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

JORGE MITSUMASSA NAKAGAWA

SISTEMA GERENCIADOR DE AGENDAMENTO DE SERVIÇOS: Um estudo de caso em salão de beleza

TRABALHO DE DIPLOMAÇÃO

CORNÉLIO PROCÓPIO
2014

JORGE MITSUMASSA NAKAGAWA

SISTEMA GERENCIADOR DE AGENDAMENTO DE SERVIÇOS: Um estudo de caso em salão de beleza

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

Orientador: Prof. Dr. Fabrício Martins Lopes

CORNÉLIO PROCÓPIO

2014

TERMO DE APROVAÇÃO

SISTEMA GERENCIADOR DE AGENDAMENTO DE SERVIÇOS: UM ESTUDO DE CASO DE SALÃO DE BELEZA

JORGE MITSUMASSA NAKAGAWA

_____
Prof. Fabrício Martins Lopes
Orientador

_____
Membro de banca

_____
Membro de banca

DEDICATÓRIA

Dedico este trabalho à minha família por sempre acreditar em mim, aos amigos presentes nessa longa caminhada e ao orientador Fabrício Martins Lopes pelo apoio e incentivo que tornaram possível a conclusão deste trabalho.

AGRADECIMENTOS

A Deus por ter me dado saúde e coragem para superar as dificuldades.

A esta universidade, seu corpo docente, direção e administração que fizeram possível a realização de um sonho.

Ao meu orientador Fabrício Martins Lopes, pelo suporte no pouco tempo que lhe coube, pelas suas correções, incentivos e sabedoria com que me guiou neste caminho.

Aos meus pais, pelo amor, incentivo e apoio incondicional.

E a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

RESUMO

NAKAGAWA, Jorge. **Sistema gerenciador de agendamento de serviços: Um estudo de caso de salão de beleza**. 2014**.71**. Trabalho de Conclusão de Curso em Tecnologia em Análise e Desenvolvimento de Sistemas-Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2014.

O crescimento da área de beleza e o aumento expressivo do número de salões existentes no mercado fez com que oferecer um serviço diferenciado fosse necessário para atrair e manter clientes como uma vantagem competitiva em relação a outros salões. Com a agitação do dia a dia, o tempo livre está cada vez mais escasso, fazendo com que as pessoas procurem a praticidade e comodidade para realizar o agendamento. Este trabalho apresenta o desenvolvimento de um sistema gerenciador de agendamento de serviços. O objetivo é oferecer uma solução web e mobile para o agendamento de serviços em um salão de beleza, aplicando a metodologia ágil para focar na real necessidade do cliente e a utilização dos diagramas da *Unifield Modeling Language* (UML) para o melhor entendimento do requisito. A codificação foi realizada na linguagem *Ruby,* conjuntamente com o *framework Rails e Android* para o desenvolvimento do aplicativo mobile. Para o armazenamento de dados foi utilizado o *Active Record* como mapeardor de objeto relacional sobre o Sistema gerenciador de banco de dados *PostgreSQL*.

Palavras-chave: Agendamento de serviços, Ruby On rails, Android, Web, Mobile.

ABSTRACT

NAKAGAWA, Jorge. **System manager scheduling system: a case study of Salon.** 2014.**71**. Graduation work in Tecnology in Analisys and Development of System – Federal Technology University of Paraná. Cornélio Procópio, 2014.

The growth of the beauty area, and the significant increase in the number of existing salons on the market, made to offer a differentiated service was necessary to attract and retain customers as a competitive advantage compared to other salons. With the hustle of daily free time is increasingly scarce, causing people to seek the practicality and convenience to make an appointment. This work presents the development of system manager scheduling service. The objective is to provide a mobile and web solution for scheduling services in a salon applying agile methodology to focus on real customer need and use case of diagrams *Unifield Modeling Language* (UML) for better understanding of requirement. Coding was carried out in the *Ruby* language with the *Rails* framework and *Android* for mobile application development. The datastore *Active Record* is used as object relational mapping on the system manager *PostgreSQL* database.

Keywords: Scheduling service, Ruby On rails, Android, Web, Mobile.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

GHz.	Giga Hertz

LISTA DE SIGLAS

GB	Gigabytes
XP	Extreme Programming
SQL	Structured Query Language

LISTA DE ACRÔNIMOS

RAM	Random Access Memory

SUMÁRIO

1. INTRODUÇÃO

O gerenciamento das informações, agilidade nos processos e qualidade no atendimento, torna-se cada vez mais essenciais para uma empresa obter o sucesso em um mercado cada vez mais competitivo. Segundo Mullins (2004, p.33), o sucesso da organização só será obtido caso a organização busque a necessidade do cliente, mantendo sempre o foco no cliente, com suas metas organizacionais e o comprometimento. De acordo com Silva e Barbosa (2002), sistemas que aumentam a velocidade de execução e reduzem os erros de determinada atividade são formas de agilizar os processos. De acordo com Dalfovo e Amorim (2000. P.17) a necessidade de administrar as informações geradas é essencial para o sucesso das empresas, pois existe uma crescente demanda e sofisticação na utilização das tecnologias da informação aplicadas ao software e hardware para realizar o registro consistente da informação, contribuir na melhoria o fluxo interno da organização e automatizar os processos manuais. Com estes recursos disponibilizados pela tecnologia da informação, o gestor da organização tem acesso a uma informação consistente e de qualidade para auxiliar na tomada de decisão. Assim pode se concluir que o gerenciamento da informação de maneira eficiente é uma condição necessária para o sucesso das empresas.

A necessidade da interação entre o ser humano e o computador é uma realidade nos dias de hoje, mas por muito tempo este processo não passava do mouse, teclado e monitor, no caso dos computadores pessoais. Os recursos e tecnologias evoluíram, e atualmente com o simples toque a tela, gesto e comando de voz é possível interagir com um dispositivo móvel.

No setor de beleza, principalmente nos salões de beleza é notável a falta de sistemas para auxiliar na organização dos horários de agendamentos com o consumidor. Geralmente os controles são realizados através da agenda e blocos de anotações. O controle manual dos agendamentos é facilmente extraviado e pode gerar perdas de informações e aumenta a dificuldade em efetuar buscas e pesquisas. Assim surge a necessidade de um funcionário específico para gerenciar os horários e entrar em contato com os clientes para confirmar os horários agendados manualmente.

A evolução tecnológica permitiu que os trabalhos feitos manualmente fossem informatizados e assim otimizados, pois o agendamento registrado em papel não é eficiente para atingir todas as necessidades para obter o sucesso no mercado.

O agendamento eletrônico oferece suporte das informações aos funcionários do salão, auxilia na preparação das máquinas e ferramentas que serão utilizadas na prestação de do serviço, movimenta funcionários de acordo com a demanda, possibilita realizar a projeção da quantidade de atendimentos, auxiliando na tomada de decisão do empresário.

Considerando a importância do agendamento em salão de beleza, os processos realizados manualmente para alimentar os dados do agendamento podem ser informatizados através de sistemas computacionais cliente/servidor.

Segundo (ZACKER; DOYLE, 2000), tais sistemas podem ser descritos como “um sistema computacional no qual o processamento efetivo necessário para efetuar uma determinada tarefa é dividido entre um computador principal, o servidor, e uma estação de trabalho individual, o cliente”, e podem ser representados conforme exibido na Figura 1.

O trabalho de diplomação é apresentado da seguinte forma:

- No capítulo 3 constam os conceitos gerais para o entendimento do trabalho desenvolvido;
- No capítulo 4 é descrita a metodologia utilizada para o projeto;
- No capítulo 5 são listadas as tecnologias aplicadas para o desenvolvimento do projeto;
- No capítulo 6 é apresentada a abordagem do processo de desenvolvimento ágil aplicado ao projeto do agendamento de serviço;
- No capítulo 7 é apresentado o resultado do projeto desenvolvido;
- No capítulo 8 está o cronograma de execução do desenvolvimento juntamente com os recursos utilizados no desenvolvimento;
- No capítulo 9 são relatadas as considerações finais do projeto e trabalhos futuros em relação a este projeto.

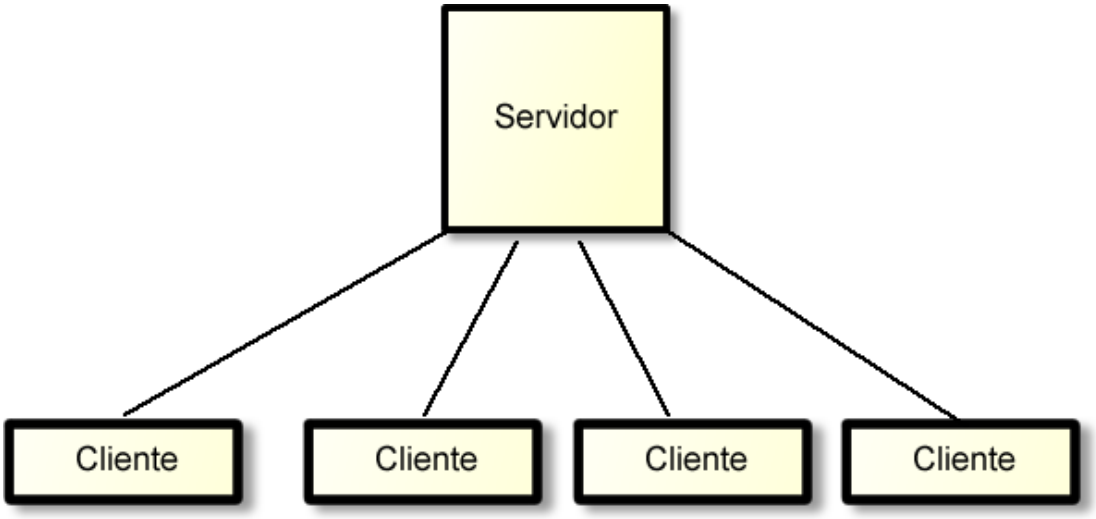


Figura 1 - Representação da arquitetura cliente/servidor.

1.1. OBJETIVO GERAL

O objetivo geral deste trabalho de diplomação é desenvolver uma aplicação web e móvel para o agendamento de serviços de um salão de beleza utilizando a tecnologia *Android* e *Ruby On Rails* aplicando o conceito do desenvolvimento cliente/servidor e método ágil.

1.2. OBJETIVO ESPECÍFICO

Os objetivos específicos deste trabalho de diplomação é desenvolver um sistema capaz de gerenciar os agendamentos de um salão de beleza, considerando os seguintes aspectos:

- Permitir o gerenciamento do perfil do cliente - Informações como nome completo, data de nascimento, endereço, email, telefone;
- Gerenciamento do perfil profissional - Informações como nome completo, especialidade do profissional e telefone;
- Gerenciamento da atividade do profissional - Informações das atividades executadas pelo profissional;

- Gerenciamento do horário do profissional - Informações dos horários disponíveis do profissional para o agendamento;
- Gerenciamento de agendamento online - Disponibilização dos horários dos profissionais no salão com seus serviços disponibilizados, possibilidade de realizar o agendamento pelo sistema e envio de e-mail com o aviso de agendamento;
- Oferecer recursos necessários para diminuir a demanda do atendente;
- Facilitar o agendamento do serviço, oferecendo a comodidade aos clientes;
- Oferecer recursos necessários para eliminar a forma manuscrita de anotação.

2. JUSTIFICATIVA

Segundo um levantamento da Associação Nacional do Comércio de artigos de higiene pessoal e beleza (Anabel, 2013) o número de salões no país cresceu 78% em cinco anos, superando até lanchonetes e bares em quantidade de registros conforme pode ser visto na Figura 2. De acordo com dados do Serviço Brasileiro de apoio às Micro e Pequenas Empresas (SEBRAE, 2013), o número de empreendedores individuais formalizados já chegou a 1,5 milhões no país. Os cabeleireiros ocupam o segundo lugar na lista das principais ocupações, com 7,6% no total de registros, ficando atrás apenas de comércio de vestuários e acessórios, conforme pode ser visto na Figura 3.

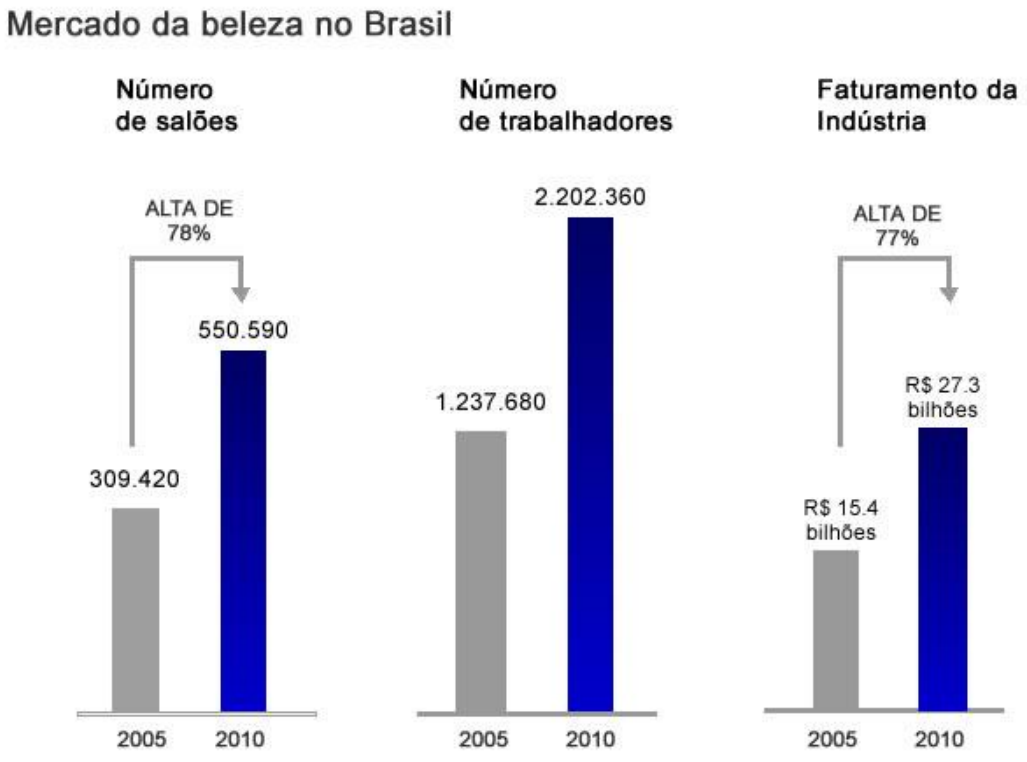


Figura 2 - Mercado da beleza do Brasil. Fonte: Associação Nacional do Comércio e Artigos de Higiene Pessoa e Beleza (ANABEL, 2013).

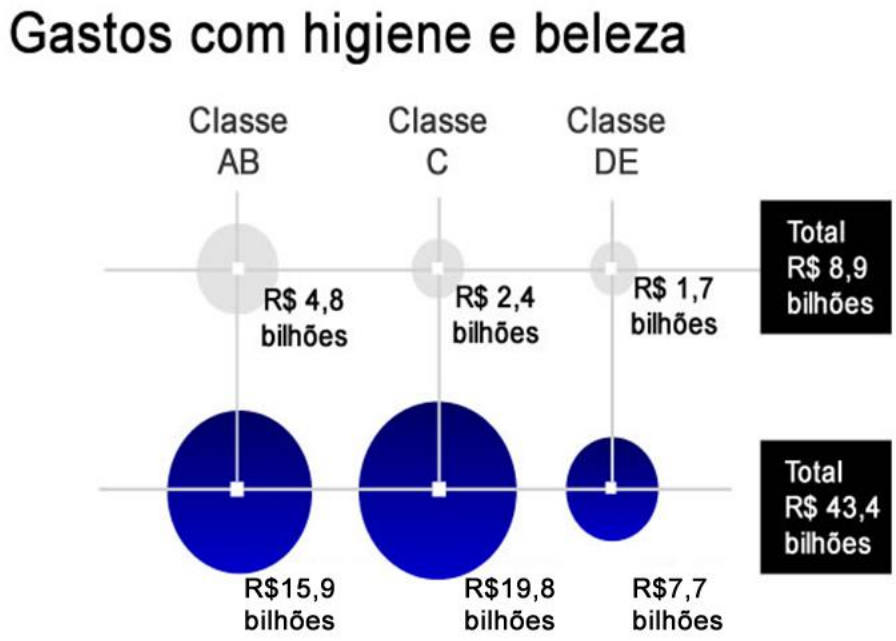


Figura 3 - Mercado da beleza do Brasil. Fonte: Associação Nacional do Comércio e Artigos de Higiene Pessoa e Beleza (ANABEL, 2013).

Analisando os dados levantados, pode-se verificar um crescimento significativo no setor de beleza com o aumento no número de salões, trabalhadores e faturamento da indústria. Com estes dados é possível induzir que a quantidade de usuários também aumentou, fazendo com que a concorrência no mercado do setor de beleza se torne cada vez mais alta e promissora para ser explorada, aumentando assim a necessidade de um sistema capaz de gerenciar o agendamento de maneira eficiente a fim de contribuir para sucesso no mercado, oferecendo facilidade e comodidade ao usuário.

De acordo com Boynton (1993), as organizações estão percebendo de que as melhores estratégias para o sucesso neste ambiente altamente competitivo requerem a melhoria das formas organizacionais, ultrapassando a combinação de estruturas que eram voltadas as estratégias de baixo custo e diferenciação. Um recurso importante frequentemente associado a essa nova organização que se forma é determinado pelo uso tecnologia da informação.

Para Pimentel (2008), o crescimento da utilização de TI é direcionado em todas as extensões, por várias tendências tecnológicas nos sistemas de informação.

O crescimento do desenvolvimento de TI desencadeia o progresso nas práticas de trabalho dos sistemas de informação dentro das organizações.

O agendamento de um serviço tem a finalidade de auxiliar o processo de marcação de um compromisso com entre o cliente e o fornecedor desse serviço, considerando informações como o horário livre, o profissional disponível, servindo de fonte de informação para a tomada de decisão e também com meio de comunicação compartilhado entre todos os profissionais envolvidos. Mas o uso de agendamento em papel tem sido um fator prejudicial para exercer estas finalidades, dentre os quais pode-se relacionar:

- Conteúdo é livre, variando na ordem, algumas vezes é ilegível, incompleto e com informação ambígua, levando a perdas frequentes de informações;
- Só pode estar em um lugar ao mesmo tempo, não permitindo o acesso simultâneo às informações;
- Para análise das informações, o conteúdo precisa ser transcrito, o que muitas vezes predispõe ao erro;
- Dificuldades de efetuar buscas e pesquisas do registro histórico, levantamento do agendamento futuro, prejudicando na tomada de decisão;
- É necessário que o funcionário gerencie os horários disponíveis da agenda;
- Para confirmar o horário é necessário que o funcionário entre em contato com cada cliente manualmente via telefone, e-mail ou mensagem;

De acordo com levantamento realizado pela empresa de desenvolvimento de agendamento online de consultas médicas a YepDoc, o serviço de agendamento de consultas médicas pela internet se mostrou eficaz para a diminuição do índice de não comparecimento de pacientes nas suas consultas marcadas. De acordo com o levantamento, apenas 2% dos agendamentos realizados pela internet não cumpriram o compromisso, já no agendamento manual, por telefone ou no balcão, a taxa de desistência foi de 12% (ZeroHora, 2013).

3. REVISÃO BIBLIOGRÁFICA

3.1. LINGUAGEM DE MODELAGEM UNIFICADA - UML

A UML é uma linguagem visual cuja finalidade é padronizar conceitos de modelagem para que todos os envolvidos no desenvolvimento do projeto tenham a mesma visão das funcionalidades do artefato (CARLSON, 2002).

O objetivo de definir diversos diagramas na UML é oferecer múltiplas visões do sistema a ser modelado (GUEDES, 2008). Cada visão nos permite ter uma projeção voltada a um determinado aspecto do sistema, sendo elas uma complementar a outra. A UML é muito expressiva, pois detalha todas as visões necessárias para o desenvolvimento e implantação do sistema, sendo divida em cinco (BOOCH, 2000):

- Visão de Casos de Uso: focaliza o comportamento do sistema, suas funcionalidades, de modo que todos os envolvidos no projeto tenham a mesma visão de como deve ser determinada funcionalidade.
- Visão de Projeto: focaliza a estrutura do sistema, tendo em vista os elementos necessários para construção do sistema.
- Visão de Processo: focaliza o desempenho e escalabilidade.
- Visão de implementação: focalizam os artefatos físicos como codificação, bibliotecas, banco de dados, entre outros, para a construção do sistema.
- Visão de implantação: focaliza a topologia de hardware, instalação, liberação entre outros artefatos.

Dentre os diagramas definidos pela UML, nas sessões seguintes são apresentados de forma breve os que foram utilizados para a modelagem do trabalho. Para maiores detalhes sobre os conceitos básicos da UML, e seus diagramas, recomenda-se a leitura de (GUEDES, 2008) e (BOOCH et al., 2000).

3.2. DIAGRAMA DE CASO DE USO

Os diagramas de Casos de Uso definem os requisitos funcionais do sistema ou parte dele. São representados por atores, casos de uso e relacionamentos. Seus relacionamentos ocorrem conforme a necessidade de execução das atividades de cada funcionalidade. Assume um importante papel na modelagem comportamental, pois apresenta uma visão externa de como os elementos podem ser utilizados no contexto.

Este diagrama possui como finalidade a visão de casos de uso, responsável por uma descrição do comportamento do sistema de modo abrangente, ou seja, todos os envolvidos no projeto devem ter a mesma visão da funcionalidade do sistema ou parte dele (BOOCH et al., 2000).

3.3. DIAGRAMA DE CLASSE

O diagrama de classe representa a estrutura classes de negócio. Cada classe pode estar relacionada ou não uma a outra por meio de associação (conectadas entre si), dependência (uma classe depende ou usa outra classe), especialização (uma classe é uma especialização de outra classe), ou em pacotes (classes agrupadas por características similares).

Este diagrama é fundamental para uma especificação orientada a objeto. Com esse diagrama é possível descrever de forma mais próxima à estrutura do código do programa. São representados os atributos, métodos e relacionamentos entre as classes.

4. METODOLOGIA

4.1. EXTREME PROGRAMMING

Em (SOMMERVILLE, 2008) é definido o processo de software como um “conjunto de atividades e resultados associados que produzem um produto de software”.

A regra de negócio abordado no projeto é complexa e há a necessidade de validação continua para que o requisito seja conforme a real necessidade do usuário.

Neste sistema, o processo de desenvolvimento adotado é baseado na Extreme Programming (XP) por representar as necessidades do projeto a ser desenvolvido, mudanças rápidas, *feedback* constante do cliente, desenvolvimento iterativo. A Extreme programming (XP) é uma metodologia ágil para equipes pequenas e médias que desenvolvem software baseados em requisitos vagos e que se modificam rapidamente (BECK, 1999). Os seus principais valores são:

Comunicação - A interação do usuário no desenvolvimento de software auxiliar na melhor comunicação com a equipe, fazendo com que a essência do requisito seja mantida de forma que não haja mal-entendido da real necessidade levantada;

Simplicidade - O estudo sobre a utilização das funcionalidades do software foi apresentado pelo presidente Jim Johnson do Standish Group em 2002 na terceira conferência internacional sobre Extreme Programming (JOHNSON, 2002). O resultado do estudo apontou que 45 por cento das funcionalidades encontradas em um sistema não são utilizadas e 19 por cento raramente são usadas conforme pode ser visto na Figura 4. Segundo Vinicius (2005, p.65) “Em outras palavras, os projetos de software frequentemente investem grandes parte dos recursos (tempo, pessoas e dinheiro) em esforços desnecessários.”. Os desenvolvedores de um projeto XP procuram desenvolver as funcionalidades que agregam o valor no software para cada iteração;

Feedback - Uma das dificuldades no processo de desenvolvimento do software é a interpretação correta dos requisitos;

A compreensão das necessidades do usuário é uma das atividades mais difíceis e importantes de serem realizadas. No entanto, a abstração e entendimento dos requisitos é naturalmente difícil, e a tarefa de transmissão é complexa para os usuários em geral. Segundo Vinícius (Brooks, 1987), “nenhuma outra parte do trabalho conceitual é tão difícil quanto estabelecer detalhadamente os requisitos técnicos, incluindo todas as interfaces (...) Nenhuma outra parte é mais difícil de corrigir mais tarde.”;

Coragem - Beck e Fowler (2001) destacam alguns dos fatores que influenciam no psicológico dos integrantes da equipe que afetam diretamente no todo ciclo de desenvolvimento do software.

E os clientes temem:

- Não obter o que pediram;
- Pagar demais por muito pouco;
- Jamais ver um plano relevante;
- Não saber o que está acontecendo
- Fixarem-se em suas primeiras decisões e não serem capazes de reagir a mudanças nos negócios.

Os desenvolvedores temem:

- Ser solicitados a fazer mais do que sabem fazer;
- Ser ordenados a fazer coisas que não façam sentido;
- Ficar defasados tecnicamente;
- Receber responsabilidades, sem autoridade;
- Não receber definições claras sobre o que precisa ser feito;
- Sacrificar a qualidade em função de prazo;

Segundo Vinicius (2005, p.69) “Equipes XP reconhecem estes temores e buscam formas de lidar com eles de maneira corajosa.” afirmando que o desenvolvimento iterativo não elimina o problema, o cliente pode ter feito uma solicitação incorreta, mas este problema é descoberto somente no final da Iteração.

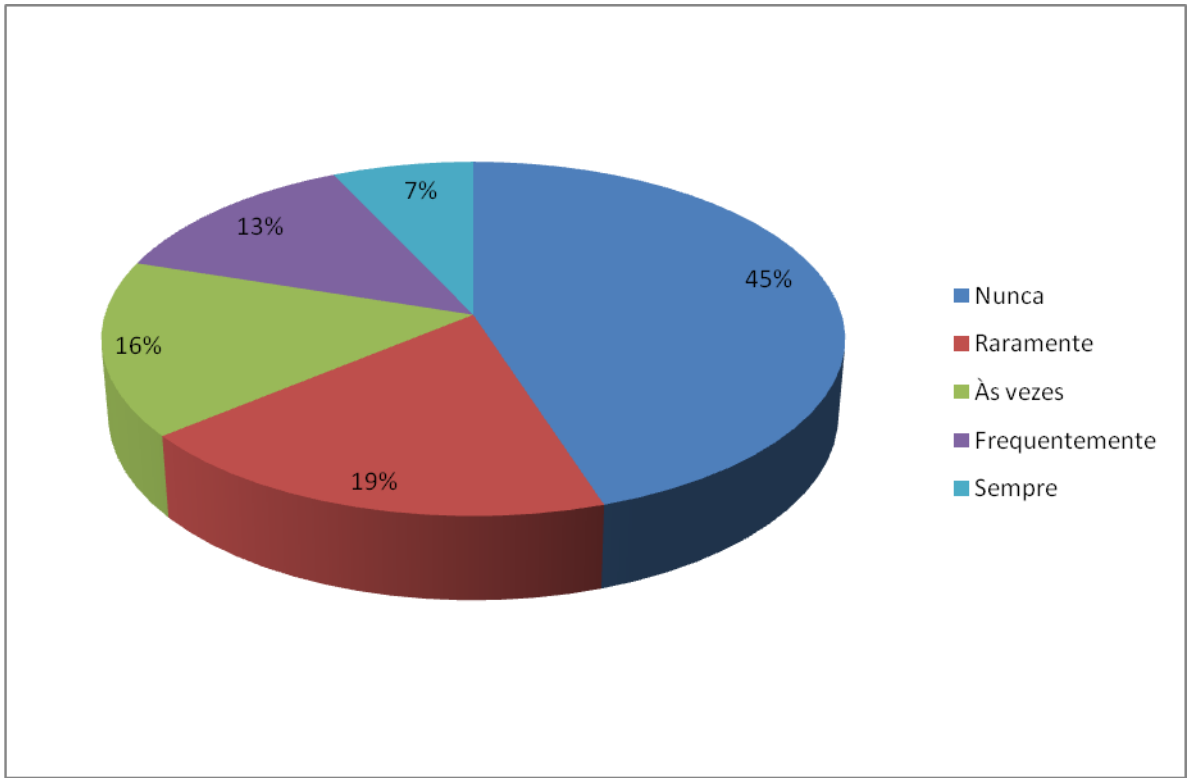


Figura 4 - Estatística sobre a utilização da funcionalidade. Fonte: Vinícius (2005).

5. LINGUAGENS E TECNOLOGIAS

5.1. RUBY

Ruby é uma linguagem orientada a objetos desenvolvida com o propósito de tornar a programação divertida e rápida. Se trata de uma linguagem de fácil utilização, sintaxe intuitiva incorporando funcionalidade completa de orientação objeto e poderosa biblioteca de classes. O criador da linguagem Ruby, o Matsumoto projetou na base da sua experiência para oferecer um conjunto de recursos sólidos que permite focar na solução do problema proposto (Matsumoto, 2001).

Ruby pode ser utilizado para a programação procedural, mas o código é transformado em métodos do objeto global. Ao longo do desenvolvimento da linguagem Ruby, o Matsumoto focou as suas energias para tornar a programação mais rápida e fácil. Para isso foi desenvolvido o que é chamado de "*Principle of least surprise*" este princípio busca diminuir códigos confusos que não expressa o seu significado (Matsumoto, 2001).

As características principais da linguagem Ruby são (Matsumoto, 2001):

- *Interpretive programming*: Sem a necessidade de compilar você pode editar e jogar diretamente para o interpretador. O ciclo de desenvolvimento veloz auxilia no processo de desenvolvimento.
- *Dynamic programming:* Quase tudo em Ruby é feito em runtime. Assim os tipos de variáveis e expressões são determinados em runtime.
- *Family syntax:* Baseado nas sintaxes das linguagens Java, Perl, Python, C/C++ e Smalltalk, a syntaxe da linguagem Ruby é muito fácil de ser aprendidas.
- *Iterators:* Não há a necessidade de controlar o contador na iteração, pois Ruby já possui seus métodos de iteração.
- *Class libraries:* O Ruby vem com uma grande quantidade de classes e bibliotecas que auxilia e que aumenta a velocidade no processo de desenvolvimento.
- *Portable:* O Ruby suporta múltiplas plataformas, incluindo Unix, Dos, Windows, OS/2, etc. A linguagem é executada nestas plataformas sem a necessidade de realizar modificações.

Garbage collection: Programação orientada a objetos realiza a alocação de muitos objetos durante a sua execução, e o coletor recicla objetos que não estão sendo executados.

Desenvolvido tanto para a programação em grande escala quanto para codificação rápida, tem um suporte a orientação a objetos que tem o objetivo de ser simples e prático.

5.2. RAILS

Rails foi apresentado em versão beta em julho de 2004, extraído do aplicativo comercial Basecamp da empresa 37signals, e concluído em apenas dois meses de implementação com quatro mil linhas de código por apenas um único desenvolvedor (AKITA, 2006).

Com a ajuda da comunidade ativa, o Rails teve o seu primeiro lançamento da versão 1.0 após 15 meses de sua apresentação. O fato mais importante é a participação da comunidade na evolução deste framework que teve o seu início com apenas um criador para mais de 100 colaboradores que auxiliaram na conclusão deste projeto. Rails é definitivamente um projeto da comunidade open source (AKITA, 2006).

É necessário que seja entendido o conceito de framework, antes de descrever sobre as características do Rails. Segundo Oliveira (2005), um framework é uma solução para o conjunto de problemas em comum com o uso de classes e interfaces que disponibilizam objetos com funcionalidades comuns a várias aplicações. A utilidade de um framework pode trazer benefícios em relação à agilidade de desenvolvimento podendo reduzir seus custos.

Rails é um framework para desenvolvimento web utilizando a linguagem Ruby, com código aberto e gratuito, lançado ao público pela primeira vez em julho de 2004 por David Heinemeier Hansson. Este framework possui o objetivo de aumentar muito a produtividade sustentável e a diversão do programador, possui uma linguagem simples e elegante a exemplo do Ruby (AKITA, 2006). O framework traz princípios baseados no desenvolvimento ágil como o *Don`t Repeat Yourself (DRY)*, e *Convention over Configuration* (CoC), que é enfatização de que deve-se

realizar a configuração somente em casos específicos, pois ele já segue diversos padrões estabelecidos. Ruby on Rails é chamada de meta-framework, pois ele é formado com a junção de alguns frameworks, entre eles (AKITA, 2006):

- *Action* Support**:** Coleção de várias classes e extensões de bibliotecas padrões, consideradas úteis para o Ruby on Rails;
- *Action Dispatch:* Análise das requisições web para realizar o processamento e encaminhamento da requisição;
- *Action Controller:* Após *action dispatch* ter processado a requisição ele realiza o roteamento do controller. É responsável por controlar a *model*, *view*, gerenciamento de sessões de usuários, fluxo de aplicação, recursos de cache;
- *Active Record:* Camada responsável por representar dados de negócio oferecendo facilidade para a criação de objetos persistentes em um banco de dados servindo como mapeador de objeto-relacional;
- *Action Web Service:* Fornece recursos necessários para que o aplicativo possa responder como um Web Service, o seu aplicativo se torna um provedor de *APIs* a outros aplicativos de forma simples, permitindo criar *API* exposta via *WSDL* com mensagens transportadas via *SOAP*;
- *Action Mailler:* Responsável por serviços de e-mail permite que envie e-mails a partir do aplicativo usando um modelo *Mailler* e *View*;
- *Activion Pack:* Possui o Action View, geração de visualização de usuário, como HTML (Linguagem de Marcação de Hipertexto), *XML* (Linguagem de Marcação), *JavaScript*, entre outros e o *Action Controller*, que controla de fluxo de negócio.

A Figura 5 exibe a arquitetura Ruby On Rails.

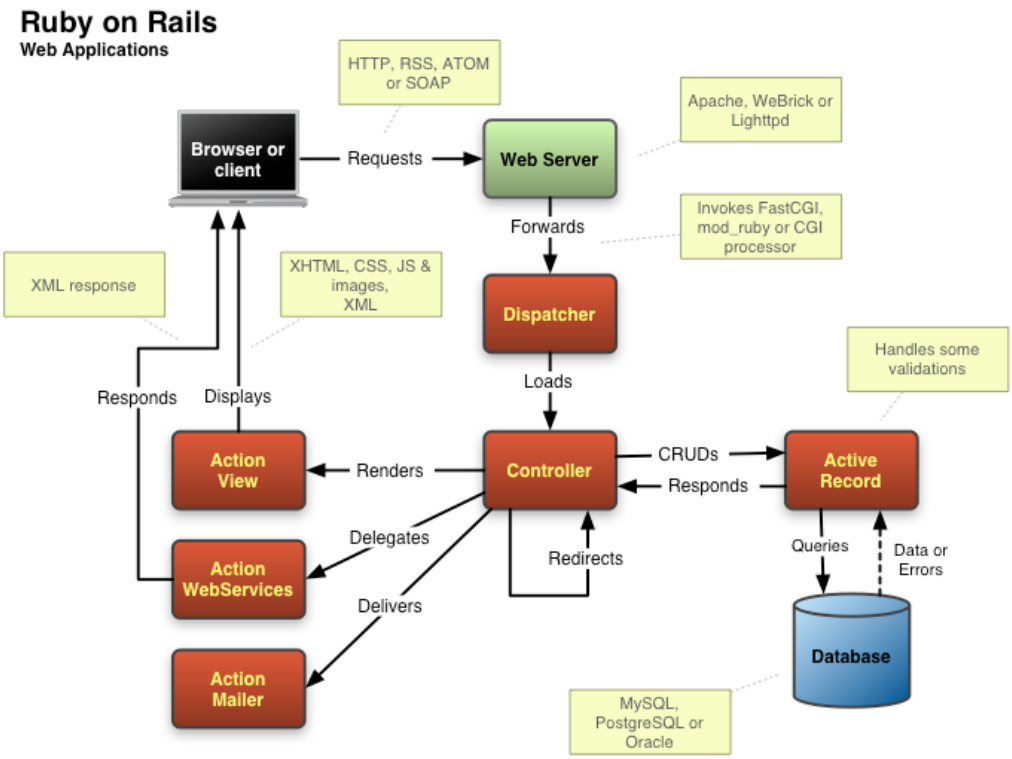


Figura 5 - Arquitetura Rails. Fonte: http:/www.Adrianmeija.com.

5.2.1. SEGURANÇA EM RAILS

Frameworks de aplicações web são feitas para auxiliar os desenvolvedores para construir um sistema. E um dos auxílios é ajuda-los na construção de um sistema seguro. Rails possui métodos inteligentes, por exemplo, contra SQL Injection para dar o suporte aos desenvolvedores para que este não seja o problema nas aplicações Rails.

SQL Injection é um método para influenciar a consulta no banco de dados através da manipulação dos parâmetros de aplicações web. Os ataques mais comuns através do SQL injection é burlar a autorização, realizar a manipulação e leitura dos dados. Seguinte exemplo que não deve ser utilizado na consulta real do sistema (Ruby on rails security, 2014): *Usuario.where (“email = ? and password=?“, email_u, password_u).first*.

A utilização do exemplo da instrução SQL visto a pouco exibem parâmetros que abrem brechas para que sejam injetados os comandos SQL, e que poderão burlar o comportamento original da consulta.

Rails possui um filtro nativo para caractere especial da consulta SQL, no qual caracteres ‘, “ e NULL irão escapar, ou seja, serão tratados como texto, evitando que sejam utilizadas na clausula de maneira maliciosa. Rails recomenda também a utilização do *Hash* no parâmetro em vez de utilizar a String: *Usuario.where (email: email_u, password: password_u).first.*

5.3. ANDROID

A empresa Google adquiriu a Android Inc em Junho de 2005, com sede em Palo Alto, CA, com o objetivo de desenvolver um sistema operacional para dispositivos móveis baseado em Linux tendo como premissa proporcionar uma estrutura flexível e atualizável.

Android é definido por um conjunto de sistema operacional, bibliotecas, frameworks de middleware e aplicações chave, podendo ser visto como uma completa pilha de componentes de software (Android, 2012). Com algumas exceções, como por exemplo, o Linux kernel patches que estão sob a licença GPLv2, a maior parte do Google Android é distribuído sob a licença Apache 2.0, como descreve (Lecheta, 2009):

> A licença do Android é flexível e permite que cada fabricante possa realizar alterações no código-fonte para customizar seus produtos, e o melhor de tudo, sem necessidade de compartilhar essas alterações com ninguém, O Android também é “free” e os fabricantes podem usar e abusar dele sem precisar pagar por isso” (Lecheta 2012, p.22).

A plataforma Android utiliza a linguagem Java e o conjunto de bibliotecas Java para o desenvolvimento de aplicativos. O código fonte não é compilado para bytecodes da máquina virtual Java (JVM). Os arquivos ao serem compilados são convertidos para a máquina virtual Dalvikm com a extensão .dex (Dalvik Executable).

A arquitetura da plataforma Android é composta por cinco camadas: Linux Kernel, Libraries, Android Runtime, Application Framework e Applications. Essa divisão pode ser observada na Figura 6.

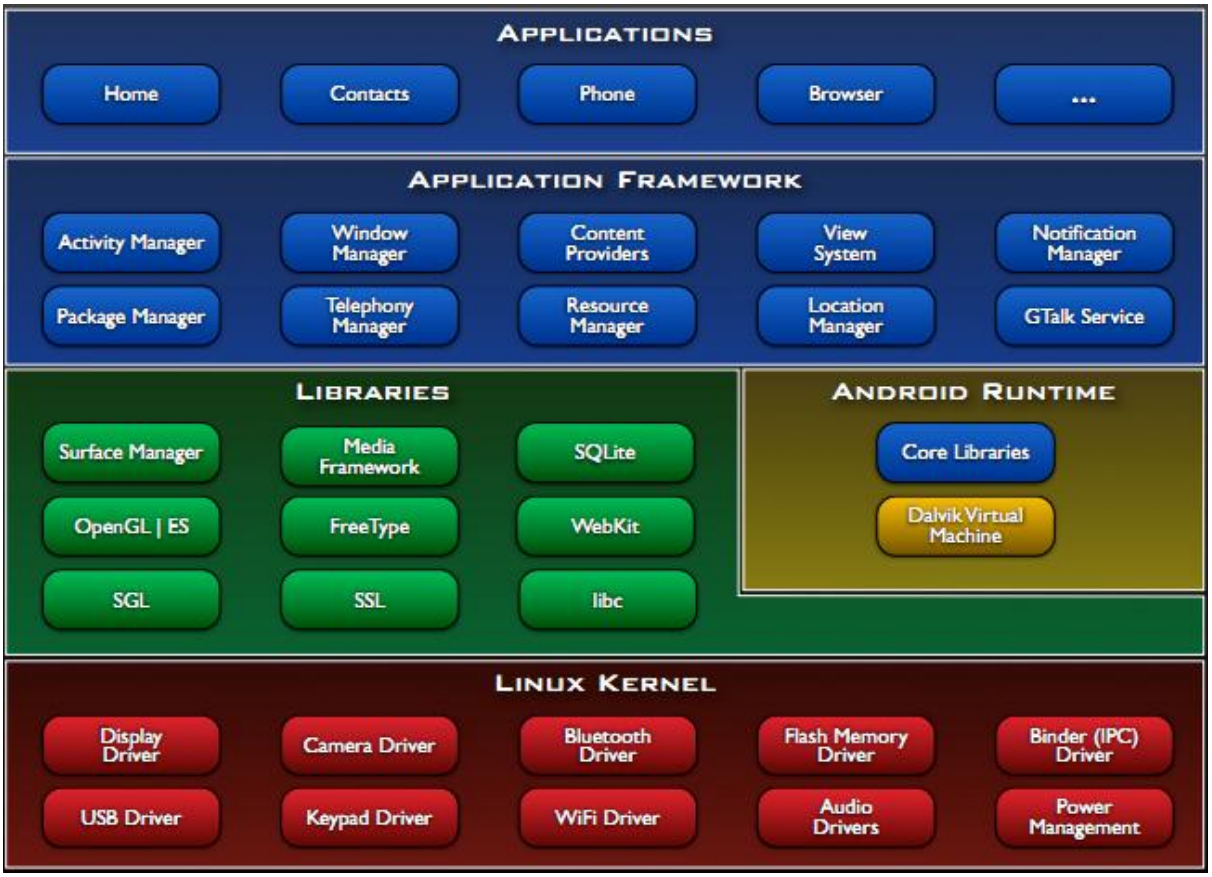


Figura 6 - Arquitetura android. Fonte: Http://developer.android.com/images/system-architecture.

A primeira camada *Linux Kernel* atua como uma camada de abstração entre hardware e o software do dispositivo. O Android é baseado no *kernel* 2.6 do Linux que é responsável pela segurança, gerenciamento de memória, gestão de processos, rede e drivers (Developer Android, 2012).

A camada *Libraries* é composta por um conjunto de bibliotecas C/C++ utilizadas por vários componentes do sistema. Estas funcionalidades são disponibilizadas aos desenvolvedores na camada *Application Framework* (Developer Android, 2012).

A camada *Android runtime* é o ambiente de execução da plataforma *Android*, composta por um conjunto de funcionalidades fornecidas pelas bibliotecas da linguagem de programação Java, além da máquina virtual *Dalvik*, onde cada aplicação é uma instancia da maquina virtual. Cada aplicação é executada como um processo distinto dos demais (Developer Android, 2012).

A camada *Application Framework* contêm todos os recursos e APIs utilizadas pelos aplicativos tais como botões, caixas de texto e outros componentes para composição do *layout* das aplicações, podem ser encontrados também provedores de conteúdo e gerenciadores de recursos, notificações e pacotes. Tal arquitetura foi projetada para simplificar a reutilização de componentes, fazendo com que qualquer aplicativo possa publicar novos componentes e qualquer aplicação fazer uso deles (Developer Android, 2012).

A camada *Application* contém uma grande variedade de aplicações, incluindo cliente email, SMS, mapas, calendário, *browser*, agenda de contatos, entre outros, todas escritas em linguagem Java (Developer Android, 2012).

5.4. BANCO DE DADOS POSTGRESQL

Os primeiros bancos de dados surgiram na década de 60 para suprir a necessidade de armazenamento de dados que antes eram em arquivos de texto, sem nenhum controle de redundância. A partir do surgimento do banco de dados, foi possível armazenar os dados de forma consistente, facilitando assim seu gerenciamento (HEUSER, 2004). A Figura 7 exibe a Interação de um banco de dados com usuário.

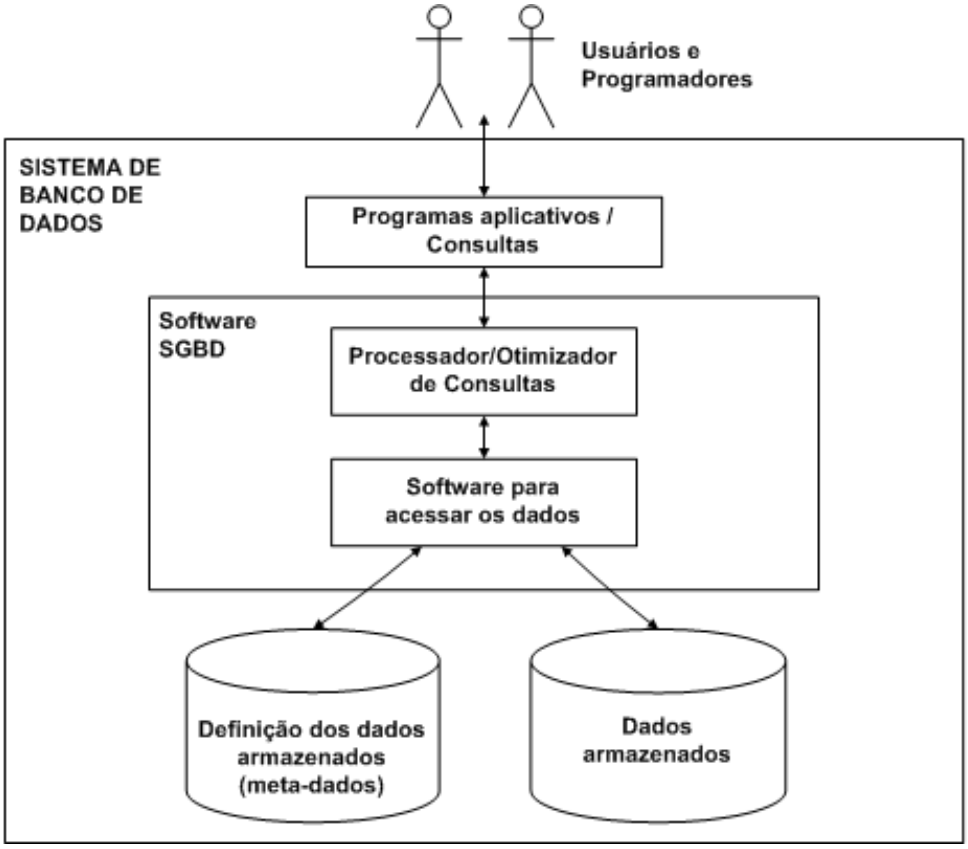


Figura 7 - Interação de um banco de dados com usuário. Fonte: http://www.ime.usp.br.

O PostgreSQL é um sistema gerenciador de banco de dados objeto-relacional de código aberto. Tem mais de 15 anos de desenvolvimento ativo e uma arquitetura de forte reputação de confiabilidade, integridade de dados e conformidade aos padrões. Pode ser executado na maioria dos sistemas operacionais, incluindo GNU/Linux, Unix (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), e Windows. Possui interfaces nativas de programação para C/C++, Java, DotNet, Perl, Python, Ruby, Tcl, ODBC entre outros (POSTGRESQL, 2014).

A Figura 8 apresenta a *interface* do psql usado para manipulação dos dados armazenados no banco de dados. Com esta aplicação é possível consultar os registros das tabelas, estrutura da tabela, e realizar o todo gerenciamento do banco de dados.

```
agendovc_development=# select * from colaboradores;
 id | empresa_id | usuario_id | created_at |         updated_at         | ativo |    cargo     | foto_file_name
----+------------+------------+------------+----------------------------+-------+--------------+----------------
 12 |          5 |         25 |            | 2014-08-01 08:40:02.7768   | t     | Manicure     |
 13 |          5 |         26 |            | 2014-08-01 08:40:16.816796 | t     | Cabeleireiro |
 10 |          4 |         19 |            | 2014-08-01 08:41:39.397611 | t     | Manicure     |
 14 |          6 |         17 |            | 2014-08-01 17:49:22.333958 | t     | Manicure     |
 15 |          6 |         18 |            | 2014-08-01 17:49:39.145664 | t     | Manicure     |
 16 |          6 |         24 |            | 2014-08-01 17:49:51.813977 | t     | Cabeleireiro |
```

Figura 8 - PSQL.

5.5. GIT

O Git é um sistema distribuído de controle de versão livre e de código aberto projetado para lidar com tudo, desde pequenos a grandes projetos com velocidade e eficiência (Git, 2014).

Segundo Mason (2006), o sistema de controle de versões (SCV), consiste basicamente, em um local para armazenamento de artefatos gerados durante o desenvolvimento de sistemas de software.

5.6. BITBUCKET

O Bitbucket é um site de hospedagem para o sistema de controle de versão distribuído para Git e Mercurial. O serviço inclui rastreador de problemas e Wiki, bem como a integração com vários serviços populares como Basecamp, Flowdock e Twitter (Bitbucket, 2014).

Através da sua interface é possível realizar a análise do código alterado e discutir com a equipe sobre a alteração realizada e a análise da atividade realizada no projeto. A figura 9 exibem os projetos disponíveis para o usuário autenticado e suas atividades recentes. A figura 10 exibem as informações dos *commits* realizados no projeto, permitindo visualizar o autor, mensagem e a data do *commit*.

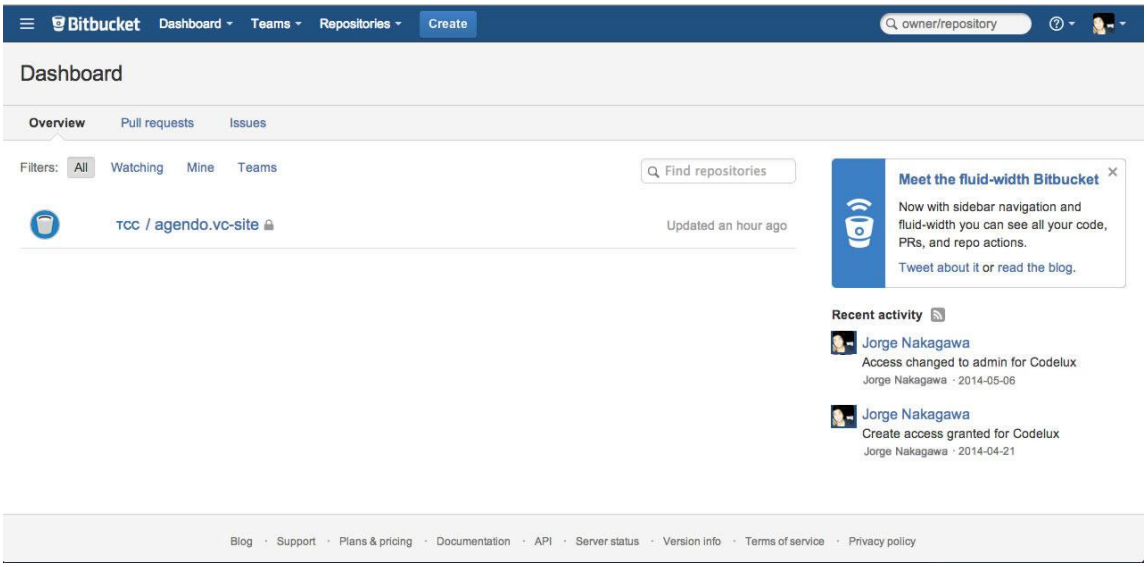


Figura 9 - Página de gerenciamento de projeto - Bitbucket.

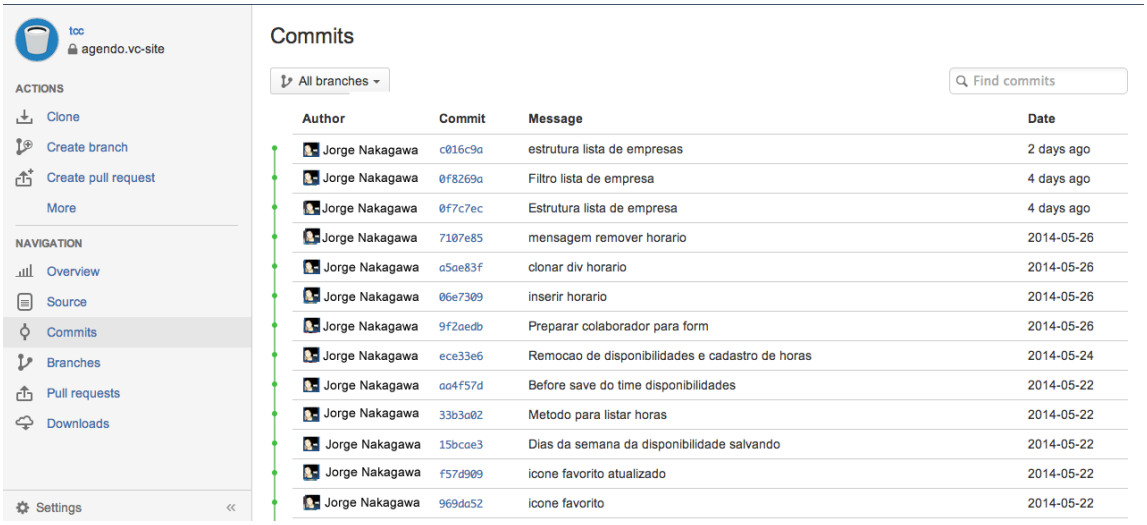


Figura 10 - Página de consulta de commits realizados no projeto.

5.7. CODESHIP

CodeShip é um serviço de hospedagem de integração contínua, que ao verificar a alteração no código fonte, executa processos pré-configurados como atualização das dependências, build automático, execução dos testes e publicação do projeto (CodeShip, 2014). Na Figura 11 é possível verificar todos os commits realizados, juntamente com a descrição, usuário, data e hora, e resultado.

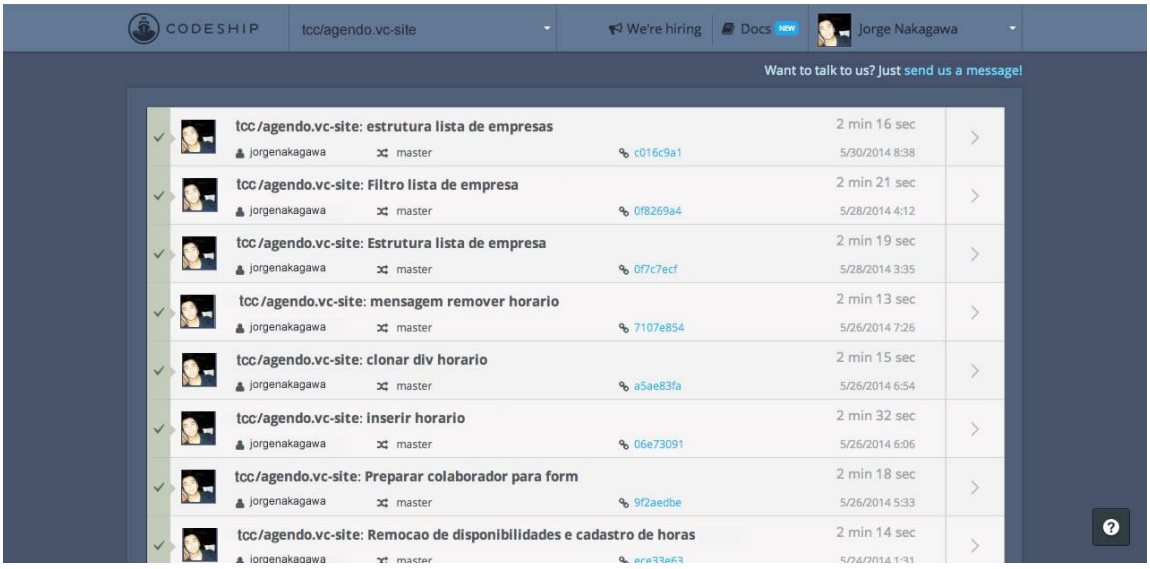


Figura 11 - Página de listagem de commits do projeto.

5.8. HEROKU

Heroku é uma plataforma de serviços em nuvem (PaaS), que suporta várias linguagens de programação (Heroku, 2014). Saleforce.com é o proprietário da Heroku, o qual é uma das primeiras plataformas em nuvem, que iniciou o seu desenvolvimento em junho de 2007, quando o suporte de linguagem era apenas para Ruby, mas iniciou o suporte para Java, PHP, Python, Scala, Node.js e Clojure. Heroku também é uma hospedagem para aplicação Ruby on Rails, com suporte a add-ons para o melhor gerenciamento da aplicação publicada. A Figura 12 exibe as configurações da máquina virtual utilizada pela aplicação *Web.*

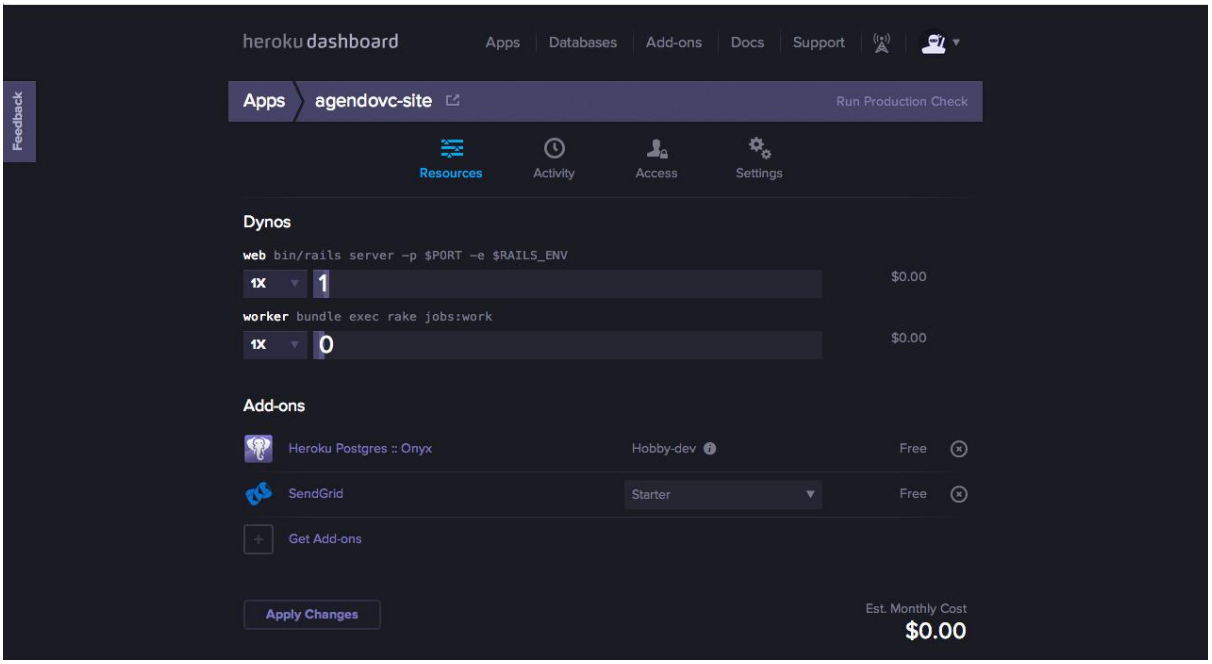


Figura 12 - Página de configuração do Heroku.

6. DESENVOLVIMENTO DA METODOLOGIA

6.1. EXPLORAÇÃO

Esta etapa do projeto é voltada principalmente para o levantamento de dados necessários para a definição do escopo do sistema. É nesta fase que as User Stories são criadas e também a utilização das ferramentas necessárias para o melhor entendimento do negócio. Para este projeto foi necessário a utilização de alguns diagramas UML para o melhor entendimento sobre a regra de negócio. Na Figura 13 é possível verificar as User Stories levantadas nesta fase.

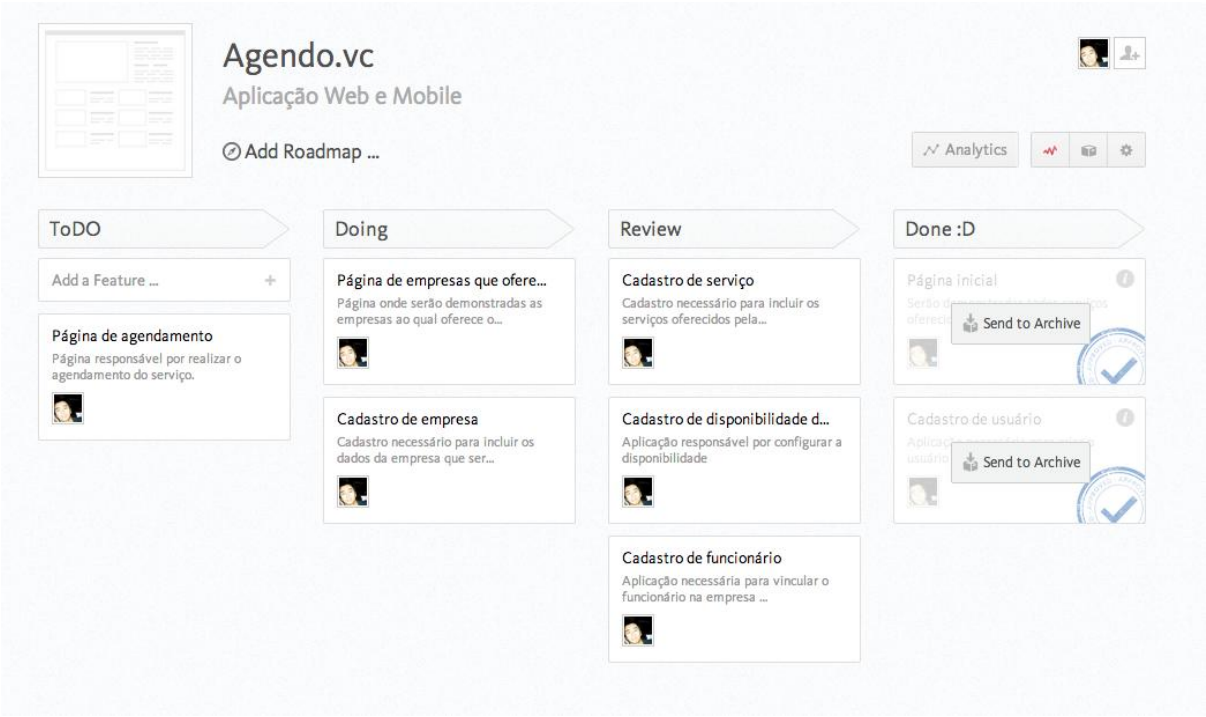


Figura 13 - Criação dos user stories no Blossom.io.

6.1.1. LEVANTAMENTO DE ATORES

O levantamento de atores na fase de especificação define quais serão os atores que irão interagir com o sistema. Esses atores podem ser um usuário humano do sistema ou outro sistema computacional e/ou dispositivos de hardware (GUEDES,

2008). Na Figura 14 estão representados os atores levantados que interagem diretamente com o sistema desenvolvido.

Figura 14 - Relação de atores do sistema.

- **Usuário empresa**: usuário que possui cadastro na base de dados do sistema, e possui o perfil de empresário. Possui a permissão de acesso nas páginas de empresa, serviço, funcionário e agenda;
- **Usuário consumidor**: usuário que possui cadastro na base de dados do sistema, e possui o perfil de consumidor. Possui a permissão de acesso nas páginas de visualização de serviço, e realizar o agendamento;
- **Usuário visitante**: usuário que não possui cadastro para autenticação no sistema, pode apenas navegar pela *web site*.

6.1.2. DIAGRAMA DE CASO DE USO

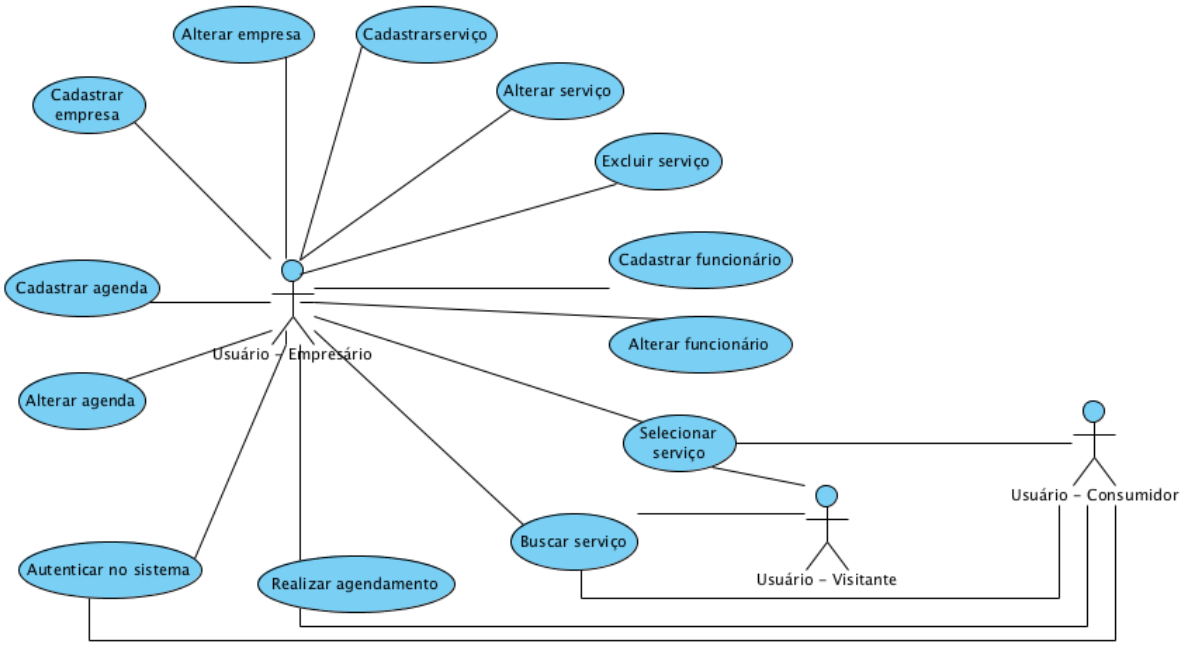


Figura 15 - Diagrama de Casos de Uso Geral.

Na Figura 15 é exibido o agrupamento de atores e funcionalidades do sistema. Por meio de relacionamentos é possível notar que existem restrições de acesso aos atores que interagem com o software.

6.1.3. DESCRIÇÃO DE CASO DE USO

O caso de uso na forma de descrição textual traz detalhamento do processo em que ocorre a interação do usuário com o software, auxiliando no melhor entendimento da regra e o seu objetivo. O exemplo de descrição de caso de uso é representado na Tabela 1, na qual é apresentada a interação do ator com o sistema para atingir o objetivo do caso de uso. O resultado pode ser positivo, quando o usuário atinge o objetivo, ou negativo, quando o ator não atinge o seu objetivo, assim o usuário deve realizar novamente os procedimentos. (As demais descrições dos casos de uso estão disponíveis no Anexo A.)

Tabela - 1 - Descrição do caso de uso UC

Caso de Uso	UC04 | Cadastrar Serviço
Finalidade	Inserir um novo serviço oferecido pela empresa.
Atores	Usuário autenticado.
Pré-condições	Os campos obrigatórios devem ser preenchidos no cadastro para que a inserção seja realizada com sucesso.
Pós-condições	Permitir a visualização do serviço na página de serviços oferecidos caso estiver ativo; O sistema deverá permitir que o usuário autenticado altere ou exclua um serviço cadastrado.
Sequencia de execução	1. O usuário deverá estar autenticado no sistema para ter acesso à página de cadastro de serviço; 2. Será exibida a página de cadastro de serviços, onde o usuário deverá preencher os dados do serviço que deseja adicionar na sua empresa, e os funcionários que oferece o serviço que está sendo cadastrado; 3. Ao acionar o botão de salvar, será disparado o método de validação dos dados a serem inseridos; 4. Se os dados informados apresentarem inconsistência, será exibida uma mensagem de aviso com relação de inconsistências para que o usuário possa informar os dados corretamente; 5. Com os dados corretos, o objeto será persistido na base de dados; 6. Será exibida a mensagem de cadastro realizado com sucesso.

6.1.4. REQUISITOS FUNCIONAIS

Os requisitos funcionais descrevem as funcionalidades que o sistema deve executar ou que são desejadas no sistema (MACORATTI, 2014). Para que o sistema satisfaça a expectativa do cliente, os requisitos funcionais são essenciais. A Tabela 2 apresenta os dados sobre requisitos funcionais que devem ser atendidos pelo sistema.

Tabela 2 - Requisitos Funcionais

ID	REQUISITOS FUNCIONAIS	ATOR
E01	Inserir, inativar e alterar dados da empresa.	Usuário - Empresa.
E02	Inserir, inativar, alterar dados dos funcionários.	Usuário - Empresa.
E03	Inserir, inativar, alterar dados dos serviços.	Usuário - Empresa.
E04	Inserir, excluir, alterar disponibilidade da empresa.	Usuário - Empresa.
E05	Visualizar serviços oferecidos pelas empresas na visão cliente.	Usuário - Empresa, Usuário - Consumidor, Usuário - Visitante.
E06	Inserir, inativar, alterar dados dos serviços oferecidos pela empresa.	Usuário - Empresa.
E07	Visualizar os serviços oferecidos.	Usuário - Consumidor, Usuário - Visitante, Usuário - Empresa
E08	Buscar o serviço.	Usuário - Consumidor, Usuário - Visitante, Usuário - Empresa
E09	Realizar o agendamento.	Usuário - Consumidor, Usuário - Empresa

6.1.5. REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais definem procedimentos para o melhoramento da qualidade do sistema (MACORATTI, 2014). A seguir na Tabela 3 são relacionados os requisitos não funcionais necessários para obtenção de um melhor resultado de satisfação do usuário na utilização do sistema.

Tabela 3 - Requisitos não funcionais

ID	REQUISITOS NÃO FUNCIONAIS	CATEGORIA
RN01N	O sistema deverá ser acessado por qualquer navegador de *internet*.	Usabilidade.
RN02N	O layout do front-end do sistema, deve ser desenvolvido utilizando apenas o HTML, Javascript e CSS.	Interface.
RN03N	O sistema deverá utilizar o SGBD PostgreSQL para armazenamento e manipulação de dados.	Padronização.
RN04N	A linguagem de programação utilizada no desenvolvimento será a Ruby.	Padronização.

6.1.6. LEVANTAMENTO DE CASO DE USO

Analisando os requisitos levantados, é possível concluir o objetivo que o sistema deve alcançar para satisfazer o usuário. Na Tabela 4, é especificado o levantamento dos casos de uso dos requisitos principais, ou seja, os serviços que compõem o sistema de agendamento de serviço.

Tabela 4 - Levantamento de casos de uso dos requisitos principais.

ID	Caso de Uso	Prioridade
UC01	Criar uma conta.	Alta
UC02	Configurar uma empresa.	Alta
UC03	Convidar o colaborador.	Alta
UC04	Cadastrar o serviço.	Alta
UC05	Configurar o serviço.	Alta
UC06	Realizar o agendamento.	Alta
UC07	Consultar as empresas que fornecem o serviço.	Alta
UC08	Inativar o colaborador	Alta
UC09	Cadastrar Disponibilidade	Alta

6.2. PLANEJAMENTO

Esta etapa é necessária para priorizar as estórias levantadas na etapa de exploração. No critério da seleção deve ser levado em conta, o que agrega de fato o valor no cliente, assim podemos certificar de que o desenvolvimento está o caminho correto (KUHN e PAMPLOMA, 2009).

6.3. ITERAÇÕES PARA VERSÕES

A fase responsável para o desenvolvimento da versão do software. As versões são compostas por Estórias selecionadas e somente será colocado em produção após serem validadas pelos testes unitários. Na Figura 16 pode ser visto o Diagrama de entidade de relacionamento e Diagrama de classe que foram elaborados de acordo com as Estórias selecionadas na iteração.

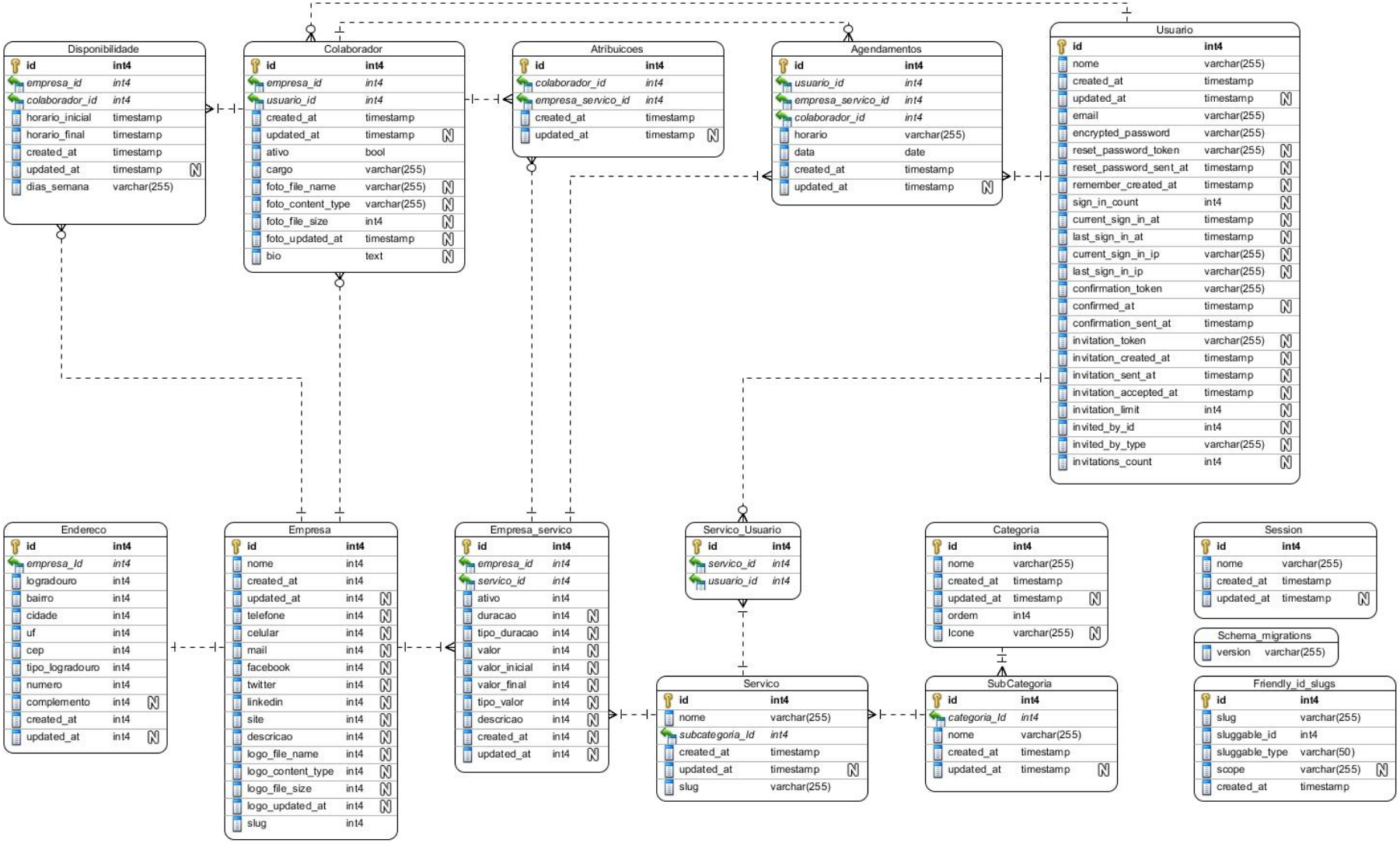


Figura 16 - Diagrama de entidade de relacionamento.

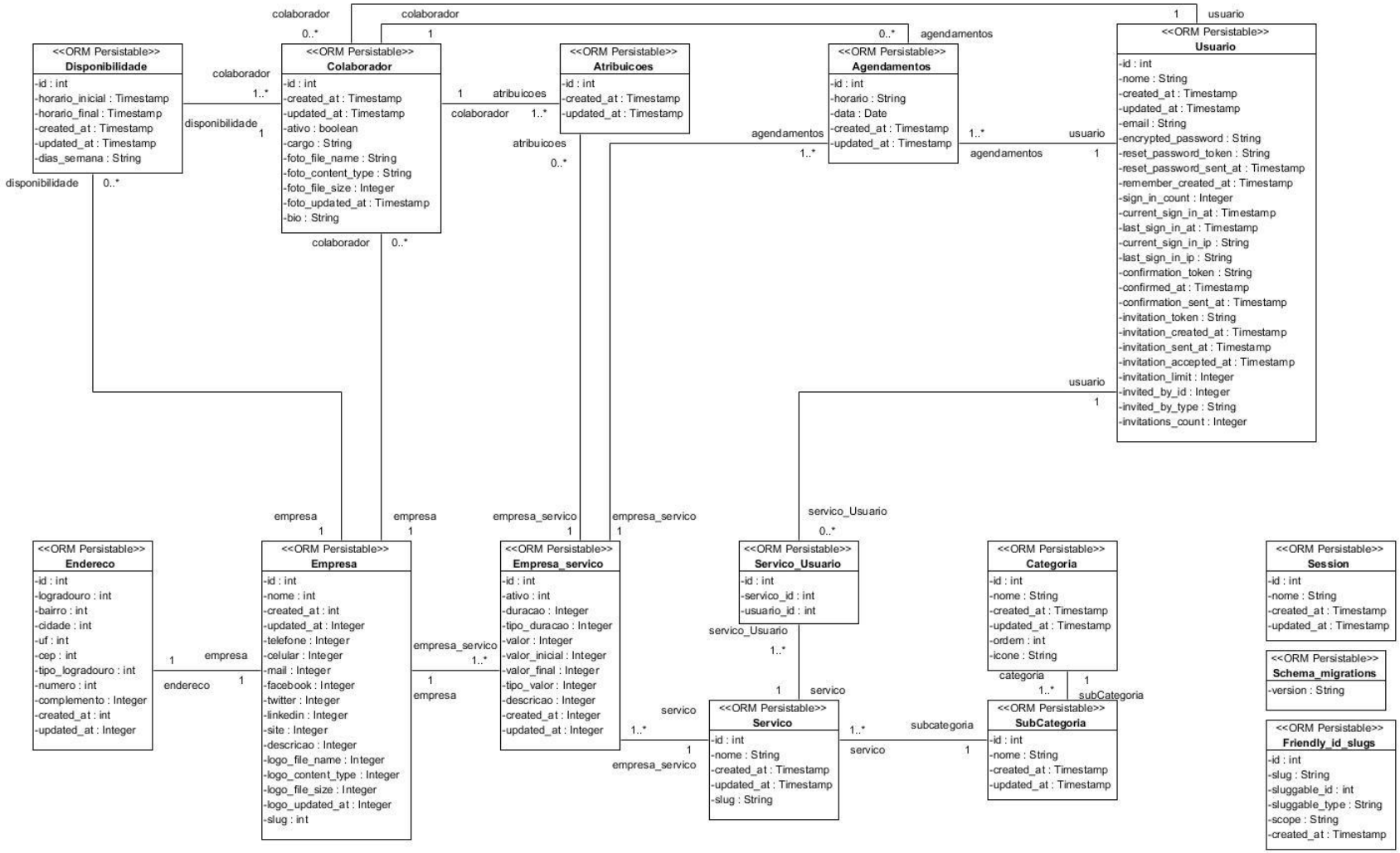


Figura 17 – Diagrama de classe

6.3.1. INTEGRAÇÃO CONTÍNUA

Integração Contínua é uma pratica de desenvolvimento de software na qual os membros de um time que integram o seu trabalho, pelo menos cada pessoa faz a integração diariamente - podendo haver múltiplas integrações por dia. Cada integração é verificada por um *build* automatizado (incluindo testes) para detectar erros de integração o mais rápido possível. Muitos times acham que essa abordagem leva a uma significante redução nos problemas de integração e permite que um time desenvolva software mais coeso e mais rapidamente (Martin Fowler, 2006).

Neste projeto foi adotada a prática de integração contínua, utilizando as ferramentas que auxiliam o uso desta prática. O ciclo se inicia com o Git ao qual é responsável por realizar o versionamento local do projeto, ou seja, o commit é realizado localmente, após realizar o *commit*, os arquivos são enviados para o repositório Bitbucket que é o responsável por armazenar e versionar todos os arquivos dos projetos, após o versionamento dos arquivos o Codeship detecta uma nova atualização de arquivos, baixa a atualização do código fonte do projeto para realizar *build* automático, executa os testes e realiza o *deploy* para uma determinada hospedagem configurada.

7. IMPLEMENTAÇÃO

A fase de implementação consiste em desenvolver o código fonte das funcionalidades especificadas anteriormente e suas *interfaces*. Ao fim da codificação e desenho de interface de interação com o usuário, é efetuada a integração entre funcionalidade e a interface. A seguir serão apresentadas algumas *interfaces* do sistema. As demais seguem o mesmo padrão por utilizarem o mesmo padrão de *layout*.

7.1. IMPLEMENTAÇÃO WEB

A Figura 18 exibe a página inicial da aplicação web, na qual são listados todos os serviços disponíveis agrupados de acordo com a sua categoria. Também é possível realizar a busca do serviço.

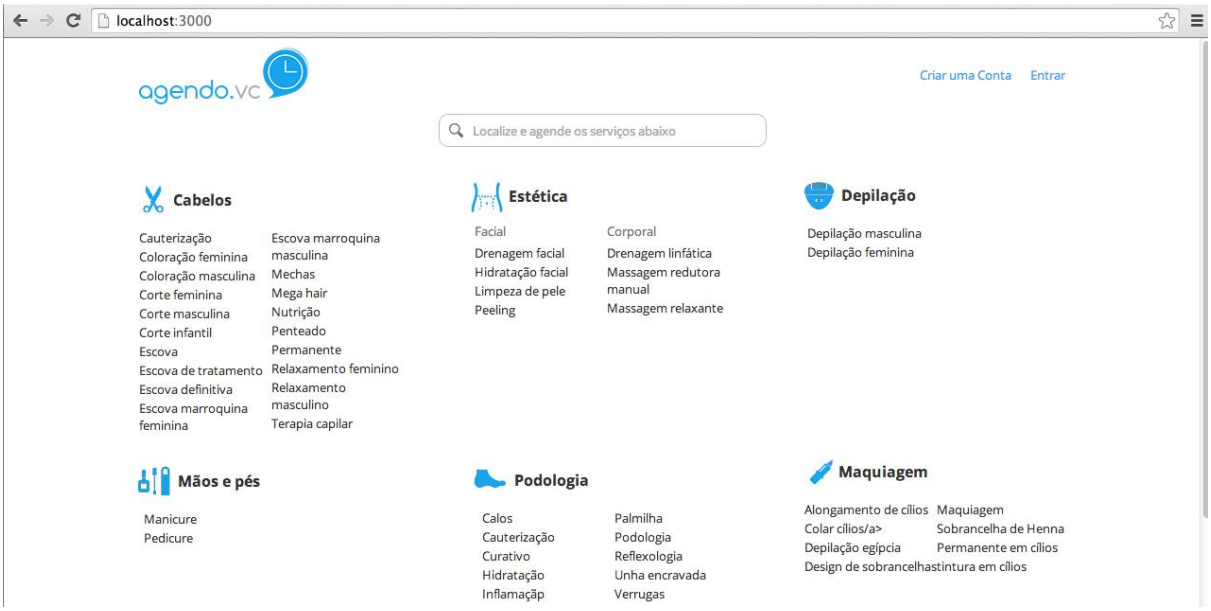


Figura 18 - Página inicial com os serviços disponíveis.

A figura 19 exibem todas as empresas cadastradas no sistema que oferece o serviço selecionado na página inicial, podendo filtrar os serviços de acordo com o tipo do prestador do serviço (Empresa ou profissional liberal), preço e disponibilidade.

Figura 19 - Listagens dos serviços oferecidos.

A Figura 20 exibem os dados da empresa como: localização, disponibilidade, sobre, valor de cada serviço e as imagens da empresa cadastrada. Ao pressionar o botão agendar, o usuário será encaminhado à página de confirmação.

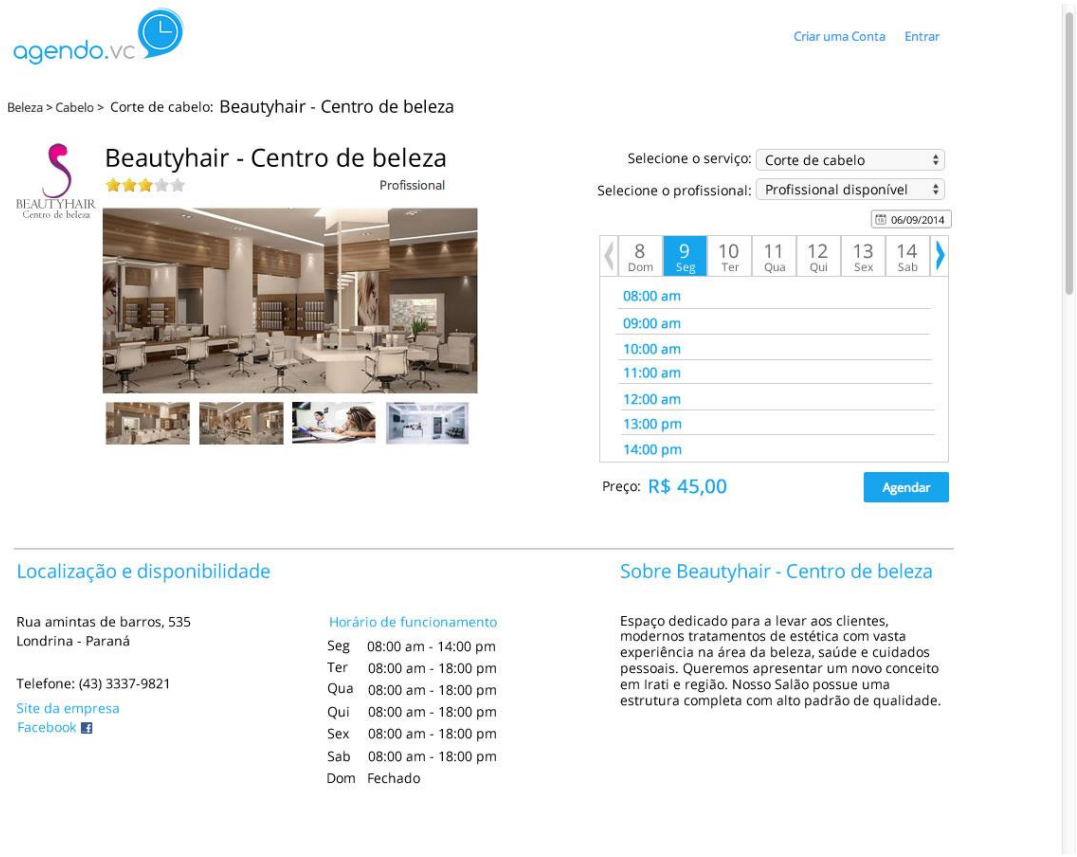


Figura 20 - Página de agendamento de serviço.

A Figura 21 exibe o resumo do agendamento. Demonstrando de forma detalhada a descrição do serviço, empresa fornecedora de serviço, horário agendado, o valor total do serviço, sobre a empresa, telefone de contato, site, rede social e localização.

Figura 21 - Página de confirmação de agendamento de serviço.

A Figura 22 exibe o formulário do cadastro da conta no sistema, inserindo somente o nome, sobrenome e e-mail. Ao solicitar o cadastro, o e-mail de confirmação será encaminhado para o e-mail de cadastro.

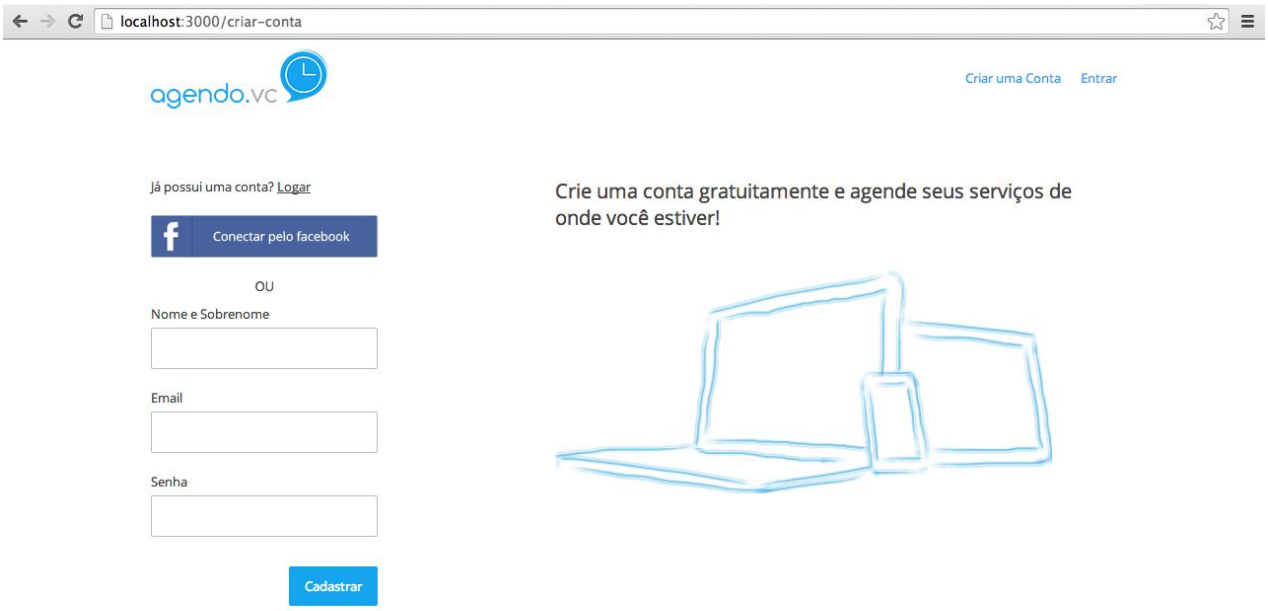


Figura 22 - Página de autenticação do usuário.

A Figura 23 exibe os perfis disponíveis no sistema, que é solicitado ao realizar o cadastro, para que o perfil selecionado seja associado à conta de acordo com o perfil escolhido, o sistema terá comportamentos diferentes em várias rotinas do sistema.

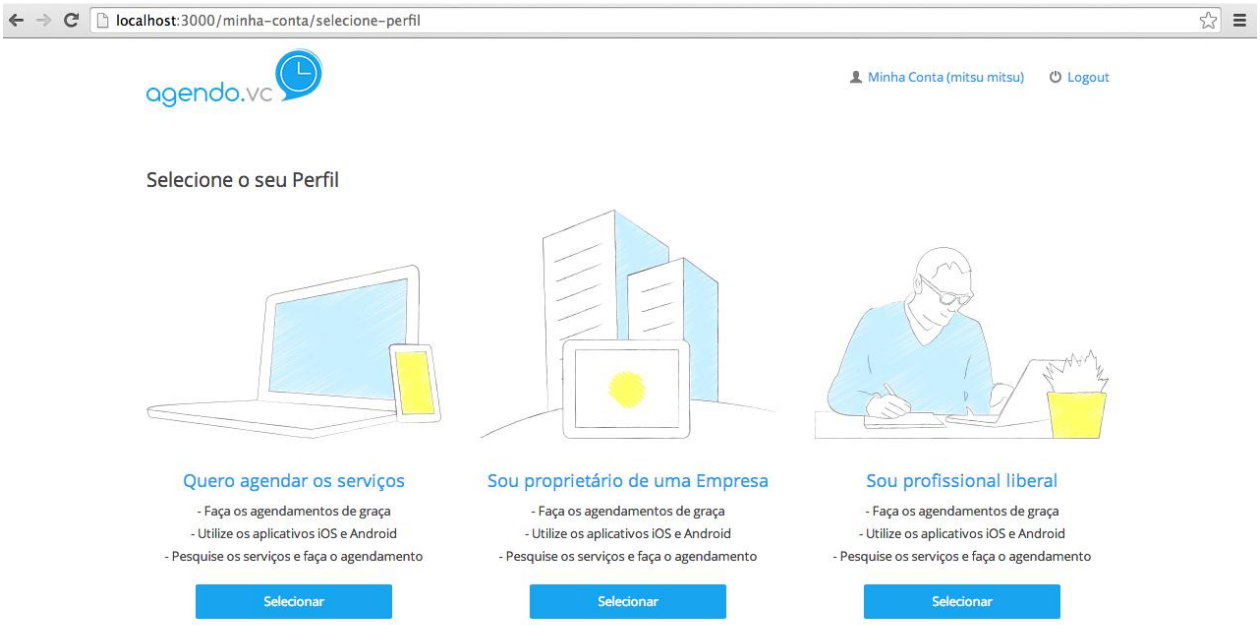


Figura 23 - Página de seleção de perfil do usuário.

A Figura 24 exibe página inicial da configuração da conta, contendo os campos para preencher os dados pessoais do usuário da conta. Alguns campos desta página serão exibidos para a empresa de serviço quando o realizar o agendamento.

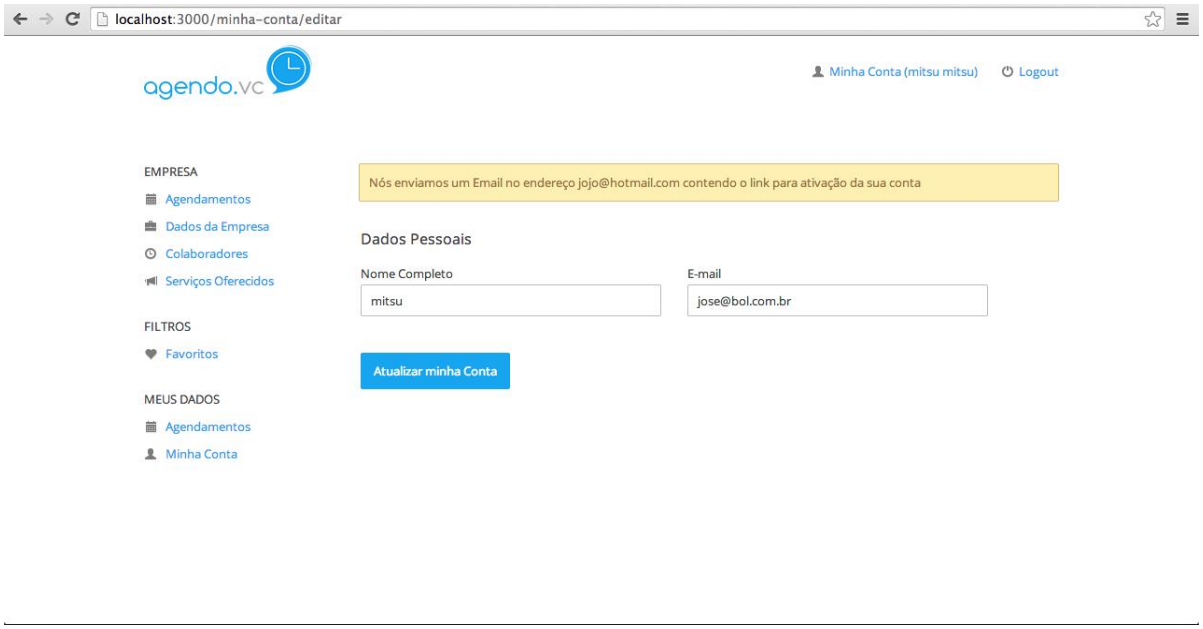


Figura 24 – Página inicial de configuração da conta – Empresa.

As Figuras 25, 26 e 27 exibem a página da configuração dos dados da empresa. A página permite a inserção do nome, logo, telefone, e-mail, endereço, descrição e redes sociais. Todos estes dados inseridos são exibidos na página de informações da empresa.

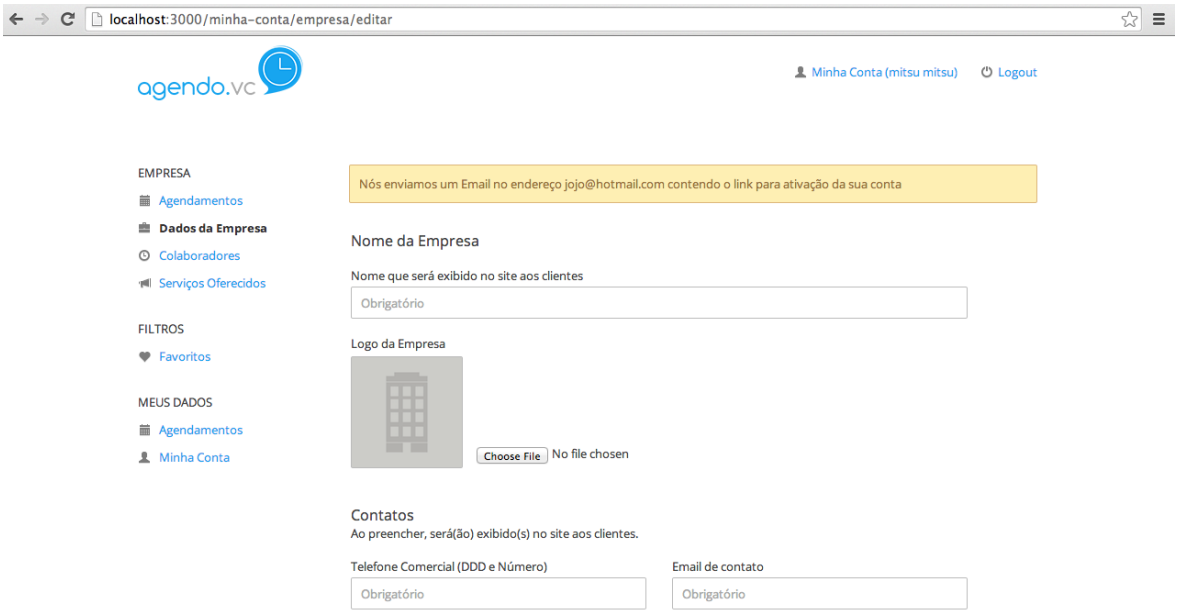


Figura 25 - Página de cadastro de empresa - Área Empresa.

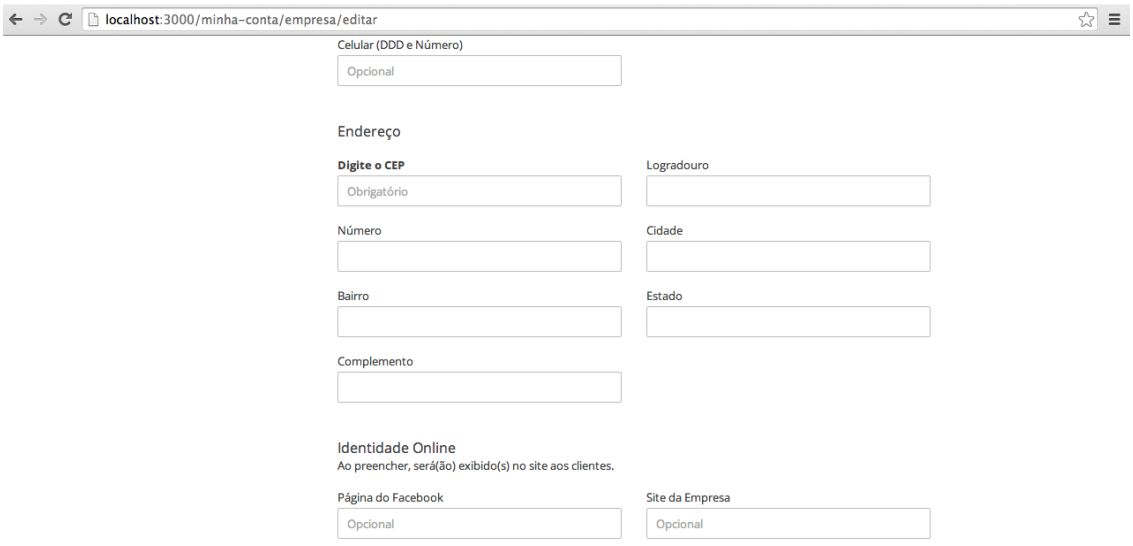


Figura 26 - Página de cadastro de empresa - Área Empresa (Continuação).

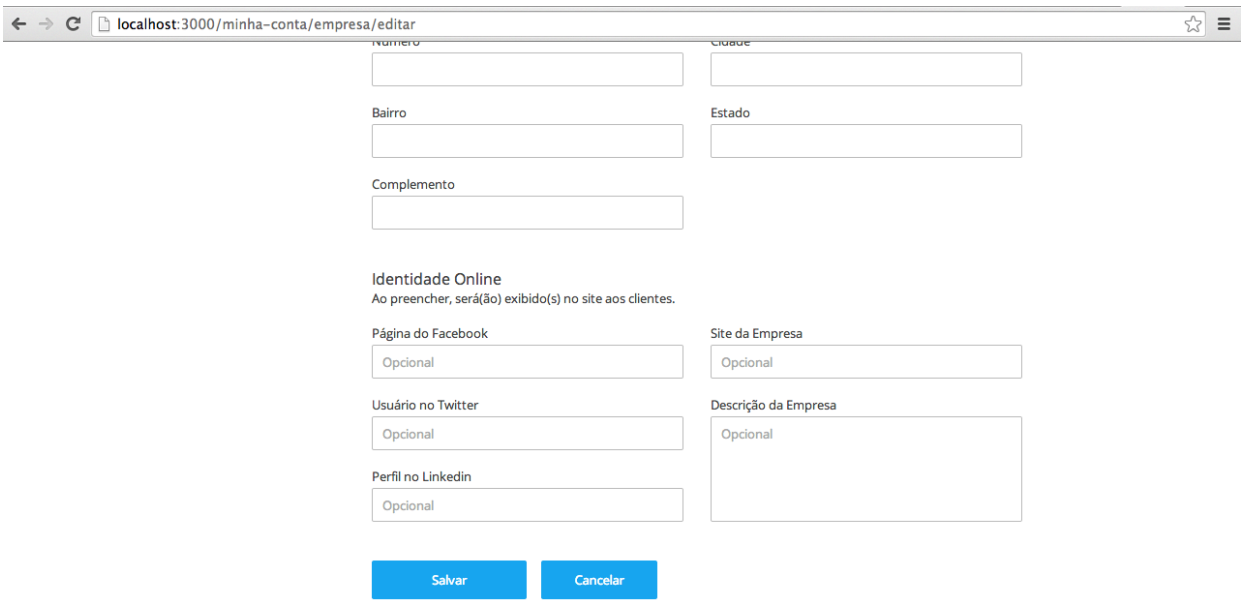


Figura 27 - Página de cadastro de empresa - Área Empresa (Continuação).

A Figura 28 exibe a página de cadastro de colaborador da empresa. Todos os colaboradores da empresa são listados com os dados básicos como: nome, cargo, situação.

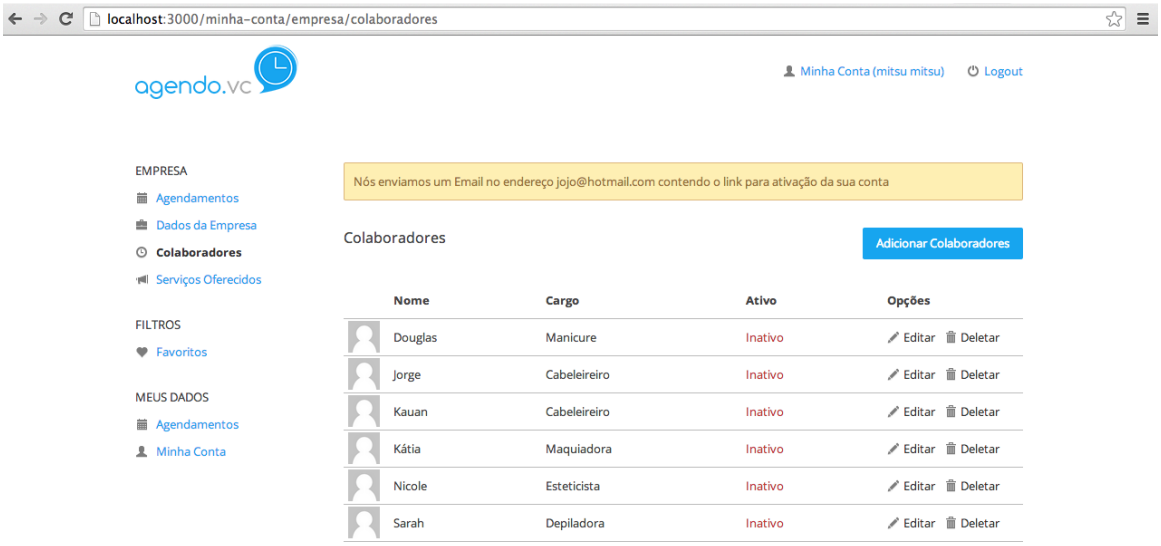


Figura 28 - Página de cadastro de colaborador - Área Empresa.

A Figura 29 exibe a página de envio de convite ao colaborador. A partir desta página é possível encaminhar um convite para o colaborador para fazer parte da empresa. Ao encaminhar o convite o e-mail é enviado com o link para a confirmação e caso o colaborador convidado não tenha uma conta no sistema, o link do cadastro de conta é encaminhado.

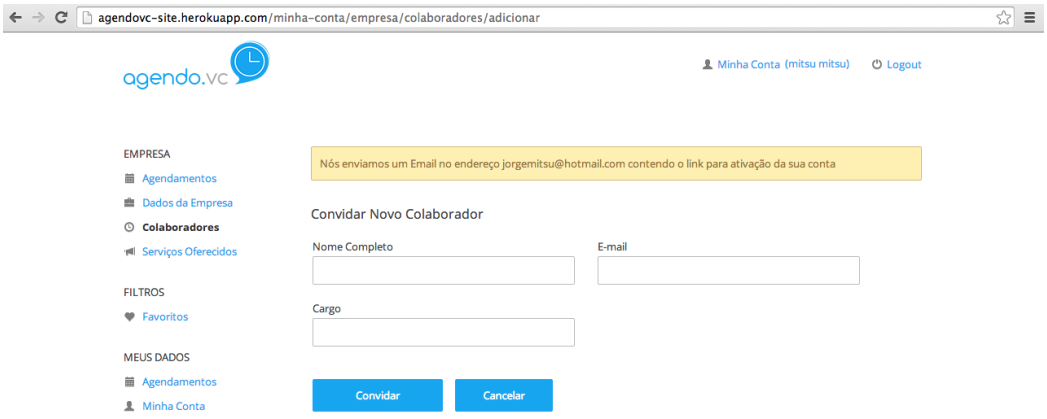


Figura 29 - Página de envio de convite ao colaborador.

A Figura 30 exibe a página de configuração de colaborador. A partir desta página é possível configurar os dados do colaborador referente à empresa. Podendo configurar dados como cargo, horário de trabalho, informações pessoais e profissionais.

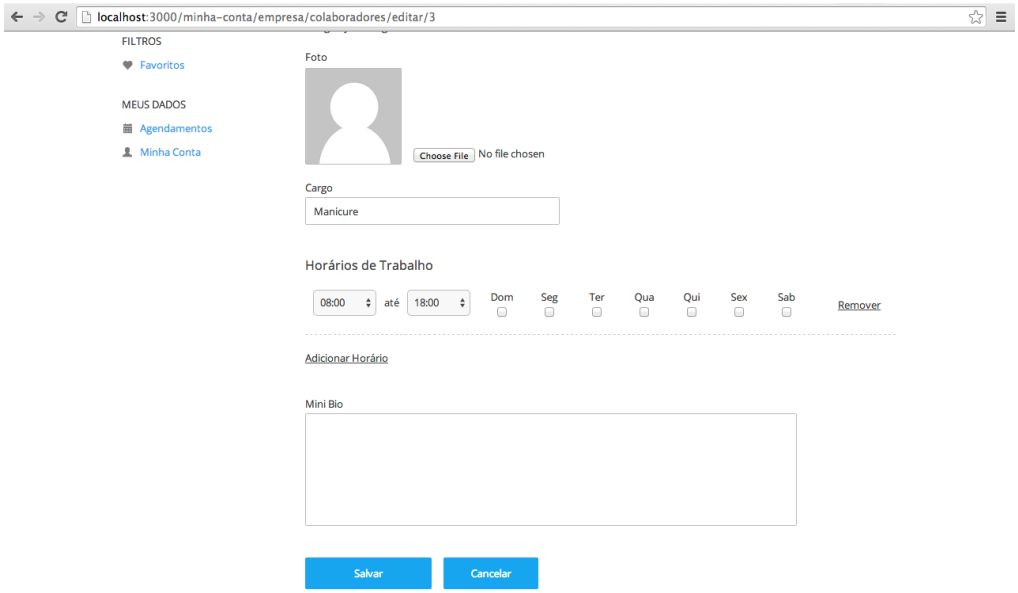


Figura 30 - Página de configuração de colaborador - Área Empresa.

A Figura 31 exibe a página de listagem de serviços oferecidos. A partir desta página é possível visualizar todos os serviços cadastrados pela empresa, exibindo dados como duração, valor e situação.

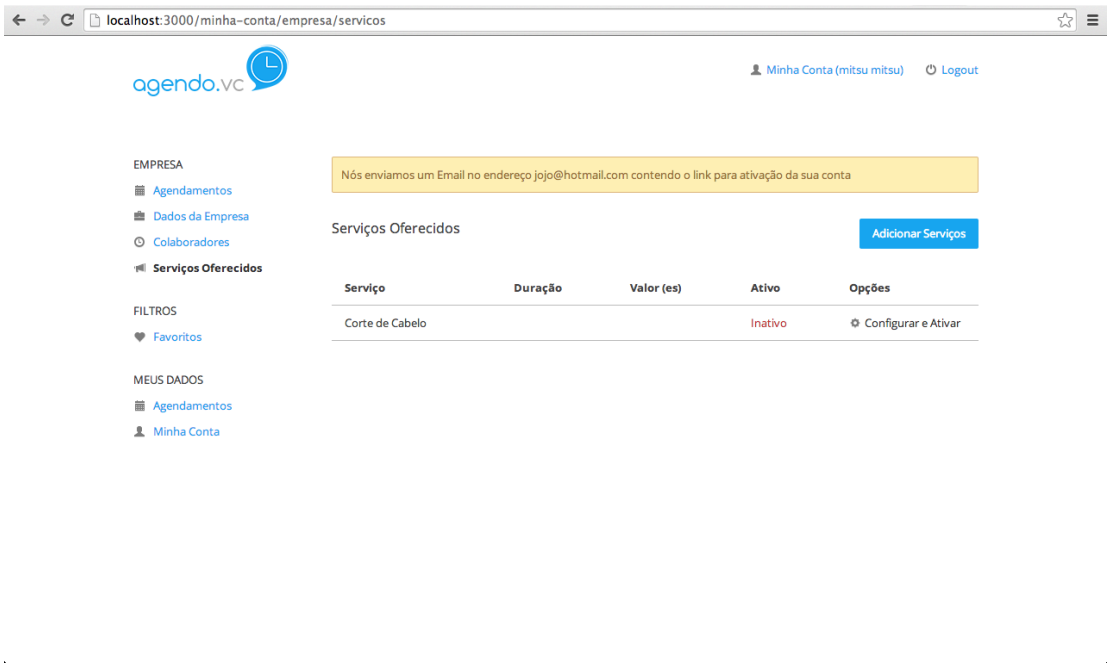


Figura 31 - Página de listagem de serviços - Área Empresa.

A Figura 31 exibe a página de é possível adicionar o serviço oferecido pela empresa. Ao confirmar o serviço será carregada a página de configuração.

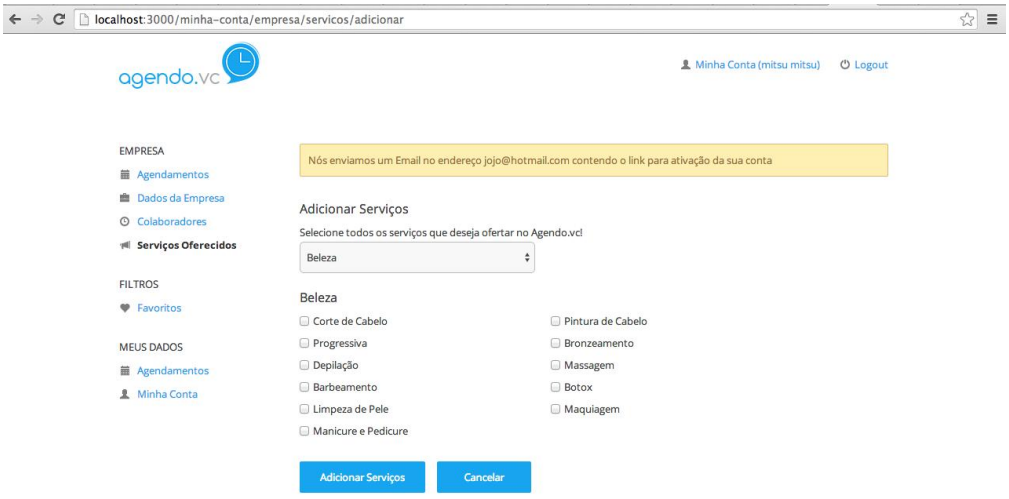


Figura 32 - Página de configuração de serviço - Área Empresa.

A Figura 32 exibe a página de configuração do serviço. A partir desta página é possível realizar a configuração do serviço, podendo configurar os dados como valor, duração, detalhamento do serviço, e permite vincular os colaboradores que oferecem este serviço na empresa.

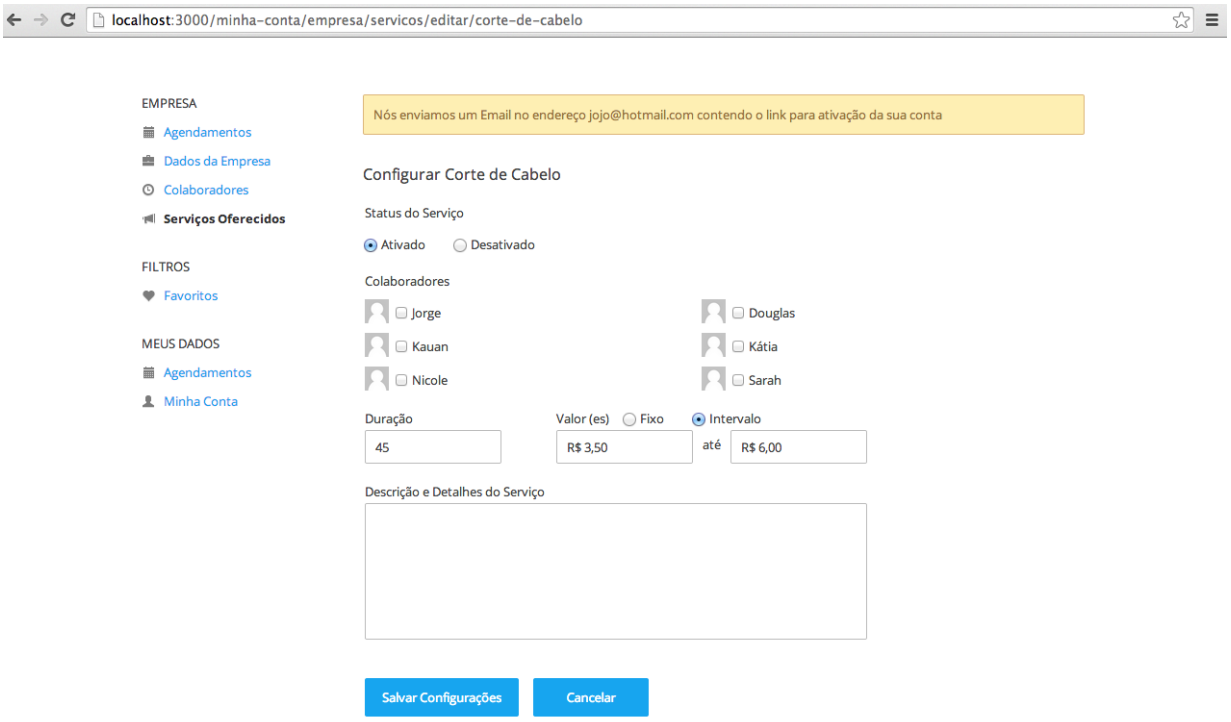


Figura 33 - Página de configuração de colaborador - Área Empresa.

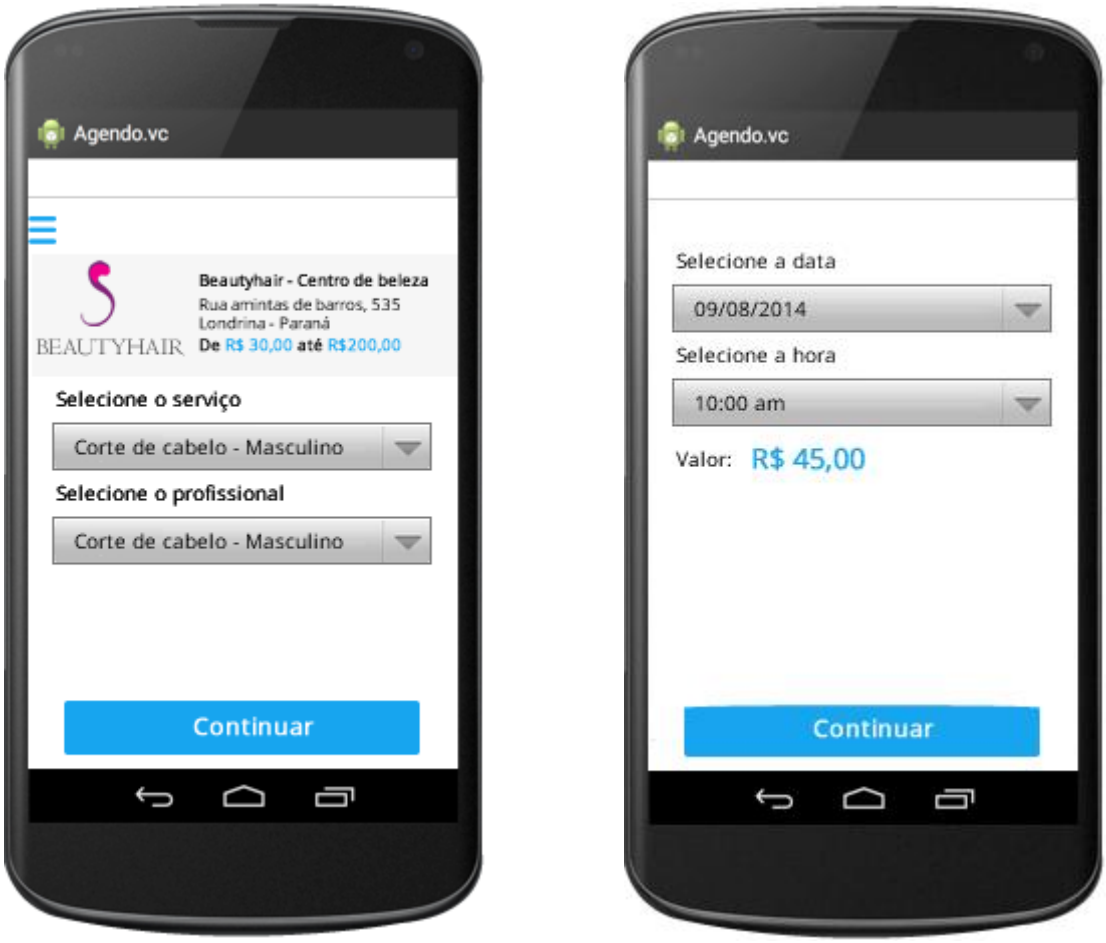
A Figura 33 exibe a página de configuração do serviço, permitindo atribuir o status, colaboradores que oferecem este serviço, duração, valor fixo, valor com intervalo e descrição do serviço oferecido.

7.2. IMPLEMENTAÇÃO MOBILE

(a) **(b)**

Figura 34 - As categorias dos serviços oferecidos e listagem de empresas.

A Figura 34 (a) exibe a interface inicial do aplicativo. Permite a busca do serviço desejado e a lista as categorias dos serviços oferecidos. A figura 34 (b) exibe uma listagem das empresas que oferecem o serviço selecionado na tela anterior exibindo o logo da empresa, nome da empresa, localização e os valores dos serviços.



(a) **(b)**

Figura 35 - Informações sobre o serviço, profissional, data e hora da empresa.

A Figura 35 (a) exibe a Interface do agendamento com os dados da empresa, serviços ofertados e o profissional da empresa. A Figura 35 (b) exibe a data, a hora e o valor do serviço selecionado.

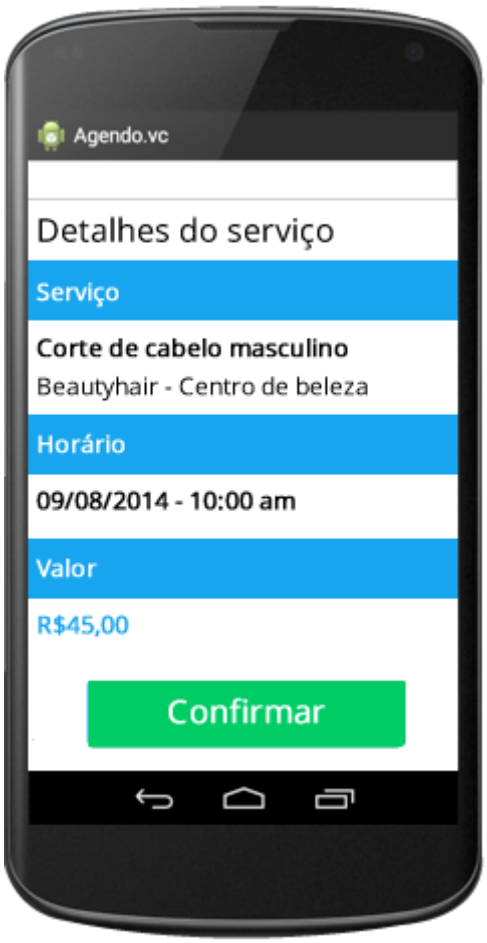


Figura 36 - Detalhamento sobre o serviço a ser agendado.

A Figura 36 exibe a interface de confirmação do agendamento. Exibindo os detalhes do agendamento de forma resumido a informação geral do agendamento. O agendamento será registrado ao pressionar o botão confirmar.

8. CRONOGRAMA E RECURSOS ALOCADOS

8.1. CRONOGRAMA

Atividades a serem realizadas durante o desenvolvimento do trabalho de diplomação:

1. Revisão bibliográfica
2. Requisitos
3. Análise
4. Projeto
5. Implementação
6. Teste
7. Redação da monografia
8. Depósito do trabalho de diplomação

Atividades	Dez/13				Jan/14				Fev/14				Mar/14				Abr/14				Mai/14			
Revisão bibliográfica	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Redação da monografia	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Depósito do trabalho de diplomação																								■
Requisitos	⊘				⊖				⊜				◐				□				×			
Análise		⊘				⊖				⊜				◐				□				×		
Projeto			⊘				⊖				⊜				◐				□				×	
Implementação				⊘				⊖				⊜				◐				□				×
Teste				⊘				⊖				⊜				◐				□				×

O cronograma de execução do desenvolvimento do software foi dividido em 6 iterações, sendo que para cada iteração foram executadas as seguintes etapas: levantamento de requisitos, análise, projeto, teste e implementação. No requisitos são registrados na forma de *User Stories* no *Blossom.io.* Na etapa de projeto é feito o filtro dos *User Stories* para selecionar os requisitos que agregam os valores ao usuário do sistema e após a seleção foi criado o diagrama de caso de uso. Na etapa de projeto foi criado o diagrama de classe e diagrama de entidade de relacionamento para o melhor entendimento da regra de negócio e o comportamento dos objetos. Na etapa de teste foram criados os testes unitários de acordo com a regra de negócio levantados nas etapas anteriores para garantir o comportamento correto dos objetos. Na etapa de implementação é realizada a implementação dos requisitos para que o código seja aprovado pelos testes unitários escritos na etapa anterior, mantendo assim o comportamento correto dos objetos de acordo com a necessidade da regra de negócio.

⦸ - Iteração 1, nesta etapa foram levantados os requisitos gerais do comportamento do sistema que foi seguido até o final do projeto. O resultado desta iteração é a página de autenticação do usuário, página de cadastro de usuário página inicial do sistema e o *layout* da página de listagem das empresas.

⊖ - Iteração 2, nesta etapa foram levantados requisitos para realizar o gerenciamento dos usuários e a definição de métodos de segurança dos usuários de sistema. O resultado desta iteração é a página de cadastro de conta e página de autenticação da conta.

⍟ - Iteração 3, nesta etapa foram levantados os requisitos para definir o comportamento dos serviços que são ofertados pelas empresas. O resultado desta iteração é a página de listagem e cadastro de serviço.

⍟ - Iteração 4, nesta etapa foram levantados os requisitos essenciais para realizar o gerenciamento geral dos dados da empresa. Para este requisito foram envolvidas as regras de negócios referentes aos colaboradores, serviços, horários, agendamentos e dados da empresa.

□ - Iteração 5, nesta etapa foram levantadas requisitos para desenvolver a página de agendamento do serviço. O resultado desta iteração é a página de agendamento e página de confirmação de agendamento.

× - Iteração 6, nesta etapa foram analisados os requisitos para o desenvolvimento da aplicação mobile para o usuário consumidor para permitir o

agendamento via interface móbile. O resultado desta iteração é o aplicativo mobile para o usuário consumidor capaz de realizar o processo de agendamento.

8.2. RECURSOS ALOCADOS

Visando a modelagem, codificação e documentação do sistema, para o desenvolvimento da metodologia, foram utilizados os seguintes recursos de software e hardware:

8.2.1. RECURSO DE SOFTWARE

1. Sistema Operacional Mac OSX 10.9.2;
2. Microsoft Word 2007 profissional;
3. Microsoft Visio;
4. Ruby 2.1.1p76;
5. Rails 4.1.0;
6. Visual paradigm;
7. Android studio v0.3.2;
8. Java development Kit versão 1.7;
9. Android Software Development kit 3.4;
10. Sublime text 2;
11. PostgreSQL 9.1.3;
12. Google Chrome 18.0;
13. Mozzila Firefox 11.0;
14. Internet Explorer 6/7.

8.2.2. RECURSO DE HARDWARE

15. Notebook Macbook Intel I7 2.9 Ghz, 8Gb de RAM, 800GB de disco rígido;
16. Smartphone Nexus 4.

9. CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Em geral a informatização das empresas favorecidos oferece um diferencial no mercado com o gerenciamento das informações, automatização do processo e facilidade para o consumidor faz com que a empresa possa seguir no caminho de sucesso.

Neste trabalho foi apresentado o sistema de gerenciamento de agendamento de serviços. A partir do sistema é possível realizar o controle da agenda, colaboradores da empresa, serviços oferecidos e dados como valor, tempo e descrição. Os dados configurados pela empresa são exibidos aos usuários consumidores, permitindo que os consumidores tenham maior comodidade para realizar os agendamentos, sem a necessidade de entrar em contato via telefone ou até mesmo se encaminhar ao salão, que é o caso mais comum na atualidade.

Para o desenvolvimento do sistema web foi necessário o estudo da linguagem Ruby e do framework Rails para o desenvolvimento web. Para o desenvolvimento mobile foi necessário aprimorar o conhecimento sobre a Android.

Com relação a linguagem Ruby, utilizada para desenvolver toda parte do backend, a mesma apresentou o resultado acima do esperado, uma linguagem elegante, focado em trazer a essência da programação orientada a objetos fazendo com que tudo se torne natural, de forma simples, cumprindo com um dos desejos do criador da linguagem (Matz, 2000), “o Ruby é simples na aparência, mas muito complexo no interior, tal como o corpo humano”.

O aprofundamento dos conhecimentos adquiridos durante o curso foi necessário para o desenvolvimento deste projeto, aplicando-os em suas respectivas etapas de projeto, desde o levantamento de requisitos, projeto e desenvolvimento. A aplicação da metodologia de desenvolvimento permitiu a melhor compreensão da prática no processo de desenvolvimento de software, a importância de um processo iterativo, de um requisito claro, definido e focado no que realmente agrega o valor no sistema e o uso do diagrama UML para o melhor entendimento da regra de negócio.

A falta de domínio nas linguagens utilizadas para o desenvolvimento web e mobile foi um grande desafio, fazendo com que o prazo estipulado inicialmente não fosse cumprido, devido a necessidade de um estudo intensivo e pesquisas necessárias para a compreensão da linguagem.

Em comparação aos sistemas apresentados no capítulo 3, o software desenvolvido tem como vantagens o fato de ser web e mobile, possibilitando o agendamento via interface móbile, trazendo maior comodidade e acessibilidade aos usuários.

O software desenvolvido ainda não está disponível comercialmente, pois é necessário o melhor aprofundamento da regra de negócio, e melhorias necessárias para oferecer melhor comodidade aos usuários. Existem grandes possibilidades do software ser comercializado futuramente.

O software poderá estender para outras áreas, permitindo com que outras áreas comerciais possam utilizar o sistema para gerenciar o agendamento do seu serviço. O aplicativo mobile sofrerá alterações para que possa realizar o gerenciamento via interface mobile, oferecendo melhor comodidade aos donos de negócio.

Novos recursos serão incluídos para melhor atender a realidade do dia a dia do profissional, permitindo melhor gerenciamento e eficiência no processo de agendamento. A integração com o sistema de pagamento online será necessária para que o software se torne completo, gerenciando assim todo o ciclo de agendamento de serviço, desde o seu início onde garante o atendimento, e o recebimento do valor do serviço agendado.

REFERÊNCIAS

AKITA, Fabio. **Repensando a web com Rails**. Rio de Janeiro: Brasport, 2006.

ANDROID Architecture. Disponivel em: <http://elinux.org/Android_Architecture>. Acesso em: 10 de Abril de 2014.

Codeship. Disponivel em: <http://codeship.io>. Acesso em: 02 de maio de 2014.

BECK, K. **Programação Extrema Explicada**. Bookman, 1999.

Bibucket. Disponivel em: <http://bitbucket.org>. Acesso em: 01 de maio de 2014.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML – Guia do Usuário**. 1. Ed. Rio de Janeiro: Campus, 2000.

BOYNTON, A. C. **Archieving dynamic stability through information technology – Carlifornia Management Review**. Berkeley, winter, 1993.

DALFOVO, Oscar (Org.). **Sistemas de informação**: estudos e casos. Blumenau: Acadêmica, 2004.

DEVELOPER Android. Disponível em: http://developer.android.com/. Acesso em: 10 de Abril de 2014.

Especificacao de requisitos. Disponível em: <http://www.macoratti.net/07/12/net_fer.htm>. Acesso em: 14 de maio de 2014.

FLOWER. Martins. Continuous integration. Disponível em <http://www.martinfowler.com/articles/continuousIntegration.html#BenefitsOfContinuousIntegration>

Git. Disponivel em: <http://git-scm.com>. Acesso em: 30 de abril de 2014.

GUEDES, G.T.A. **UML: Uma Abordagem Prática**. 3ª Ed. Rio de Janeiro: Novatec, 2008.

Heroku. Disponivel em: <https://heroku.com>. Acesso em: 02 de maio de 2014.

HEUSER, Carlos Alberto**. Projeto de Banco de Dados**. 5ª Edição. Editora Sagra Luzzatto, 2004.

JAQUELINE, Sordi, Cresce o número de sites para agendamento online de consultas, **ZEROHORA**, Porto Alegre, Mar. 2013. Disponível em: <http://zerohora.clicrbs.com.br/rs/vida-e-estilo/bem-estar/noticia/2013/03/cresce-o-numero-de-sites-para-agendamento-online-de-consultas-4078202.html>. Acesso em 16/01/2014.

LECHETA, R. Ricardo. **Google Android: aprenda a criar aplicações para dispositivos móveis com o Android**. 2. ed. São Paulo: Novatec, 2012

Manual de Referência do PostgreSQL 9.3. Disponível em: <http://www.postgresql.org.br/sobre>. Acesso em: 14 de maio de 2014.

MASON, M. Pragmatic Version Control: Using Subversion (The Pragmatic Starter Kit Series). Pragmatic Bookshelf, 2006.

Matsumoto. Y. **Ruby in a Nutshell**. O'Reilly Media, 2001.

Monteiro, João B**. Google Android:** crie aplicações para celulares e tablets: 1 ed. São Paulo: Casa do código, 2012**.**

MULLINS, L. J. **Gestão da hospitalidade e comportamento organizacional**. 4. Ed. Porto Alegre: Bookman, 2004.

PRESSMAN, R. **Engenharia de software**. McGraw-Hill Interamericana do Brasil.

OHNSON, Jim. ROI, It's your job. Published Keynote Third International Conference on Extreme Programming. Alghero, Itália, 2002.

PIMENTEL, Alex. **Estratégias Essenciais de marketing – Como fidelizar e encantar seu cliente por meio do uso do CRM e do Marketing**. São Paulo: Digerat Books, 2008.

Ruby. Disponível em**:** <https://ruby-langa.org/pt/about/#fn1>. Acesso em: 10 de abril de 2014.

Ruby on rails security. Disponível em: <http://guides.rubyonrails.org/security.html>. Acesso em: 21 de abril de 2014.

SILVA, Clóvis L. M.; BARBOSA, Solange de Lima. **Estratégia, fatores de competitividade e contexto de referência das organizações**: uma análise arquetípica. Revista de Administração Contemporânea, Curitiba, v.6, n.3, set./dez. 2002. Disponível em: <http://www.scielo.br/scielo.php?pid=S1415-65552002000300002&script=sci_arttext&tlng=en>. Acesso em: 16/01/2014.

SILVA, R. P. e. **UML 2 em Modelagem Orientada a Objetos**.Florianópolis:Visual Books, 2007.

SOMMERVILLE, I. Engenharia de software. São Paulo. Addison-Wesley, 2008.

Souza, Lucas**. Ruby:** aprenda a programar linguagem mais divertida: 1 ed. São Paulo: Casa do código, 2013**.**

TANENBAUM, A. S. Redes de Computadores. 3. ed. Rio de Janeiro: Campus, 1997.

TEOREY, Toby J. Projeto e modelagem de Banco de dados. Elsevier, 2007.

ZACKER, C; DOYLE, P. **Redes de computadores**: Configuração, Manutenção e Expansão. São Paulo: MAKRON Books, 2000.

ANEXO A

Caso de Uso	UC02 | Criar a conta.
Finalidade	Criar uma nova conta no sistema.
Atores	Usuário.
Pré-condições	
Pós-condições	Encaminhar o e-mail de confirmação de cadastro.
Sequencia de execução	1. O usuário deverá acessar a página de cadastro de conta; 2. Será exibida a página com os dados iniciais de cadastro; 3. Os dados preenchidos devem ser validados; 1. Após o preenchimento, o formulário deverá ser submetido por meio do botão de cadastrar; 2. O e-mail de confirmação de cadastro deve ser encaminhado; 4. Deverá exibir a página com o tipo de perfil da conta; 5. Será exibida a página de configuração de conta com as opções de acordo com o perfil selecionado.

Caso de Uso	UC03 | Configurar uma empresa.
Finalidade	Inserir os dados da empresa na base de dados.
Atores	Usuário.
Pré-condições	A conta do usuário deve estar cadastrada no sistema.
Pós-condições	Os dados inseridos deverão ser demonstrados na página da empresa configurada.
Sequencia de execução	1. O usuário deverá acessar a área de dados da empresa; 2. Serão exibidos os campos com os dados da empresa; 3. Os dados preenchidos devem ser validados; 4. Após o preenchimento, o formulário deverá ser submetido por meio do botão salvar; 5. Ao selecionar o botão salvar, os dados inseridos serão persistidos, caso esteja consistente.

Caso de Uso	UC04	Convidar funcionários.
Finalidade	Inserir os funcionários da empresa no sistema.	
Atores	Usuário.	
Pré-condições	A conta do usuário deve estar cadastrada no sistema.	
Pós-condições	Disponibilizar visualização de funcionários no agendamento de serviços; Caso o funcionário foi inserido por meio de associação da conta do agendo.vc, deverá ser encaminhado um e-mail de confirmação para o colaborador.	
Sequencia de execução	1. O usuário deverá acessar a área de funcionários da empresa; 2. Serão exibidos os funcionários cadastrados pela empresa; 3. Ao selecionar a opção adicionar funcionário, deverá ser exibida a página com os dados do funcionário, ou para associar a conta do funcionário; 4. Se os dados inseridos estiverem corretos, os dados serão inseridos na base de dados; 5. Caso os dados inseridos não estiverem corretos, será exibida a mensagem de aviso e as informações não serão persistidas na base de dados.	

Caso de Uso	UC05	Cadastrar o serviço.
Finalidade	Cadastrar os serviços oferecidos pela empresa no sistema.	
Atores	Administrador.	
Pré-condições		
Pós-condições		
Sequencia de execução	1. O usuário deverá acessar a área de serviços oferecidos; 2. Serão exibidos os serviços cadastrados pela empresa; 3. Ao selecionar o botão adicionar serviço, deverá ser carregada a página com os serviços disponíveis pelo sistema para ser incluído; 4. Após selecionar os serviços preenchidos, o usuário deve submeter os serviços por botão adicionar serviços; 5. Se os dados inseridos estiverem corretos, os dados serão inseridos na base de dados; 6. Caso os dados inseridos não estiverem corretos, será exibida a mensagem de aviso e as informações não serão persistidas na base de dados.	

Caso de Uso	UC06	Configurar o serviço.
Finalidade	Configurar as informações do serviço cadastrado.	
Atores	Usuário.	
Pré-condições	O serviço deve estar cadastrado no sistema.	
Pós-condições	Caso o serviço foi inativado, não deverá ser mais exibido na página de agendamento.	
Sequencia de execução	1. O usuário deve selecionar o serviço cadastrado; 2. Ao clicar no botão configurar, deverá exibir a página de configuração com as opções do serviço; 3. Os dados inseridos devem ser validados; 4. Se os dados inseridos estiverem corretos, os dados serão inseridos na base de dados; 5. Caso os dados inseridos não estiverem corretos, será exibida a mensagem de aviso e as informações não serão persistidas na base de dados.	

Caso de Uso	UC07	Realizar o agendamento.
Finalidade	Reservar um horário na agenda cadastrada.	
Atores	Usuário.	
Pré-condições	Usuário deverá possuir a conta no sistema, e a empresa que fornece o agendamento deve possuir a agenda cadastrada na sua conta.	
Pós-condições	O horário selecionado pelo usuário deverá ficar reservado pelo usuário.	
Sequencia de execução	1. O usuário deve realizar a autenticação no sistema; 2. Selecionar o serviço da empresa que deseja marcar o agendamento; 3. Selecionar um horário na agenda disponibilizada pela empresa; 4. Com o serviço selecionado, o usuário deverá encaminhar a solicitação de reserva por meio de botão de agendamento; 5. Os dados inseridos deverão ser validados; 6. A solicitação deverá ser encaminhada para a conta da empresa e persistida na base de dados.	

Caso de Uso	UC 08 | Consultar as empresas que fornecem o serviço.
Finalidade	Visualizar as empresas que fornecem o serviço selecionado.
Atores	Administrador.
Pré-condições	
Pós-condições	
Sequencia de execução	1. O usuário deve selecionar o serviço desejado na página inicial; 2. Ao selecionar o serviço, deverá listar todas as empresas que fornece o serviço, junto com suas características; 3. Ao selecionar a empresa, deverá exibir a pagina de agendamento com as informações do serviço e a empresa.

Caso de Uso	UC09 | Inativar o colaborador.
Finalidade	Inativar o colaborador do sistema.
Atores	Administrador.
Pré-condições	O colaborador deverá estar cadastrado no sistema. O funcionário deve estar ativo.
Pós-condições	O funcionário não poderá ser listado no quadro de funcionário da empresa que recebeu a inativação.
Sequencia de execução	1. O usuário do sistema deverá realizar a autenticação no sistema; 2. Deverá acessar a página de gerenciamento de funcionários. 3. A página deve listar todos os colaboradores da empresa; 4. Para a inativação do usuário, o mesmo deverá ser selecionado na lista de colaborador da empresa, e clicar na opção inativar; 5. Para exclusão de um cliente, deverá selecioná-lo na lista de clientes cadastrados; 6. Ao acionar o botão de exclusão, o sistema deverá exibir uma janela de confirmação de exclusão; 7. Caso o administrador opte por excluir, os dados do cliente deverão ser marcados como inativo da base de dados do sistema; 8. Se o administrador selecionar a opção de cancelamento de exclusão, o sistema deverá redirecioná-lo para a página anterior e nenhum dado do cliente deverá ser alterado ou removido da base de dados.