) Clique em ATENDIMENTO e em seguida em EXIBIR ATENDIMENTOS. 3) Clique em CANCELAR FATURA.
Resultado esperado	Os dados devem ser deletados do banco de dados e o atendimento não deverá mais ser exibida
Data	22/05/2018
Analista	Vitor
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 31 – Matriz de teste 29

ID	29
Módulo	Faturas
Descrição	Exibir faturas
Roteiro	1) Se autenticar na aplicação. 2) Clique em FATURAS.
Resultado esperado	Ao efetuar um atendimento o valor do mesmo deverá ser adicionado ao valor em aberto do cliente em questão
Data	22/05/2018
Analista	Vitor
Resultado do teste	Executado com sucesso

Correções / Melhorias	

Fonte: Autor

Quadro 32 – Matriz de teste 30

ID	30
Módulo	Faturas
Descrição	Exibir faturas
Roteiro	1) Se autenticar na aplicação. 2) Clique em FATURAS.
Resultado esperado	Deverá exibir todos os pagamentos agrupados por cliente e competência, juntamente com o número de atendimentos efetuadas referente o pagamento em questão, o valor total e seus status
Data	22/05/2018
Analista	Vitor
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 33 – Matriz de teste 31

ID	31
Módulo	Faturas
Descrição	Efetuar pagamento
Roteiro	1) Se autenticar na aplicação. 2) Clique em FATURAS. 3) Clique em QUITAR em uma fatura específico.
Resultado esperado	Ao efetuar um pagamento o status deverá ser alterado de "Em aberto" para "Pago"
Data	22/05/2018
Analista	Vitor
Resultado do teste	Executado com sucesso

Correções / Melhorias	

Fonte: Autor

Quadro 34 – Matriz de teste 32

ID	32
Módulo	Faturas
Descrição	Efetuar pagamento
Roteiro	1) Se autenticar na aplicação. 2) Clique em FATURAS. 3) Clique em QUITAR em uma fatura específico. 4) Clique em FATURAS. 5) Verifique se o valor em aberto diminuiu para o cliente que foi quitada a fatura.
Resultado esperado	Ao efetuar um pagamento o valor deve ser subtraído do valor em Faturas em aberto do cliente em questão
Data	22/05/2018
Analista	Vitor
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 35 – Matriz de teste 33

ID	33
Módulo	Faturas
Descrição	Estornar pagamento
Roteiro	1) Se autenticar na aplicação. 2) Clique em FATURAS. 3) Clique em ESTORNAR em um pagamento específico.
Resultado esperado	Ao efetuar um estorno de pagamento o status deverá ser alterado de "pago" para "Em aberto"
Data	22/05/2018
Analista	Vitor

Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 36 – Matriz de teste 34

ID	34
Módulo	Faturas
Descrição	Efetuar pagamento
Roteiro	1) Se autenticar na aplicação. 2) Clique em FATURAS. 3) Clique em ESTORNAR em um pagamento específico. 4) Clique em FATURAS. 5) Verifique se o valor em aberto aumentou para o cliente que foi quitado o pagamento.
Resultado esperado	Ao efetuar um estorno de pagamento o valor deve ser adicionado do valor em Faturas em aberto do cliente em questão
Data	22/05/2018
Analista	Vitor
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 37 – Matriz de teste 35

ID	35
Módulo	Relatório Financeiro
Descrição	Exibir relatório financeiro
Roteiro	1) Se autenticar na aplicação. 2) Clique em RELATÓRIOS em seguida em FINANCEIRO POR COMPETÊNCIA.
Resultado esperado	O relatório deverá exibir o valor previsto e o valor realizado (valor que o profissional já recebeu), agrupados por competência. O valor previsto deve ser a soma de todas as consultas marcadas no mês e o valor realizado deverá ser o total recebido refente ao mês de atendimento.
Data	22/05/2018

Analista	Vitor
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

Quadro 38 – Matriz de teste 36

ID	36
Módulo	Logout
Descrição	Fazer o logout
Roteiro	1) Se autenticar na aplicação. 2) Clique em SAIR.
Resultado esperado	O usuário deverá ser desconectado e redirecionado para tela de login
Data	22/05/2018
Analista	Vitor
Resultado do teste	Executado com sucesso
Correções / Melhorias	

Fonte: Autor

3.3.3.1 Testes unitários

Os testes unitários tem como objetivo testar cada funcionalidade do sistema isoladamente, facilitando assim a identificação e correção de erros. São benefícios dos testes unitários: ajuda a determinar especificações, fácil entendimento do código, maior facilidade no desenvolvimento de documentações (GULATI; SHARMA, 2017).

Utilizamos como ferramentas de testes unitários no backend o Rspec (https://github.com/rspec/rspec-rails) e Cypress no frontend (https://cypress.io).

Disponibilizamos um exemplo de teste feito com rspec no apêndice A.

Figura 22 – Logotipo Rspec

Fonte: http://rspec.info/images/logo.png

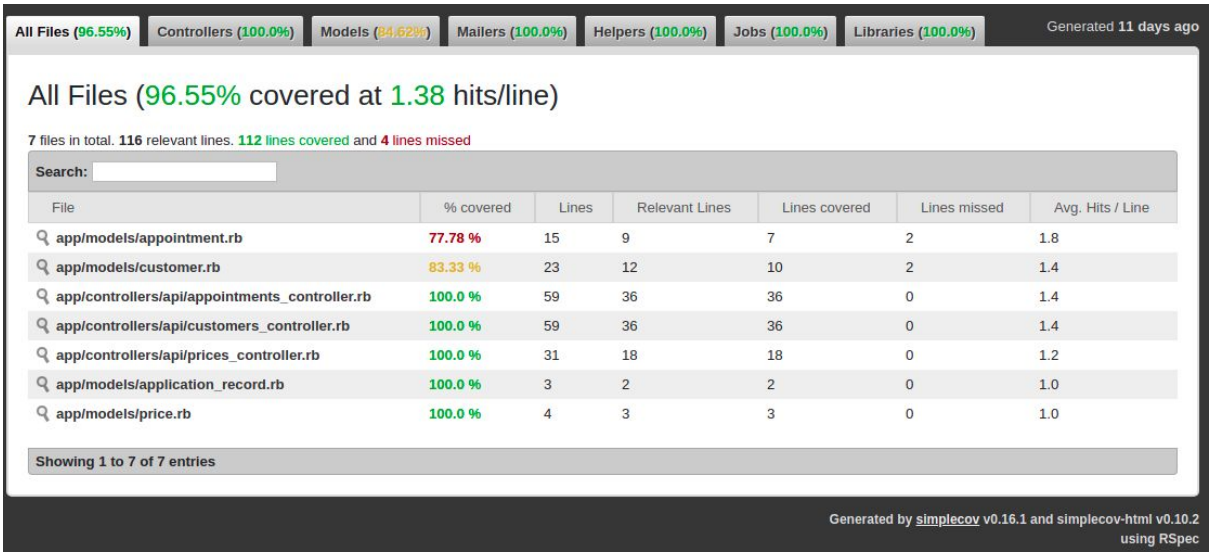
Figura 23 – Logotipo Cypress

Fonte: https://www.cypress.io/img/logo-dark.36f3e062.png

3.3.3.2 Cobertura de testes - api

Para monitorar a cobertura de testes da api foi utilizado uma Gem chamada Simplecov, que mostra quais as partes do código estão cobertas por testes.

Figura 24 – Print screen dashboard simplecov



Fonte: Autor

Já a tela a seguir mostra um exemplo de detalhado de uma *model* da nossa aplicação. Veja que a ferramenta destaca, em vermelho , os pontos que ainda não foram testados

Figura 25 – Print screen detalhe simplecov

app/models/customer.rb
83.33 % covered
12 relevant lines. 10 lines covered and 2 lines missed.

```
1.  class Customer < ApplicationRecord
2.    has_many :prices
3.    has_many :appointments
4.    validates :name, :email, :phone, presence: true
5.
6.    def status_customer
7.    status ? 'Ativo' : 'Inativo'
8.    end
9.
10.   def prices
11.     Price.where(customer_id: self).order(readjust: :desc)
12.   end
13.
14.   def current_price
15.     self.prices.order(readjust: :desc).
16.                 where("readjust < ?", Time.now).
17.                 limit(1)
18.   end
19.
20.   def active_customers
21.     Customer.where(status: true)
22.   end
23. end
```

Fonte: Autor

3.4 BANCO DE DADOS

Um item muito sensível em grande parte dos projetos de software são os bancos de dados, pois os dados armazenados tem um valor significante ao negócio.

> Os dados de uma empresa, se não forem o elemento mais precioso, estão entre eles. Uma informação armazenada incorretamente, ou de forma desordenada, pode custar todo o negócio. (CARVALHO, 2017, p. 3)

Neste projeto os dados são tratado na camada de API e o banco de dados utilizado é o PostgreSQL.

Figura 26 – Logotipo PostreSQL

Fonte: http://blog.dbaacademy.com.br/postgresql-configuracao-zabbix-em-centos/

O PostgreSQL é um sistema de gerenciamento de banco de dados (SGBD) relacional de código aberto baseado em SQL e alguns dos motivos que escolhemos esse SGBD foram:

- Linguagem SQL;
- É gratuito;
- Fácil integração com Rails;
- Fácil integração com Heroku.

3.5 SEGURANÇA

3.5.1 Controle de acesso

O controle de acesso é feito por um sistema de autenticação que solicita um login, no nosso caso o email e a senha do usuário. Essa autenticação cumpre muito bem o pilar de confidencialidade garantindo assim que as informações só serão acessadas por um usuário autorizado. Cumprindo de o papel de “assegurar acesso de usuário autorizado e prevenir

acesso não autorizado a sistemas de informação" como cita a Norma Brasileira ABNT NBR ISO/IEC 27002 (2005, p. 66 - primeira edição)

3.5.2 Prevenção de SQL injection

SQL injection é um ataque baseado em digitar comandos SQL em *inputs* de formulários da aplicação. Caso a aplicação não previna este tipo de ataque, através de validações, o estrago causado por um ataque desses pode ser catastrófico. Um exemplo disso é se o ataque possuir um comando "delete from customers;", esse ataque se executado de maneira correta pode apagar nossa tabela de clientes, trazendo um prejuízo enorme, caso não possua *backup*.

Para prevenir SQL injection utilizamos uma técnica que o Rails chamada de *Strong Parameters*. Esta técnica protege os atributos evitando assim que instruções SQL sejam injetadas nos formulários.

3.5 .3 Criptografia de senha

Outro dado muito sensível é a senha de usuário. Para prevenir acesso a senha de nossos usuários, em caso de acesso a tabela de *users*, foi utilizado uma gem chamada bcrypt para criptografar as senhas de cada usuário.

3.5.4 Requisição segura

Como citado anteriormente, nosso projeto possui uma camada de API e outra aplicação onde ocorre a interação com cliente. Com isso tivemos que pensar em uma maneira de evitar requisições a API de aplicações alheias a nossa, ou seja, para que uma aplicação acesse dados da API, há a necessidade de saber se a requisição vem de uma aplicação devidamente autorizada a acessar esses dados. Para efetuar essa verificação nos utilizamos do protocolo OAuth2.

Este protocolo trabalha fornecendo tokens de autorização que identificam o usuário e suas permissões em uma aplicação a partir da aplicação de terceiro. Para isso o usuário deve

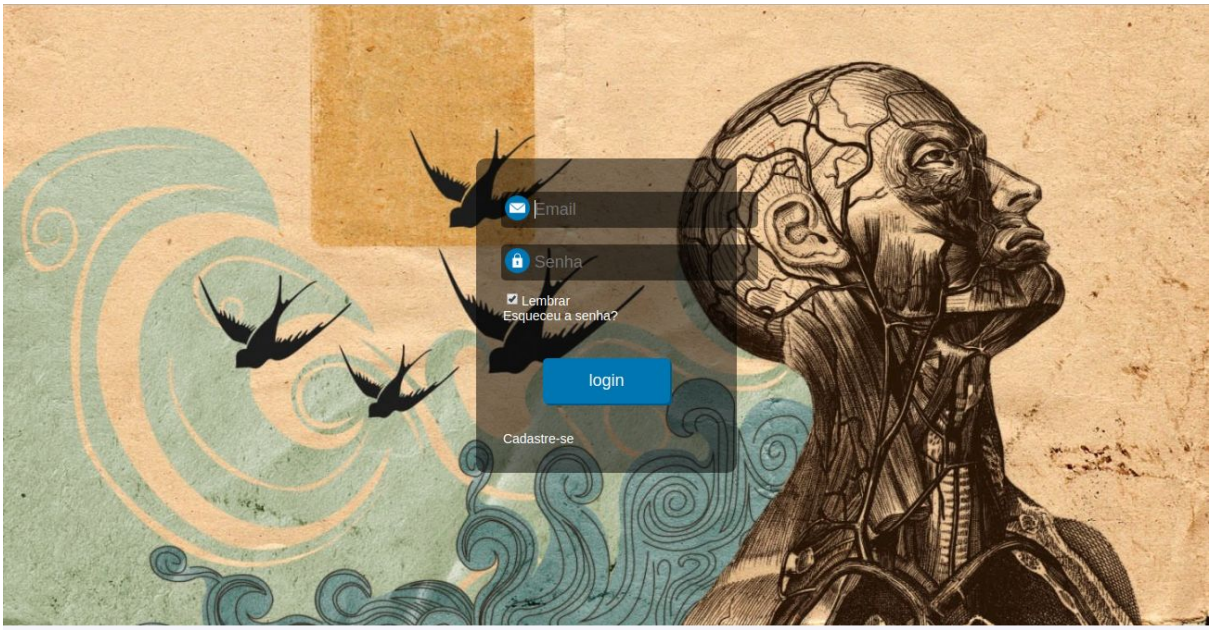
ser cadastrado em ambas aplicações para que o protocolo saiba que o usuário “Cliente AS” da aplicação x é o mesmo usuário “Cliente A da Silva” da aplicação y. Em nosso caso essa validação não é feita por usuário e sim por aplicação, ou seja, o usuário irá se autenticar normalmente na aplicação de interface com usuário e essa aplicação é que enviará o seu token para a API, informando dessa forma que ela é uma aplicação autorizada a acessar as informações da API. Neste projeto não será necessário que o token do cliente seja enviado, apenas o token da aplicação. Esse token é enviado no *header* de cada requisição. Sendo assim cada vez que o usuário logar na aplicação a api irá devolver esse token na session do usuário. Para implementar este protocolo utilizamos a gem *oauth2*.

Porém se a responsabilidade de gerar o token é da API, teríamos que pensar como iríamos garantir que a requisição feita pelo login também era válida. Para solucionar esse problema utilizamos o Json Web Token (JWT) para criptografar os dados da requisição, assim a requisição só será válida se a API conseguir decodificar a mesma utilizando a chave de acesso conhecida pelas aplicações do projeto.

3.6 DESCRIÇÃO DAS INTERFACES

3.6.1 Login

Figura 27 – Tela de login

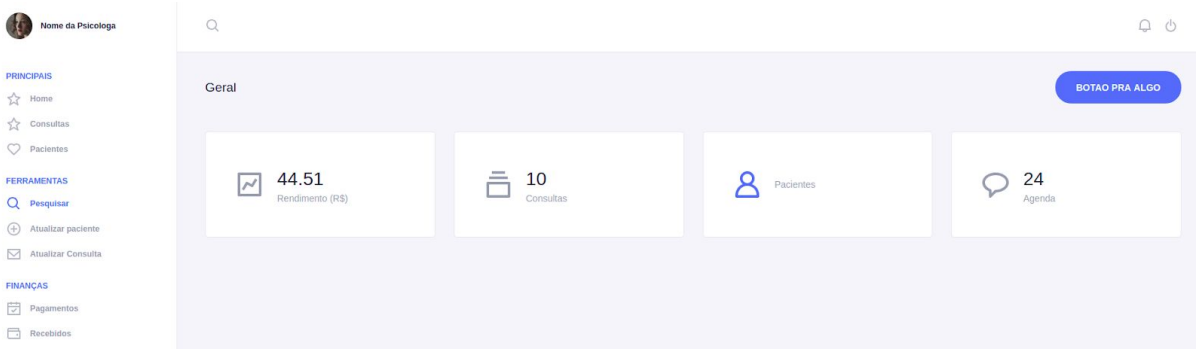


Fonte: Autor

3.6.2 Dashboard

Após autenticação, o profissional será redirecionado para tela principal, onde terá acesso a um menu lateral e atalhos com as principais funcionalidades e indicadores.

Figura 28 – Tela de dashboard

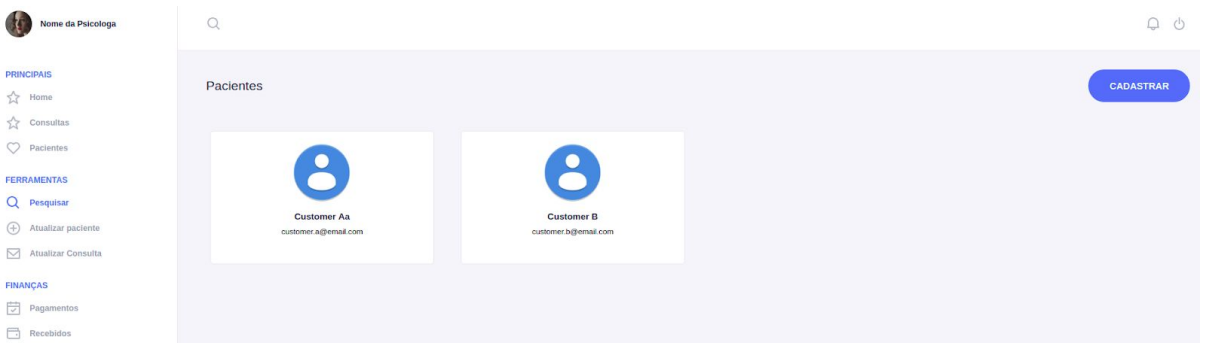


Fonte: Autor

3.6.3 Cliente

Ao clicar no menu lateral o Clientes será apresentada a lista de clientes em ordem alfabética.

Figura 29 – Tela de clientes

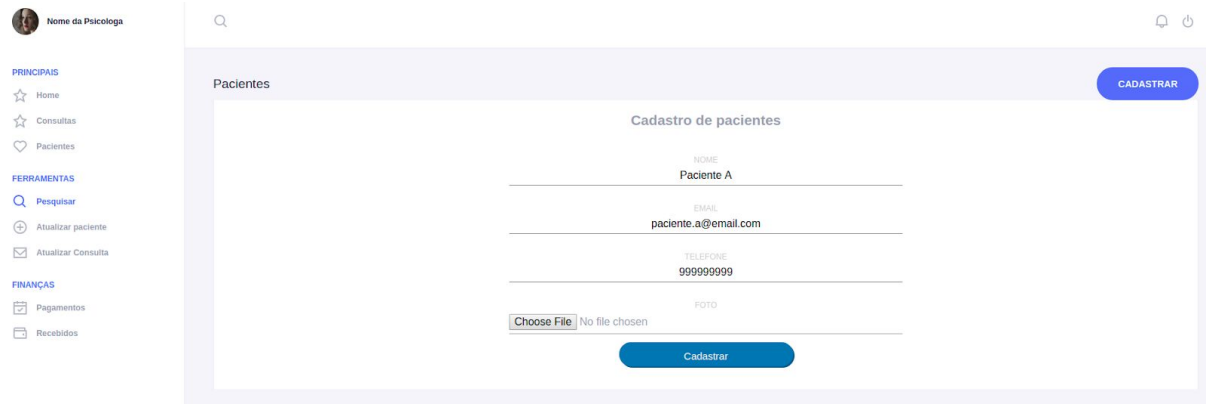


Fonte: Autor

3.6.4 Cadastro de cliente

Clicando no botão Cadastrar, que encontra-se no topo do lado direito na lista de clientes, ficará disponível o formulário para cadastro de cliente.

Figura 30 – Tela de cadastro de cliente

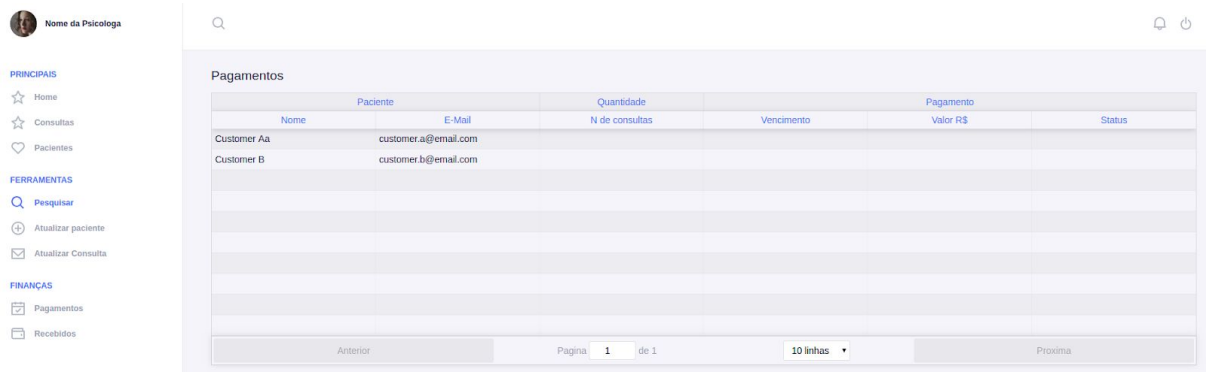


Fonte: Autor

3.6.5 Faturas

Para ver as faturas e seu status o usuário deverá clicar em Faturas no menu lateral e serão listados as faturas em aberto e seus pagamentos.

Figura 31 – Tela de pagamentos

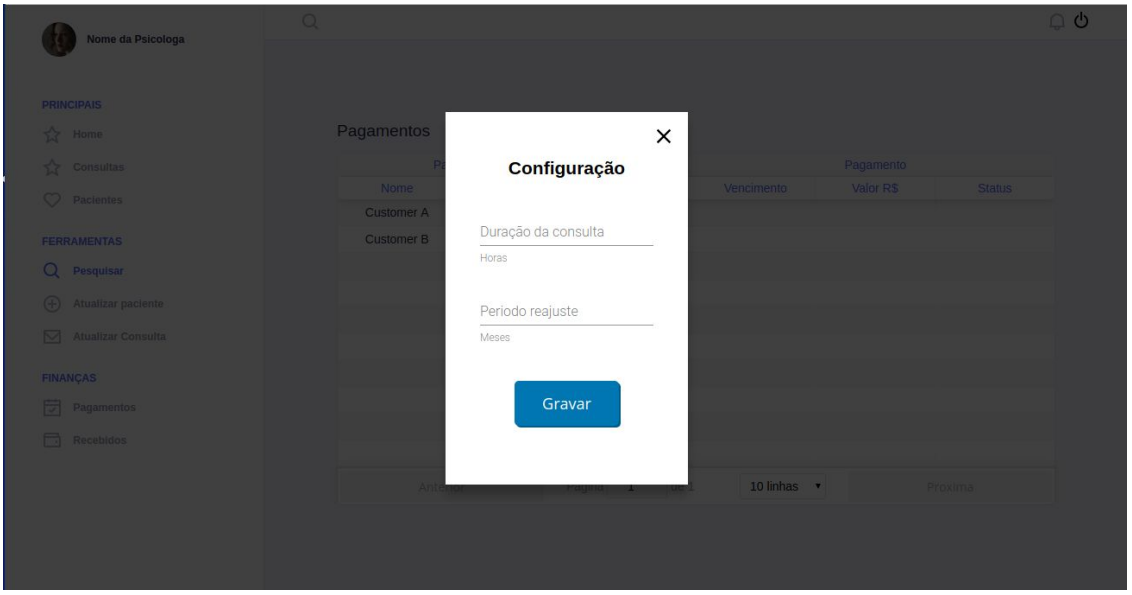


Fonte: Autor

3.6.6 Configurações

As configurações de quanto tempo deverá durar um atendimento e qual será o intervalo de reajuste poderão ser configurados no menu de configurações que poderá ser acessado no ícone do sino que fica no topo direito ao lado do botão de logoff.

Figura 32 – Tela de configurações



Fonte: Autor

3.7 CRITÉRIOS DE USABILIDADE

Em primeiro momento a aplicação foi desenvolvida para web de forma responsiva, porém a utilização de micro serviço facilitará muito o desenvolvimento da aplicação mobile, já que o banco de dados é fornecido por uma aplicação a parte via API.

3.8 INFRAESTRUTURA E CONFIGURAÇÕES

Para tornar esse projeto ser acessível aos usuários contratamos um serviço de servidor cloud dedicado, ou seja, um infraestrutura dedicada a nossa aplicação, sem compartilhamento com outros usuários ou serviços. Esse serviço é conhecido como infrastructure as a Service (IaaS), ou em português infraestrutura como um serviço.

As configurações desse servidor são:

Memória: 6 GB

Processamento: 4 vCPUs

Espaço em disco: 70 GB

Transferências:6 TB

Sistema Operacional: Debian 9

4 CONSIDERAÇÕES FINAIS, RECOMENDAÇÕES E LIMITAÇÕES

4.1 RECOMENDAÇÕES

A manutenção deverá ocorrer de forma periodicamente para atualização de dependências, para evitar vulnerabilidade, além da correção de eventuais bugs.

4.1.1 Melhorias

O projeto EMA possui grandes oportunidades de melhorias, como:

- Emissão de recibos;
- Módulo de despesas, para que o profissional possa controlar seu fluxo de caixa;
- Adição do segundo fator de autenticação para melhorar a segurança do usuário*;*
- Acesso com cliente, para que o cliente seja capaz de consultar e agendar e atendimentos, além de poder consultar suas faturas.

4.2 CONSIDERAÇÕES FINAIS

A premissa do projeto era criar uma aplicação que pudesse auxiliar na organização das rotinas administrativas de um profissional autônomo.

Ao longo do processo de desenvolvimento encontramos algumas mudanças que aumentaram a complexidade do projeto, como por exemplo o fato do recibo de pagamento poder ser emitido em nome de terceiros. Ex.: O cliente é uma criança e o responsável pelo pagamento um mês é o pai e outro a mãe, nesse caso o recibo não é emitido no nome do cliente, além disso o pagador pode variar de um mês para o outro. Por essa restrição não ter sido identificada no início do projeto preferimos deixá-la para um segundo momento, evitando assim comprometer a entrega do projeto.

Por outro lado, desenhamos a aplicação de forma a facilitar novas implementações, como uma futura aplicação mobile, pois utilizamos técnicas de microsserviços, que torna o *back-end (*regra de negócio) independente do *front-end*. Dessa forma podemos migrar a

linguagem de programação ou criar aplicações para diversas plataformas sem precisar reescrever a regra de negócio.

Apesar do recibo não ter sido implementado nesta fase do projeto o resultado foi satisfatório, pois cobriu os demais requisitos levantados em seu planejamento, tornando-se uma aplicação capaz de auxiliar um profissional autônomo em suas rotinas básicas de acordo com o propósito do trabalho.

Fica agora a oportunidade de transformá-lo em em um sistema robusto que contemple um número maior de atividades, como algumas melhorias citadas no item 4.1.1 , para auxiliar ainda mais seus usuários.

REFERÊNCIAS

AGÊNCIA DE NOTÍCIAS IBGE, Desemprego volta a crescer no primeiro trimestre de 2018. Disponivel em : <https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/20995-desemprego-volta-a-crescer-no-primeiro-trimestre-de-2018>. Acessado em 13 de out. 2018.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS – ABNT. NBR 10719: informação e documentação – Relatório Técnico e/ou científico – Apresentação. Rio de Janeiro, 2011.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS – ABNT. NBR ISO/IEC 27002: Tecnologia da informação - Técnicas de segurança - Código de prática para gestão da segurança da informação. Rio de Janeiro, 2005.

CommonsJS. Project history. Disponivel em <http://www.commonjs.org/history/>. Acessado em 30 de set. 2018.

CARVALHO, Vinicius. PostgreSQL: Banco de dados para aplicações web modernas. São Paulo: Casa do Código, 2017.

ECMA international. Standard ECMA-262. Disponivel em <http://www.ecma-international.org/publications/standards/Ecma-262.htm>. Acessado em 10 de out. 2018.

FOWLER, Martin. Microservice Trade-Offs. Disponível em < https://martinfowler.com/articles/microservice-trade-offs.html />. Acesso em: 04 de out. 2018.

GAMBARRA, Thiago. Alta escalabilidade com microserviços. Disponível em <https://www.mundipagg.com/blog/alta-escalabilidade-com-microservicos/>. Acesso em: 04 de out. 2018.

GULATI, S.; SHARMA, R. Java Unit Testing with JUnit 5. [S.l.]: Ed Apress, 2017.

LEWIS, james; FOWLER, Martin. Microservices. Disponível em < https://martinfowler.com/articles/microservices.html/>. Acesso em: 04 de out. 2018.

MALIPENSE, L. M.; ZUCHI, J. D. Um estudo sobre o conceito e a aplicação da arquitetura de microserviços. Revista Interface Tecnológica, 30 jun. 2018.

MDN web docs. Javascript Disponivel em <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. Acessado em 13 de out. 2018.

MONTEIRO, Leandro Pinheiro. O que é linguagem de programação? Disponível em: <https://universidadedatecnologia.com.br/o-que-e-linguagem-de-programacao/>. Acesso em: 29 de out. 2018.

React. React. Disponivel em <https://reactjs.org/>. Acessado em 03 de out. 2018.

Ruby Langue. Sobre o Ruby .Disponivel em <https://www.ruby-lang.org/pt/about/>. Acessado em 15 de out. 2018.

SANTOS, Antonio Henrique dos; CARVALHO, Nelson Ribeiro. Frameworks e seus benefícios no desenvolvimento de software. Disponivel em<http://revistapensar.com.br/tecnologia/pasta_upload/artigos/a95.pdf>. Acessado em 05 de out. 2018.

APÊNDICE A – Exemplo de teste automatizado em rspec rails

```
require 'rails_helper'
require 'net/http'
describe 'Customers API', type: :request do
  it 'create customer' do
    data = {}
    data['name'] = 'Custumer A'
    data['email'] = 'customer.a@email.com'
    data['phone'] = '123456789'

    valid_params = {}
    valid_params = { name: 'Customer A', email:'customer.a@email.com',
                     phone: '123456789' }

    post api_customers_path(params: valid_params)
    data = JSON.parse(response.body)
    expect(response.status).to eq 201
    expect(data['message']).to eq 'Cliente cadastrado com sucesso.'
  end

  it 'create customer' do
    data = {}

    valid_params = {}
    valid_params = { name: '', email:'', phone: '' }

    post api_customers_path(params: valid_params)
    data = JSON.parse(response.body)
    expect(data['message']).to eq 'Cliente não pode ser cadastrado.'
  end
end
```