UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE INFORMÁTICA

CURSO DE CIÊNCIA DA COMPUTAÇÃO

GUILHERME MARTINI

Gerenciador de períodos de utilização de baias

Trabalho de graduação

Prof. Dr. Leandro Krug Wives
Orientador

Porto Alegre, novembro de 2013

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
AES	Advanced Encryption Standard
MVC	Model View Controller
CRUD	Create Read Update Delete
HTML	Hypertext Markup Language
IP	Internet Protocol
CSS	Cascading Style Sheets
AJAX	Asynchronous JavaScript and XML
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
UFRGS	Universidade Federal do Rio Grande do Sul
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	eXtensible Markup Language

LISTA DE FIGURAS

RESUMO

Este trabalho tem como objetivo o desenvolvimento de um sistema Web para gerenciar salas, baias e dispositivos, além de fazer a alocação de pessoas nas baias por certo período de tempo. O sistema também permite a criação de perfis de usuários, limitando ou expandido a quantidade de funcionalidades do sistema que os usuários associados ao perfil em questão poderão acessar. Neste trabalho, é abordado o processo de coleta de requisitos do sistema, as tecnologias utilizadas para o desenvolvimento do mesmo e a metodologia de desenvolvimento utilizada. Além disso, são abordadas as entidades do sistema, o fluxo de telas e o processo de implantação do sistema em produção.

Palavras-Chave: Gerenciador de recursos, alocação de bolsistas, metodologias ágeis, sistema Web.

Manager system of periods workstation use

ABSTRACT

This paper aims at the development of a Web system to manage room, tables, devices and allocation of people on those tables for a period of time. The system also allows the creation of user profiles, limiting or expanding the amount of system features that users associated with this profile can access. This paper disserts about the process of system requirements acquisition, the technologies used for its development and development methodology used. Also, are covered entities of the system and its use flow. Deploy system in production and a survey with the system users is also discussed.

Keywords: resource manager, allocation of trainees, agile, Web system.

1 INTRODUÇÃO

O grupo de pesquisa em sistemas de informação tem algumas salas e uma série de baias disponíveis e necessitam organizar estes recursos de maneira que os mesmos sejam aproveitados da melhor forma possível. É necessário evitar que esses recursos não sejam utilizados em algum momento. A fim de auxiliar a organização do gerenciamento desses recursos, esse trabalho é proposto.

Este trabalho consiste do desenvolvimento de um sistema Web que permitirá aos professores do Instituto de Informática da UFRGS gerenciar os períodos de utilização de baias pelos bolsistas, gerando alerta de fim de utilização de espaços e permitindo a geração de relatórios. O sistema foi planejado para resolver o problema de alocação de recursos do Grupo de Pesquisa em Sistemas de Informação do Instituto de Informática da UFRGS, mas a solução será genérica, ou seja, poderá ser utilizada por qualquer grupo que necessite alocar pessoas e recursos em salas.

É objetivo deste trabalho, também, fazer uso de metodologia ágil durante o desenvolvimento do projeto e a utilização de ferramentas que auxiliam a gerência das tarefas do projeto e o acompanhamento do desenvolvimento por parte do cliente. Além disso, far-se-á uso de ferramentas para o controle de versão de código e cadastro de bugs do projeto.

Nos capítulos a seguir será apresentada a forma de coleta de requisitos, as tecnologias utilizadas para o desenvolvimento do projeto e as ferramentas utilizadas. Além do modelo de entidades e fluxo de telas do sistema desenvolvido e o processo de implantação do sistema em produção. Finalmente, serão apresentadas as conclusões e trabalhos futuros.

2 DESCRIÇÃO DO PROBLEMA

Neste capitulo será apresentado o problema que há com o processo atual de gerenciamento de salas, baias, alunos e bolsistas. Além disso, será apresentado o método utilizado para a identificação dos requisitos do sistema e a maneira como esses foram usados durante o processo de desenvolvimento.

2.1 Processo atual

Os professores do Instituto de Informática da UFRGS possuem certo número de salas, baias, computadores e vagas de bolsa de pesquisa. Esses recursos precisam ser cadastrados e gerenciados de maneira prática e eficiente. Em especial, no Grupo de Pesquisa em Sistemas de Informação, esses dados são, atualmente, armazenados em uma planilha no Google Docs. Essa planilha contém os seguintes dados: número da salsa, identificador da mesa, nome do aluno que está usando a mesa, nome do orientador do aluno, nível do aluno (graduação, mestrado...), previsão de término do uso da mesa, descrição do computador utilizado pelo aluno, número do patrimônio desse computador e um campo para observação.

Os principais problemas enfrentados pelo método atual são: manter o histórico de uso das salas, visualizar términos de período de utilização, saber o horário em que os alunos estão utilizando a sala, entre outros. O desenvolvimento deste sistema visa melhorar o processo de gerenciamento desses recursos, tornando mais prática e eficaz a identificação de disponibilidade e uso dos mesmos.

2.2 Levantamento de requisitos

A identificação de requisitos do sistema foi feita através de diversas reuniões com os possíveis usuários. Como se optou por seguir uma metodologia ágil para a construção do sistema (ver capitulo 3.1), os requisitos foram incrementais, identificados em reuniões ao longo do processo de desenvolvimento, ou seja, em cada reunião novos requisitos eram identificados e outros modificados para melhor satisfazer as necessidades dos usuários.

As reuniões de levantamento de requisitos e de acompanhamento do desenvolvimento ocorreram a cada duas semanas. Com isso foi possível obter os requisitos de maneira detalhada conforme o sistema ia sendo construído.

Consequentemente, foi possível reduzir a ocorrência de retrabalho causado por um requisito mal identificado.

Os requisitos foram recolhidos de maneira bastante informal, através de relatos dos usuários sobre as suas necessidades e perguntas para tentar identificar maiores detalhes. Os requisitos levantados foram organizados na forma de *user stories* e usados como base para o desenvolvimento do trabalho.

2.3 User stories

User stories (histórias de usuários) são anotações feitas que descrevem funcionalidades do sistema que serão valiosas para um usuário. As histórias devem ser breves, focar numa funcionalidade, no seu objetivo e no papel do usuário que a deseja. É importante que uma história seja independente das demais - e negociável, ou seja, possa haver mudanças; além disso, deve ser estimável, pequena e testável (COHN, 2004).

As histórias de usuário identificadas no contexto deste trabalho foram organizadas de forma textual e serviram para orientar o desenvolvimento e priorizar as atividades mais importantes. Cada *user story* foi avaliada, juntamente com um usuário, para definir a importância que a mesma tem no sistema. Dessa forma, a priorização das histórias foi feita da forma mais realística possível. Abaixo segue a lista de histórias levantadas durante o desenvolvimento do sistema.

- 1: Eu como usuário quero logar no sistema para acessar funcionalidades do sistema associadas ao meu usuário.
- 2: Eu como usuário quero adicionar usuários para permitir que novas pessoas tenham acesso ao sistema.
- 3: Eu como usuário quero editar usuários para alterar o perfil e a pessoa ao qual o usuário está associado.
- 4: Eu como usuário quero remover usuários para impedir o acesso de pessoas que não fazem mais parte do sistema.
- 5: Eu como usuário quero adicionar perfis para poder criar novos perfis de acesso ao sistema escolhendo as funcionalidades do mesmo.
- 6: Eu como usuário quero editar perfis para poder alterar funcionalidades ou o nome do perfil.
- 7: Eu como usuário quero remover perfis para impedir a utilização de perfis que não são mais válidos.
- 8: Eu como usuário quero adicionar salas para poder cadastrar as salas onde serão alocados os recursos.
- 9: Eu como usuário quero indicar o usuário responsável por manter os dados de uma sala atualizados para delegar a responsabilidade de gerência da sala para o usuário indicado.
- 10: Eu como usuário quero editar salas para alterar os dados de cadastro da mesma.
- 11: Eu como usuário quero remover salas para apagar do sistema salas que não serão mais utilizadas.

- 12: Eu como usuário quero adicionar baias para cadastrar baias que estão nas salas cadastradas.
- 13: Eu como usuário quero associar uma baia a uma pessoa com papel de orientador para identificar o professor que é responsável pela baia.
- 14: Eu como usuário quero vincular a baia a uma sala identificando a sua posição física dentro da sala para conseguir identificar a posição das mesas na sala.
- 15: Eu como usuário quero editar baias para alterar os dados de cadastro da baia.
- 16: Eu como usuário quero apagar baias para excluir do sistema baias que não existem mais.
- 17: Eu como usuário quero adicionar bolsistas para cadastrar os bolsistas no sistema indicando o intervalo de datas que ele usará uma baia.
- 18: Eu como usuário quero associar um bolsista a uma pessoa e a uma baia para identificar quem é o bolsista e onde ele está trabalhando.
- 19: Eu como usuário quero editar um bolsista para alterar os dados cadastrais do mesmo.
- 20: Eu como usuário quero remover bolsistas para excluir do sistema os bolsistas que não são mais de interesse para o sistema.
- 21: Eu como usuário quero adicionar períodos de trabalho dos bolsistas para identificar os períodos do dia em que o usuário estará usando a baia.
- 22: Eu como usuário quero editar períodos de trabalho dos bolsistas para alterar dados de cadastro dos períodos.
- 23: Eu como usuário quero remover períodos de trabalho dos bolsistas para excluir períodos de trabalho incorretos do sistema.
- 24: Eu como usuário quero adicionar pessoas para cadastrar novas pessoas ao sistema.
- 25: Eu como usuário quero associar uma instituição de ensino a pessoa para identificar a instituição de origem de pessoas vindas de outras universidades.
- 26: Eu como usuário quero editar pessoas para alterar os dados cadastrados da mesma.
- 27: Eu como usuário quero remover pessoas para excluir o cadastro de pessoas que não são mais relevantes para o sistema.
- 28: Eu como usuário quero adicionar instituições para cadastrar no sistema as instituições de interesse.
- 29: Eu como usuário quero editar instituições para alterar os dados de instituições já cadastradas.
- 30: Eu como usuário quero remover instituições para excluir instituições incorretas do sistema.
- 31: Eu como usuário quero adicionar papéis para cadastrar novos papéis no sistema conforme a necessidade.
- 32: Eu como usuário quero editar papéis para alterar dados de cadastro dos papéis.
- 33: Eu como usuário quero remover papéis para excluir papéis incorretos do sistema.
- 34: Eu como usuário quero visualizar uma tela inicial com mensagem de boas vindas para ter uma tela principal do sistema.
- 35: Eu como usuário quero visualizar uma tela inicial com alertas de períodos de uso de baias que estão expirando para identificar com facilidade quais baias ficaram livres em um curto espaço de tempo.

- 36: Eu como usuário quero visualizar relatório de ocupação de das salas para identificar quais orientadores tem baias em quais salas e identificar salas com baias desocupadas.
- 37: Eu como usuário quero poder filtrar dados do relatório de ocupação de salas para refinar os dados mostrados pelo relatório.
- 38: Eu como usuário quero exportar o relatório de ocupação para Excel para ter os dados de ocupação sem precisar acessar o sistema.
- 39: Eu como usuário quero visualizar relatório por período para identificar quem está usando cada baia por quais períodos do dia e da semana.
- 40: Eu como usuário quero poder filtrar dados do relatório por período para refinar os dados mostrados pelo relatório.
- 41: Eu como usuário quero exportar o relatório por período para Excel para ter os dados sem precisar acessar o sistema.

3 Tecnologias e Ferramentas

Neste capitulo são apresentados detalhes sobre a metodologia de trabalho utilizada durante o desenvolvimento, as tecnologias escolhidas para este trabalho, além de serem descritas as ferramentas usadas para gerência das atividades de desenvolvimento e controle de versão do código gerado.

3.1 Metodologia

Para o desenvolvimento deste projeto foi utilizada uma metodologia de trabalho baseada no *Scrum*. O desenvolvimento de um projeto seguindo a metodologia Scrum deve ocorrer de forma incremental, ou seja, cada entrega deve ser construída a partir de uma entrega anterior adicionando novas funcionalidades. Essa forma de trabalho possibilita aos times serem mais criativos, responder a feedbacks e mudanças de forma mais rápida e eficiente e que o resultado final do projeto seja exatamente e apenas aquilo que é necessário (SCHWABER e SUTHERLAND, 2013).

As principais características da metodologia Scrum utilizadas no desenvolvimento deste trabalho foram: o uso de iterações, conhecidas como *sprints*, com duração de duas semanas, disponibilização de uma nova versão ao final de cada iteração e acompanhamento do cliente no processo de desenvolvimento. Com isso, foi possível identificar novas *user stories* e mudanças nas já existentes com maior facilidade e o mais cedo possível.

O desenvolvimento deste projeto foi feito por apenas uma pessoa, isso dificulta a adoção de algumas características da metodologia Scrum. Por exemplo, não faria sentido realizar reuniões diárias, as quais têm como objetivo acompanhar o que foi feito no dia anterior e discutir o que será feito no dia atual. Em relação aos papéis que a metodologia prevê, há o *Scrum master* (responsável por garantir que a metodologia seja entendida e aplicada), a equipe de desenvolvimento e o *Product Owner* (responsável por maximizar o valor do produto e o trabalho da equipe de desenvolvimento). De certa forma, mesmo com uma equipe formada por uma pessoa, esses papéis foram exercidos, mas de maneira implícita.

3.2 Python

A linguagem de programação escolhida para fazer o desenvolvimento foi Python. Python é uma linguagem de programação dinâmica que é usada em uma grande variedade de domínios de aplicações. Algumas das principais características da linguagem são: sintaxe muito clara e legível, orientação a objetos intuitiva, tipos de dados dinâmicos de nível elevado, extensões e módulos facilmente escritos em C, C++, Jython (adaptação de Python para Java) e IronPython (adaptação de Python para .NET) e pode ser embutido em aplicações como interface de *script*, além de ter uma grande quantidade de bibliotecas padrão e módulos de terceiros disponíveis (ABOUT PYTHON, 2013).

Um recurso muito interessante da linguagem é o *shell* interativo chamado IPython, ele é um programa que permite ao desenvolvedor digitar linhas de código e ver o resultado das mesmas imediatamente. Esse recurso é muito útil para tirar dúvida sobre a sintaxe de determinado comando e, também, para testar pequenos trechos de código.

O código escrito em Python fica muito legível, normalmente, por causa da endentação, que tem a função indicar início e fim de blocos de código. Comumente, em outras linguagens a endentação é opcional e o marcador de inicio e fim de bloco são símbolos específicos. Por exemplo, abre chaves, "{", para início de bloco e fecha chaves, "}", para fim. Os operadores lógicos, também, fazem a sintaxe ser bastante legível, já que eles são representados textualmente em inglês. Por exemplo, os operadores "E" e "OU" são representados por "and" e "or", respectivamente. Além disso, a documentação abundante e a facilidade de desenvolvimento rápido fizeram com que Python fosse a linguagem escolhida para o desenvolvimento deste trabalho.

3.3 Framework Django

Para o desenvolvimento do projeto foi utilizado o Framework Django. "Django é um framework web Python que estimula desenvolvimento rápido e legível, com design pragmático." (Meet Django, 2013, tradução nossa). As principais características do framework são: mapeamento objeto-relacional, interface administrativa automática, design de URL elegante, sistema de templates, sistema de cache e internacionalização.

Os *templates* do Django são arquivos de texto que unem o código HTML da página com códigos em Python. O código em Python é inserido através de dois elementos: variáveis e *tags*. As variáveis têm o seu valor avaliado e mostrado na tela no momento que o *template* é processado, e as *tags* são os comandos de controle de fluxo, como *if* e *for* (The Django Template Language, 2013). Com esses recursos é possível criar *templates* genéricos para serem usados em várias telas que mostram os mesmos tipos de dados em uma única estrutura.

Todo o projeto foi construído com apenas cinco *templates*, um para a tela de login, um para o layout básico do sistema, um para as telas de listagem, um para as telas de edição e um para as telas de relatório. Os *templates* se mostraram muito bons para tornar o desenvolvimento mais ágil, uma vez que não é necessário criar arquivos HTML

para cada página desenvolvida, bastando criar a estrutura de dados que o *template* espera receber e carregá-lo. O processo de criação de um *template* é, no entanto, mais trabalhoso do que o de uma tela HTML normal, pois é necessário criar uma estrutura de dados que seja genérica para poder ser usada em todas as telas que tenham a mesma estrutura.

O Django também tem a definição de *models*, que são classes que representam os dados que se deseja armazenar. No desenvolvimento do projeto, cada classe dos *models* foi mapeada para uma tabela no banco de dados. Essa característica é nativa do Django, e, com isso, o próprio framework mapeia essas entidades para suas respectivas tabelas. Para fazer esse mapeamento, é necessário informar o tipo de dado de cada atributo da classe, isso é feito através da classe models do Django que tem suporte a tipos básicos como VARCHAR e até a definição de relacionamentos de todos os tipos, como *Many-To-Many*. É possível definir algumas características de cada atributo como o tamanho máximo ou um valor *default*. Basicamente o framework permite o desenvolvedor definir as mesmas características que seriam possíveis ao criar a tabela diretamente no banco de dados.

Para acessar os dados, o Django fornece uma API que permite criar, recuperar, atualizar e remover entradas no banco. A operação para selecionar os elementos de uma classe é feita através do atributo "objects", métodos para apagar, criar e atualizar os dados são herdados da classe Manager do models do Django.

No desenvolvimento do projeto decidiu-se que as remoções de dados do banco seriam lógicas e não físicas, de maneira a garantir que possam ser recuperados. Com isso, todas as classes têm uma coluna do tipo "bool" que pode receber os valores verdadeiro ou falso, chamada "is_removed". Para facilitar a consulta, limitando-a a dados que não foram removidos, todas as entidades herdam de uma classe que sobrescreve a base de consulta, excluindo os objetos que têm a coluna "is_removed" com o valor verdadeiro.

```
class GenericManager(models.Manager):
    def get_query_set(self):
        return super(GenericManager, self).get_query_set().exclude(is_removed=True)
```

Figura 3.1: Superclasse de todas as outras classes do sistema.

Outra funcionalidade do framework que auxilia a tornar o desenvolvimento mais limpo e rápido são os *forms*. *Forms* são classes que definem os campos que serão mostrados na tela, segundo Working with Forms (2013) eles permitem:

> [...] mostrar um formulário HTML com campos gerados automaticamente, verificar os dados submetidos com um conjunto de regras de validação, mostrar o formulário novamente em caso de erros de validação e convertem dados submetidos para tipos de dados relevantes para o Python. (tradução nossa)

Todos os formulários do projeto foram criados usando os formulários do Django. Apenas em duas situações não foram usadas todas as funcionalidades dos *forms*: são elas a tela de login e a de edição de usuários. Em ambos os casos, essa escolha ocorreu devido a razões de segurança. Nesses casos, a funcionalidade usada foi apenas a de mostrar os campos na tela. A motivação para isso foi a criptografia da senha informada pelo usuário, feita em JavaScript. Assim, a ação do *form* foi interrompida pelo JavaScript e os dados de senha informados foram criptografados utilizando o algoritmo AES. O post do *form* foi feito utilizando AJAX, mais detalhes dessas operações serão descritos na seção 3.6.

3.4 PostgreSQL

O banco de dados escolhido para o desenvolvimento do projeto foi o PostgreSQL 9.1. A escolha se deu por ser um sistema de banco de dados objeto-relacional poderoso e gratuito, além de ser amplamente usado no mundo todo e ter um desenvolvimento ativo há mais de 15 anos e uma arquitetura comprovada que ganhou forte reputação por confiabilidade, integridade dos dados e correção (PostgreSQL About, 2013).

A criação das tabelas do banco foi feita através do Django, mas foi criado um arquivo para criar os dados básicos do sistema para o primeiro uso, tais como as funcionalidades que poderiam ser associadas a perfis, os papéis básicos e usuário padrão. Também foi necessário criar um arquivo para atualizar o sistema em uma das entregas em produção, pois nesta entrega foi acrescido o relacionamento entre sala e pessoa para atender a *user story* 13.

3.5 Apache

Este trabalho foi desenvolvido para o ambiente Web, o que permite que os usuários acessem o sistema de qualquer lugar sem ter que instalar nada em seus dispositivos, com isso também é garantido que o ambiente é multiplataforma, ou seja, pode ser acessado a partir de qualquer sistema operacional. Para colocar o projeto em produção, foi necessário configurar um servidor Web, um dos servidores Web mais utilizados é o Apache, e por isso ele foi utilizado neste projeto.

Na documentação do Django é sugerido que a implantação em produção seja feita com Apache e mod_python. Mod_python é um *plug-in* do Apache que embute o interpretador Python para dentro do servidor e carrega código Python em memória quando o servidor é iniciado (HOLOVATY e KAPLAN-MOSS, 2009). Por carregar o código em memória esse *plug-in* aumenta muito o desempenho do servidor apache, tornando a aplicação mais rápida.

3.6 JavaScript

JavaScript é uma linguagem usada na construção de páginas Web que permite fazer alterações no conteúdo exibido no navegador do cliente, ou seja, não é necessário

fazer uma requisição ao servidor para fazer o processamento. Além disso, é uma linguagem interpretada com capacidade de orientação a objetos.

> JavaScript é uma línguagem de programação interpretada, de alto nível, dinâmica e sem tipos que lida bem com programação orientada a objetos e programação funcional. (FLANAGAN, 2011, pag 1, tradução nossa).

No projeto, o JavaScript foi usado para implementar criptografia, dar funcionalidade a alguns componentes de tela, fazer requisições para o servidor e manipular dados da tela. Para isso foram usadas algumas bibliotecas de JavaScript, são elas: JQuery e SlowAES.

JQuery é uma biblioteca de JavaScript amplamente usada no desenvolvimento para Web. Ela reduz quantidade de código JavaScript que precisa ser feito e provê garantia de qualidade, além disso ela faz com que o cpódigo JavaScript fique mais intuitivo e de fácil compreensão. (YORK, Richard, 2009) As principais funcionalidades trazidas pela biblioteca são manipulação de documentos HTML, manipulação de eventos, animação e requisições Web por Ajax. Além disso, sua API é fácil de usar e funciona na maioria dos navegadores oferecendo uma combinação de versatilidade e extensibilidade. No projeto, o principal uso do JQuery foi na manipulação de componentes da tela, requisições Ajax e ele também é uma dependência do framework Twitter Bootstrap.

Ajax é uma sigla que significa *Assynchronous JavaScript and XML*, ou seja, JavaScript e XML assíncronos. Ajax é usado para trocar dados entre uma página e o servidor Web sem precisar recarregar toda a página. (AJAX TUTORIAL - 2013). A biblioteca JQuery tem uma API que permite fazer requisições Ajax de forma muito simples. Esse recurso foi usado nas telas de *login* e de edição de usuários para permitir a criptografia da senha do usuário.

Twitter Bootstrap é um *framework* de *front-end* que foi utilizado para definir o padrão de *layout* do sistema. O *framework* é composto por documentos JavaScript e arquivos CSS e será explicado na seção 3.7.

Além disso, também foi utilizada a biblioteca SlowAES para criptografar a senha dos usuários do sistema para evitar que a senha fosse transferida em claro entre o cliente e o servidor. Outra opção seria usar o protocolo HTTPS, mas essa opção não foi seguida pelo fato de envolver a emissão de um certificado por parte de um certificador oficial, o que também exigiria certo custo financeiro. Alternativamente, seria possível usar um certificado não oficial, mas, para evitar a necessidade do usuário ter de confirmar uma exceção de segurança no navegador, foi feita a opção de usar um algoritmo de criptografia em JavaScript.

3.7 CSS

O código CSS é usado para colocar o estilo em páginas Web. Usar arquivos CSS para colocar os estilos dos elementos, ao invés de colocar diretamente nos mesmos, ajuda a manter o código mais organizado e legível. “A especificação de *cascading style*

sheets funciona permitindo que você especifique como o conteúdo dos elementos do seu documento deve aparecer.” (LARSEN, pag.191, 2013, tradução nossa).

No sistema desenvolvido, foi criado apenas um arquivo CSS, o base.css, que foi utilizado para criar estilos gerais para toda a aplicação. Criar apenas um arquivo CSS foi possível porque a maior parte dos estilos do sistema é proveniente do *framework* Twitter Bootstrap (ver seção seguinte) e dos seus componentes, isso permite maior robustez para a aplicação uma vez que esse framework é amplamente usado e tem manutenção constante.

3.8 Twitter Bootstrap

Em grande parte das telas do sistema proposto foi usado o padrão visual disponibilizado pelo framework Twitter Bootstrap.Twitter Bootstrap é um framework *front-end* intuitivo que permite que o desenvolvimento Web seja mais rápido e de mais fácil manutenção.

A adoção desse *framework* se deu no meio do desenvolvimento do projeto. A maior motivação para o seu uso foi melhorar o visual do sistema. Além da melhoria visual, também houve uma melhoria, no sentido que o *framework* confere suporte à maioria dos navegadores Web e, também faz com que o sistema seja responsivo a diferentes tamanhos de tela. Além disso, alterações no *design* do sistema se tornam mais rápidas e fáceis uma vez que o *framework* traz componentes já estilizados e permite a customização de detalhes visuais de forma rápida e consistente.

A migração para o uso do *framework* foi bastante simples, já que, na maioria dos casos, a estrutura do HTML não muda, apenas são adicionadas novas classes CSS aos componentes já existentes. Os principais componentes utilizadas foram “navbar”, utilizado para fazer o menu, e “dropdowns”, para fazer listas em elementos do menu e os estilos dos campos dos formulários, dos botões e das listagens.

3.9 Ferramenta de gerência de atividades

Para o desenvolvimento do projeto se percebeu a necessidade de usar alguma ferramenta para gerência das atividades planejadas para cada *Sprint*. A partir dessa necessidade de gerência também se percebeu uma necessidade por acompanhamento por parte do cliente sobre aquilo que estava sendo feito. Com isso seria possível identificar ainda mais cedo se alguma atividade não estava de acordo com as necessidades do sistema, fazendo com que mudanças fossem identificadas e ações tomadas para fazer as alterações o quanto antes, evitando retrabalho.

A ferramenta escolhida para essa atividade foi o Trello (http://www.trello.com/). O Trello é uma ferramenta de gerencia de atividades *on line* e gratuita que organiza os projetos em quadros e permite que seus usuários: visualizem as atividades que estão sendo trabalhadas, quem está fazendo a atividade e como está o andamento do trabalho proposto para o quadro. (Getting started with Trello, 2013).

Durante o desenvolvimento do sistema, cada S*print* realizada foi cadastrada no Trello como um novo quadro. Cada quadro tem três diferentes estágios para as tarefas, *To Do* (Não iniciada), *Doing* (sendo feita), *Done* (pronta), e cada estágio é representado por uma coluna no quadro, o que possibilita uma rápida visualização do andamento do projeto.

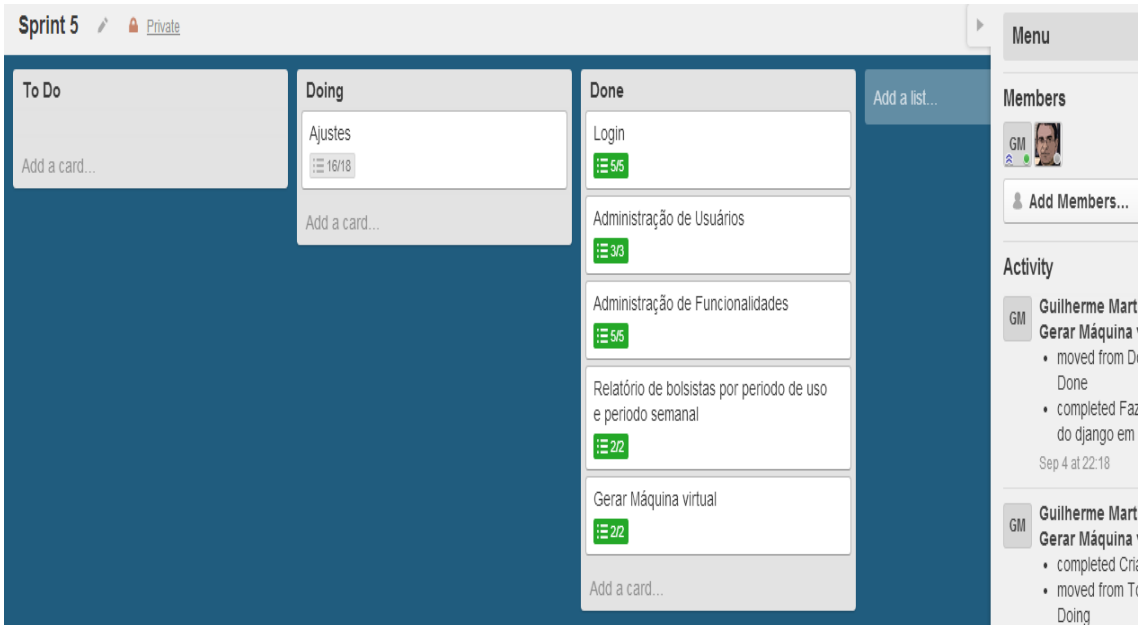


Figura 3.2: Board da Sprint 5 no Trello.

Além disso, há uma coluna na direita da tela que permite acompanhar as ações que cada usuário está fazendo no Trello e assim ver que usuário está trabalhando em qual tarefa. Dessa forma é possível identificar tarefas com possíveis impedimentos e tomar ações para resolver esses impedimentos.

As histórias de usuário foram criadas como cartões desse quadro. Em algumas situações, um cartão representou mais do que uma história de usuário, por exemplo, adicionar, editar e remover instituições formam três histórias de usuário, mas foram

representadas no Trello com apenas um cartão. Isso é possível porque dentro do cartão pode haver diferentes listas de trabalho, cada uma contendo muitas tarefas. Para este projeto cada cartão tinha três listas, uma para tarefas que ainda não tinham começado a ser feitas (*TO DO*), uma para atividades que estavam sendo feitas (*Doing*) e uma para atividades concluídas (*DONE*).

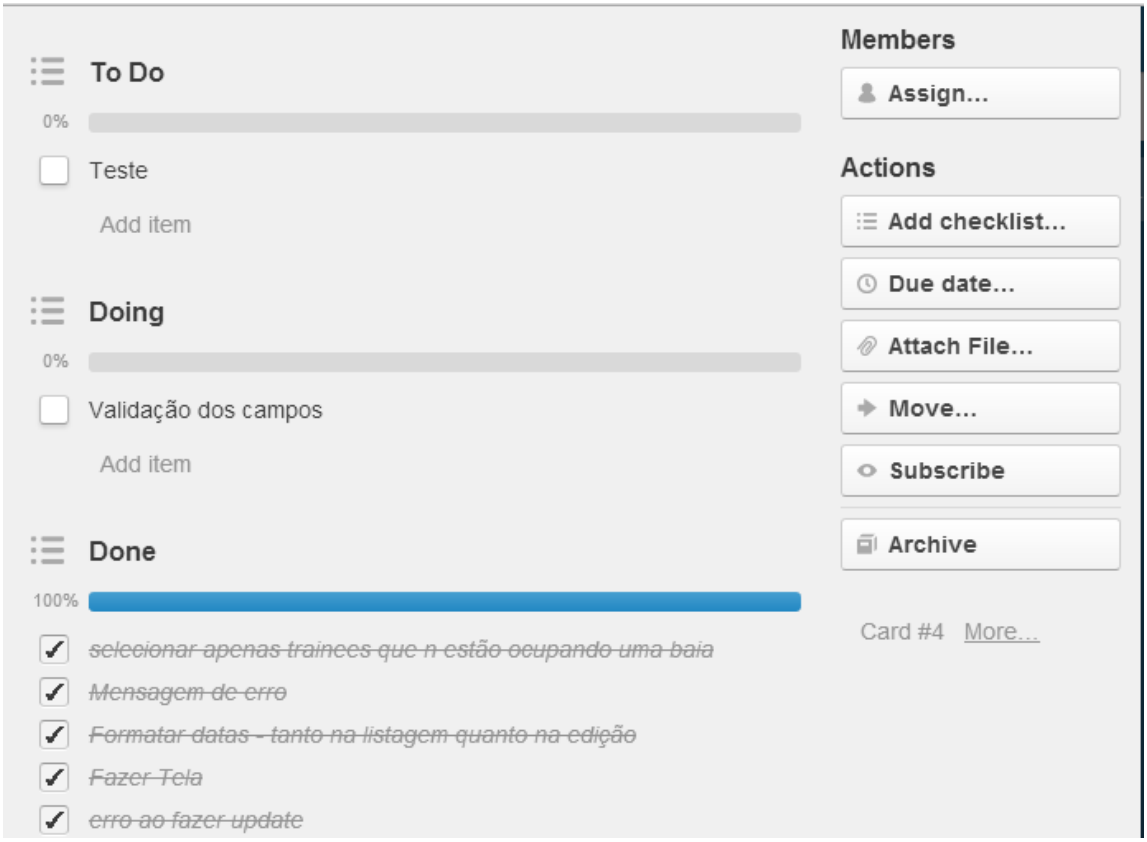


Figura 3.3: Cartão da tela de bolsistas no Trello.

3.10 Sistema de controle de versão

Durante o desenvolvimento de um software é indispensável que se tenha um sistema para fazer o controle de versão dos arquivos. Esse tipo de sistema permite manter o controle sobre mudanças que ocorreram em arquivos, quem fez essas mudanças, quando e um comentário que explique a mudança ou o porquê da mesma. Segundo Scott Chacon:

> Um Sistema de Controle de Versão permite que você: reverta arquivos para um estado anterior, reverta um projeto inteiro a um estado anterior, revise mudanças feitas durante o tempo, veja quem modificou pela última vez algo que possa estar causando problemas, quem e quando um problema foi adicionado e mais. Usar um Sistema de controle de versão também significa que se você estragar tudo ou perder arquivos, geralmente você pode recuperar facilmente. Além disso, você tem tudo isso com muito pouca sobrecarga. (2009, pág. 1, tradução nossa)

Durante o desenvolvimento deste trabalho o sistema de controle de versão usado foi o GIT. O GIT é um sistema de controle distribuído, ou seja, cada cliente tem um backup completo do repositório na sua máquina local.

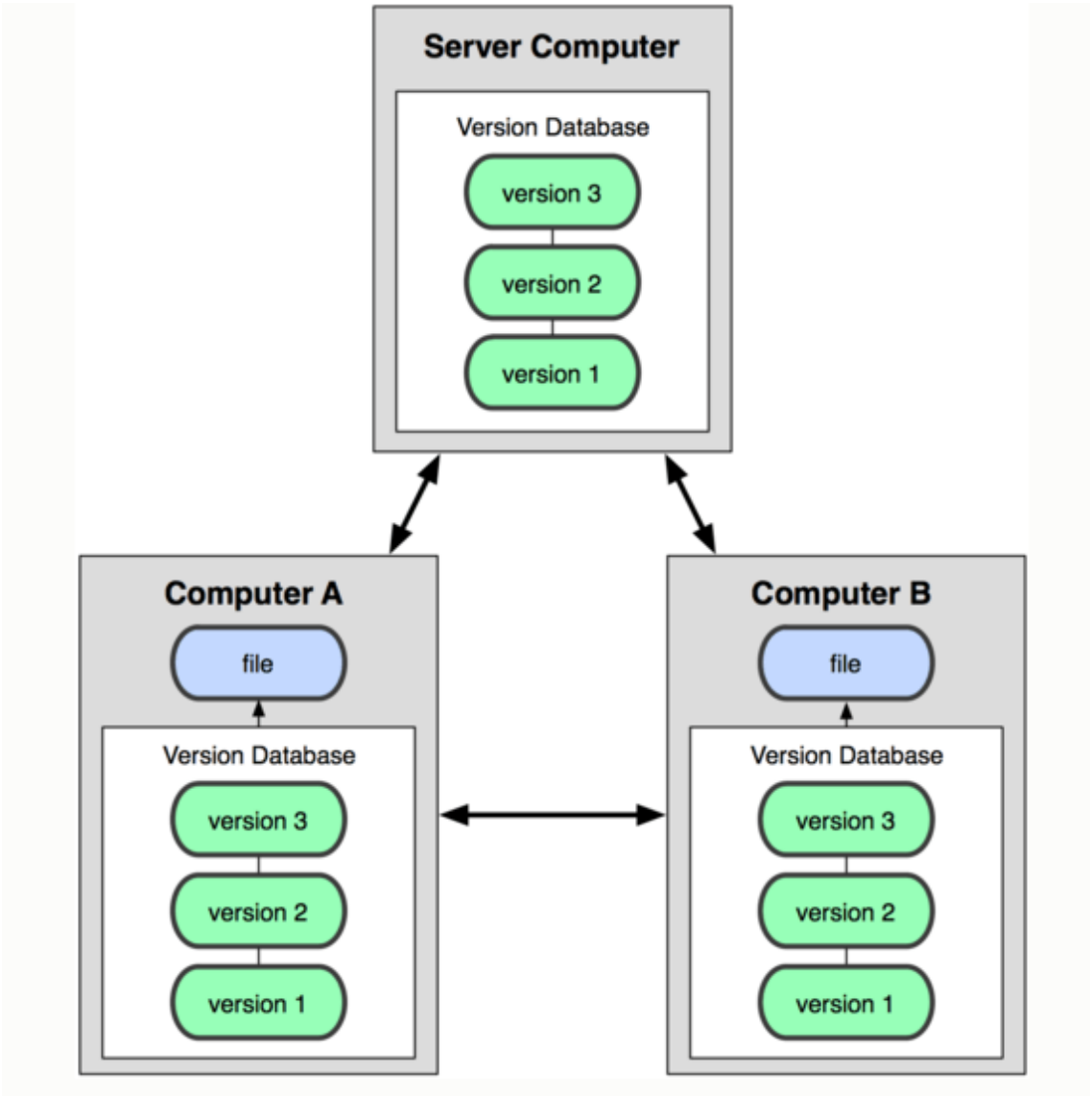


Figura 3.4: Diagrama de funcionamento de um sistema controle de versões distribuído.

Fonte: (SCOTT CHACCON, 2009, pág. 4).

Para usar o GIT é necessário instalá-lo em uma máquina que servirá como servidor, mas há algumas alternativas a necessidade de instalação desse servidor que são servidores GIT disponíveis na Internet. Há diferentes tipos de serviços disponíveis, alguns gratuitos e outros pagos, mas para projetos com poucos membros na equipe há opções gratuitas que suprem todas as necessidades dos desenvolvedores.

No desenvolvimento deste projeto, o serviço usado como servidor para o GIT foi o Github (https://github.com/), que é um sistema de hospedagem de projetos web. O site é no formato de rede social o que faz com que os desenvolvedores compartilhem projetos e troquem ideias e opiniões sobre os mesmos. O site tem diferentes modalidades de plano com diferentes recursos oferecidos. Neste projeto foi usada a

modalidade gratuita, que tem como principal limitação a impossibilidade de criar repositórios privados.

4 DESENVOLVIMENTO DO SISTEMA PROPOSTO

Neste capitulo será descrita a arquitetura do projeto e a modelagem do banco de dados. Além disso, será explicado o processo de implantação do sistema em produção e o fluxo de uso do mesmo. Também será feita uma avaliação de uso do sistema e a identificação de melhorias a serem feitas.

4.1 Arquitetura

Antes de começar o desenvolvimento de um sistema é necessário planejar como vai ser organizado o código que será desenvolvido, essa organização é chamada de arquitetura de *software* e é fundamental para facilitar a manutenção do sistema ao longo do tempo. Atualmente, o padrão de arquitetura mais usado é o MVC, sigla que significa *model, view, controller*, e, para o desenvolvimento deste projeto, esse foi o padrão de arquitetura utilizado.

O modelo MVC define que o código deve ser divido em três camadas: *model*, *view* e *controller*, elas devem ser independentes, mas devem interagir umas com as outras. Burbeck define o MCV como:

> No paradigma MVC as entradas dos usuários, a modelagem do mundo externo e o retorno visual ao usuário são, explicitamente, separados e gerenciados por três tipos de objetos, cada um especializado para suas tarefas. A view gerencia as saídas gráficas e/ou textuais da porção de tela que está alocada to a aplicação. O controller interpreta as entradas de mouse e teclado do usuário, comandando o model e/ou a view para mudar conforme apropriado. Finalmente, o model gerencia o comportamento e os dados do domínio da aplicação, responde a requisições de informação sobre seu estado (normalmente vindas da view), e responde a instruções para mudança de estado (normalmente do controller). (1987, p.1, tradução nossa)

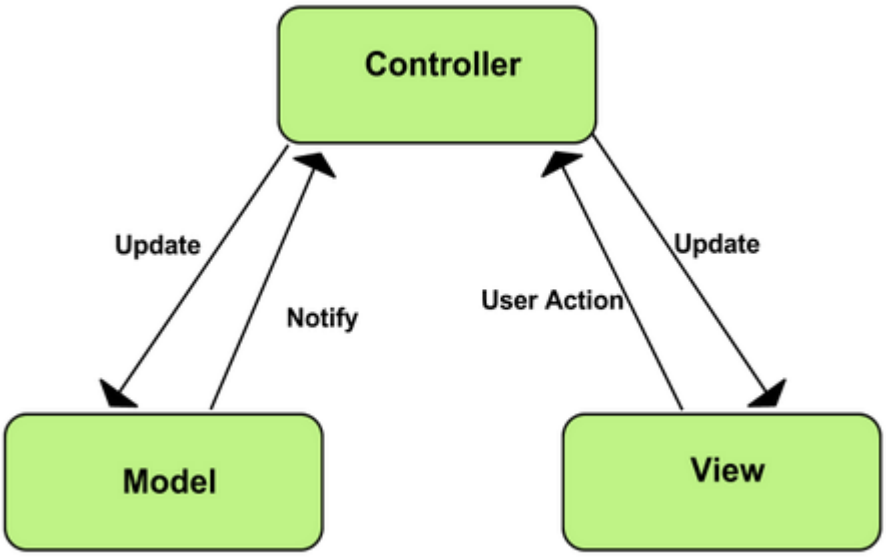


Figura 4.1: Interação entre as camadas do MVC.

Fonte: MVC Architeture, 2013.

O desenvolvimento usando o framework Django utiliza, naturalmente, a arquitetura MVC, mas o Django nomeia as camadas de forma um pouco diferente. Ao criar um novo projeto, o Django cria três estruturas que o desenvolvedor deve usar para fazer o desenvolvimento, são elas: *models*, *views* e *templates*. A *models* é um arquivo onde devem ser codificadas as classes que serão mapeadas no banco de dados, ou seja, representa a camada *models* do MVC. A *views* é um arquivo onde devem ser codificados os controladores do sistema, ou seja, os responsáveis por receber dados da tela e processá-los gerando uma resposta para a tela, e, dessa forma representa a camada *controller* do MVC. E o *templates* é uma pasta onde devem ser criados os *templates* do Django, ou seja, representam a camada *views* do MVC.

Essa diferença de nomenclatura do Django, onde o *controller* é chamado de *view* e a *view* de *template*, em relação ao MVC é um pouco estranha e pode causar alguma confusão naqueles que não estão acostumados com o framework. Outro detalhe é que, mesmo o Django criando um arquivo chamado *views* e um chamado *models*, não é necessário desenvolver todos os controladores e modelos nesses arquivos. Por uma questão de organização é melhor criar outros arquivos, com um nome mais significativo em relação a função da implementação, e apenas importar esses arquivos para os arquivos *views* e *models.*

No desenvolvimento do projeto, os modelos foram criados no arquivo padrão, models.py, mas os controladores foram criados em diferentes arquivos. Basicamente, cada CRUD, sigla para *Create* (criar), *Read* (ler), *Update* (alterar) e *Delete* (apagar), do sistema tem um arquivo separado para o seu controlador, por exemplo, há um arquivo para o controlador de pessoas, um para o de salas, e assim por diante. A única exceção são os relatórios, que foram desenvolvidos no mesmo arquivo.

4.2 Modelagem do banco de dados

No início do processo de desenvolvimento do projeto não se teve a intenção de estabelecer uma modelagem de dados rígida para o sistema. Esse processo foi sendo realizado ao longo do desenvolvimento e das reuniões de acompanhamento. Por exemplo, após o sistema ter sido colocado em produção, foi descoberta a necessidade de haver um relacionamento entre sala e usuário o que gerou uma nova *user story* e uma mudança no modelo de dados.

Para facilitar a compreensão do modelo de dados, a figura 4.2 mostra o diagrama ER resultante do processo de desenvolvimento.

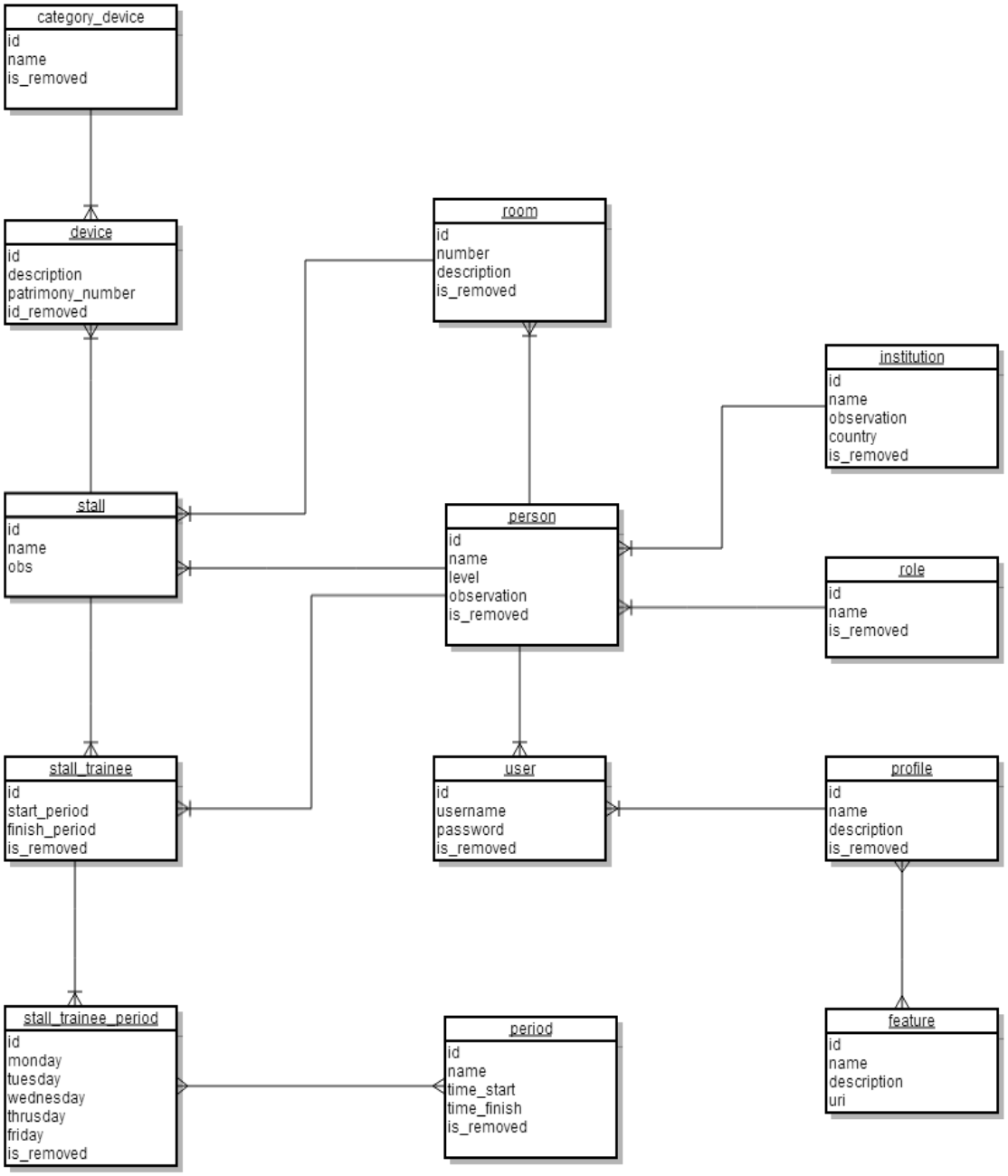


Figura 4. 2: Diagrama ER do modelo de dados.

Como é possível observar na figura 4.2, todas as entidades têm um atributo chamado *id*. Esse atributo é o um número inteiro que identifica cada uma das entidades, no banco de dados, esse atributo é a chave primária de cada uma das tabelas.

Além disso, é possível perceber que há um atributo chamado *is_removed* que, também, está presente em todas as tabelas do sistema e indica se os dados associados a ele ainda devem ser considerados. O valor padrão desse atributo é verdadeiro e só é alterado para falso quando o usuário excluir dados do sistema.

4.2.1 Entidade Room

A entidade *Room* representa as salas que podem ser usadas por bolsistas e é refletida no banco de dados com uma tabela de mesmo nome. Os campos presentes nesta entidade são: número e descrição. Ambos os campos podem ser editados na tela pelo usuário, sendo número um campo obrigatório.

Essa entidade tem um relacionamento com a entidade *User* (subseção 4.2.11), como pode ser visto na figura 4.2. Esse relacionamento indica qual usuário é o síndico da sala, ou seja, qual usuário é o responsável por manter os dados da sala atualizados. Para isso ser possível, o usuário deve ser o sindico da sala e estar associado a um perfil (subseção 4.2.12) que esteja associado a permissão "*syndic_room*" (subseção 4.2.13).

As salas também tem um relacionamento com a entidade *Stall* (subseção 4.2.2), é um relacionamento *Many-to-One*, ou seja, uma sala pode ter muitas mesas.

4.2.2 Entidade Stall

A entidade *Stall* representa as mesas que podem ser usadas pelos bolsistas em cada sala e é refletida no banco de dados com uma tabela de mesmo nome. Os atributos desta entidade são: nome e observação, sendo que o atributo nome representa o identificador de localização da mesa e é um campo obrigatório na tela, já o atributo observação não é obrigatório e é destinado a armazenar informações adicionais sobre a mesa.

Como pode ser visto na figura 4.2, a entidade *Stall* tem um relacionamento *Many-to-One* com a entidade *Person* (subseção 4.2.6), e esse relacionamento representa quem é o responsável pela mesa. O sistema impõe a restrição de que, para poder ser responsável por uma mesa, a pessoa precisa estar associada à entidade *Role* (subseção 4.2.7) cujo nome é "Orientador".

4.2.3 Entidade StallTrainee

A entidade *StallTrainee* representa um relacionamento *Many-to-Many* entre a entidade *Person* e a entidade *Stall*. Nesse caso, elementos da entidade *Person* representam os bolsistas que ocupam as mesas, elementos da entidade *Stall*. A entidade *StallTrainee* tem os seguintes atributos: período inicial e período final. Período inicial e período final descrevem a data prevista para início e fim de uso da mesa por determinada pessoa.

4.2.4 Entidade StallTraineePeriod

A entidade *StallTraineePeriod* representa os períodos do dia e os dias da semana que certa mesa será ocupada por certa pessoa. Os campos desta entidade são: segunda-feira, terça-feira, quarta-feira, quinta-feira, sexta-feira. Esses campos são do tipo *boolean* e devem receber o valor verdadeiro se a mesa for usada naquele dia.

Essa entidade tem um relacionamento *Many-to-Many* com a entidade *Period* (subseção 4.2.5). Esse relacionamento serve para indicar em quais períodos do dia a mesa será ocupada pelo bolsista.

4.2.5 Entidade Period

A entidade *Period* representa os períodos do dia que uma pessoa pode ocupar uma mesa. Essa entidade não tem uma tela para cadastro de novos dados, ou seja, os dados foram inseridos diretamente no banco de dados e não podem ser modificados. Os atributos existentes nessa entidade são: nome, hora inicial e hora final, sendo nome uma descrição do período e o atributo hora inicial e o atributo hora final indicam a hora em que o período começa e termina. Os valores existentes para essa entidade têm os respectivos valores no atributo nome: “até as 10:30”, “das 10:30 até as 12:00”, “das 12:00 às 15:30” e “a partir das 15:30”.

4.2.6 Entidade Person

A entidade *Person* representa os dados das pessoas que são cadastradas no sistema. Essa entidade tem os seguintes atributos: nome, nível e observação, onde nome é o nome da pessoa, observação é um espaço para colocar dados adicionais e nível é o grau de escolaridade.

Essa entidade tem um relacionamento *Many-to-One* com a entidade *Role* (subseção 4.2.7), esse relacionamento descreve o papel da pessoa dentro da universidade. Além disso, também há um relacionamento *Many-to-One* com a entidade *Institution* (subseção 4.2.8), que indica qual a instituição de origem da pessoa.

4.2.7 Entidade Role

A entidade *Role* representa os possíveis papéis das pessoas dentro da universidade. O único atributo presente na entidade é o nome que deve ser suficientemente claro para identificar o papel. Essa entidade tem alguns valores padrão que foram previamente cadastrados no banco de dados, são eles: “Bolsista”, “Orientador”, “Visitante” e “Temporário”. Além desses valores, outros podem ser adicionados através da interface do sistema.

4.2.8 Entidade Institution

A entidade *Institution* representa as instituições de ensino de origem das pessoas que serão cadastradas no sistema. Os atributos presentes nessa entidade são: nome, observação e país, onde nome é o nome da instituição, observação é um campo para informações adicionais e país é o país onde a instituição é sediada.

4.2.9 Entidade Device

A entidade *Device* representa os dispositivos disponíveis para o uso dos bolsistas. Essa entidade tem os seguintes campos: número de patrimônio e descrição, onde número de patrimônio é o número do patrimônio da UFRGS que aquele dispositivo tem e descrição é um campo para informações adicionais, por exemplo, para um computador poderia ser a descrição técnica do mesmo.

Essa entidade tem um relacionamento *Many-to-One* com a entidade *CategoryDevice* (subseção 4.2.10), esse relacionamento tem como objetivo indicar qual o tipo do dispositivo.

4.2.10 Entidade CategoryDevice

A entidade *CategoryDevice* representa a categoria dos dispositivos, ou seja, é responsável por identificar qual é o tipo dos dispositivos. O único atributo que essa entidade tem é o nome da categoria.

4.2.11 Entidade User

A entidade *User* representa os usuários do sistema. Os atributos dessa entidade são: *username* e senha, onde *username* é o identificador do usuário para acessar o sistema e a senha é a senha de acesso ao sistema.

Essa entidade tem um relacionamento *Many-to-One* com a entidade *Person*, esse relacionamento descreve qual pessoa aquele usuário representa. Outro relacionamento *Many-to-One* ocorre com a entidade *Profile* (subseção 4.2.12). Esse relacionamento define qual o perfil do usuário e, consequentemente, quais funcionalidades do sistema o usuário poderá acessar.

4.2.12 Entidade Profile

A entidade *Profile* representa os perfis dos usuários que acessam o sistema. Os atributos dessa entidade são: nome e descrição, onde nome é o identificador do perfil e descrição é uma breve descrição das funcionalidades que o perfil dá acesso ou uma descrição dos usuários que devem ter o perfil em questão.

Essa entidade tem um relacionamento *Many-to-Many* com a entidade *Feature* (subseção 4.2.13). Esse relacionamento define as funcionalidades do sistema que os usuários com aquele perfil terão acesso.

4.2.13 Entidade Feature

A entidade *Feature* representa as funcionalidades do sistema. As funcionalidades são fixas e foram cadastradas no sistema previamente e diretamente no banco de dados, dessa forma não há uma tela no sistema para cadastro ou manutenção das funcionalidades. Os atributos existentes nessa entidade são: nome, descrição e URI, onde nome é o nome da funcionalidade, descrição é uma descrição da mesma e URI, que vem da sigla URI, *Uniform Resource Identifier* ou, em português, identificador de recursos uniforme, e é um identificador do caminho que dá acesso a funcionalidade em questão.

4.3 Interface de navegação

Nessa seção serão descritas as telas do sistema desenvolvido e os seus respectivos fluxos. Para uma melhor compreensão dos fluxos do sistema, as telas serão agrupadas pelo fluxo das mesmas. Com isso serão divididas em quatro subseções: telas de cadastro inicial (subseção 4.3.1), telas administrativas (subseção 4.3.2), telas de cadastro principais (subseção 4.3.3) e relatórios (subseção 4.3.4).

4.3.1 Telas de cadastro inicial

As telas de cadastro inicial são compostas pelos dados básicos que o sistema precisa para começar a ser usado. Após o cadastro desses dados básicos, é possível começar a cadastrar os dados que são realmente relevantes para o sistema, como salas, mesas e bolsistas. O cadastro básico do sistema é formado pelas telas de: categorias de dispositivos (subseção 4.3.2.1), dispositivos (subseção 4.3.2.2), instituições (subseção 4.3.2.3) e pessoas (subseção 4.3.2.4).

Todos os 4 cadastros iniciais são compostos por duas telas, uma tela de listagem e uma tela de edição.

4.3.1.1 Telas de categorias de dispositivo

A tela de listagem de categorias pode ser acessada através do menu Dispositivos, subopção *Categoria de Dispositivos*. O formato da tela de listagem de categorias de dispositivos é muito parecido com o das telas de listagem que serão mostradas nas seções seguintes. Apenas o nome da categoria é mostrado na listagem.

A tela de edição de dispositivos contém apenas um campo, nesse deve ser preenchido o nome da categoria que está sendo criada. Nessa tela também se aplicam regras de validação, nesse caso, o campo nome é obrigatório.

4.3.1.2 Telas de dispositivos

O acesso da listagem de dispositivos se dá através do menu Dispositivo, submenu *Dispositivos*. A tela de listagem de dispositivos tem o mesmo padrão das telas de listagem mostradas anteriormente e mostra os seguintes dados: descrição, número de patrimônio e o nome da categoria.

A tela de edição de dispositivos tem três campos, como é possível perceber na Figura 4.3, número de patrimônio, categoria e descrição. Sendo número de patrimônio e descrição campos de texto livre e categoria uma listagem com as categorias de dispositivo previamente cadastradas. Os campos dessa tela também são validados e caso haja algum erro na validação dos mesmo a tela será exibida novamente com os dados que já foram preenchidos e uma indicação do erro que aconteceu. Nessa tela os campos número de patrimônio e categoria são obrigatórios.

É necessário cadastrar os dispositivos previamente, pois ao cadastrar uma baia será necessário selecionar os dispositivos que estão nela.

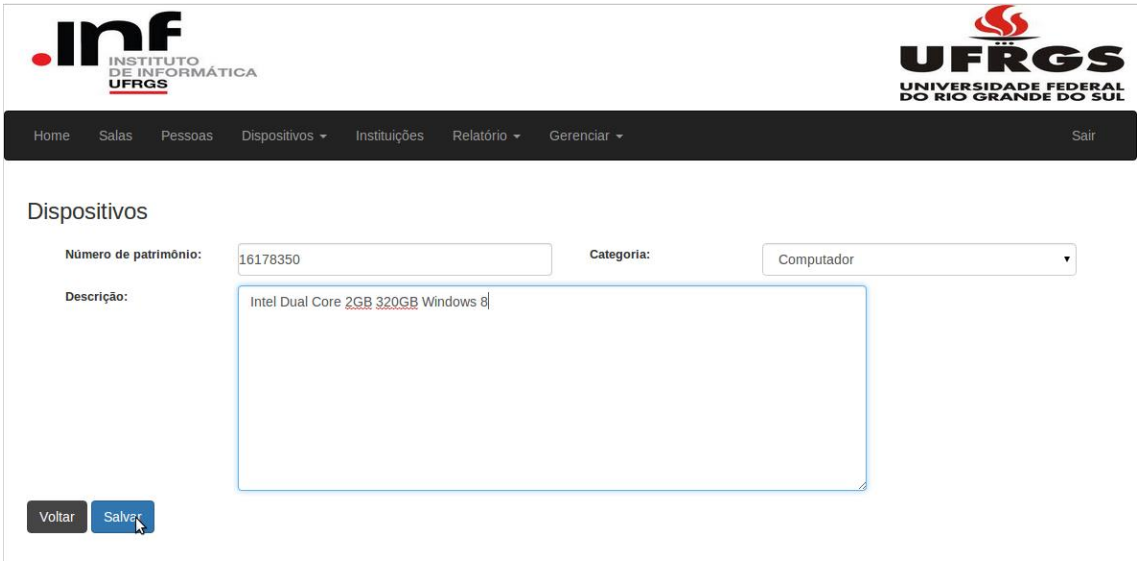


Figura 4.3: Tela de edição de dispositivos.

4.3.1.3 Telas de instituições

A tela de listagem de instituições pode ser acessada através da opção *Instituições* no menu do sistema. Essa tela segue o mesmo padrão das telas de listagem mostradas anteriormente. Na listagem são mostrados todos os campos presentes no formulário de edição.

A tela de edição de instituições contém três campos, nome, país e observação. Como se pode perceber na Figura 4.4, os campos nome e observação são campos de texto livre e o campo país mostra uma lista com todos os países. Essa lista com os todos os países está em um arquivo python, organizada em uma estrutura python do tipo lista, cada elemento dessa lista tem o nome de um país. Dessa forma a renderização dos elementos da lista de países se torna mais eficiente. Os campos obrigatórios nessa tela são nome e país.

É necessário cadastrar as instituições previamente, pois ao cadastrar uma pessoa é necessário informar a instituição de origem da mesma.

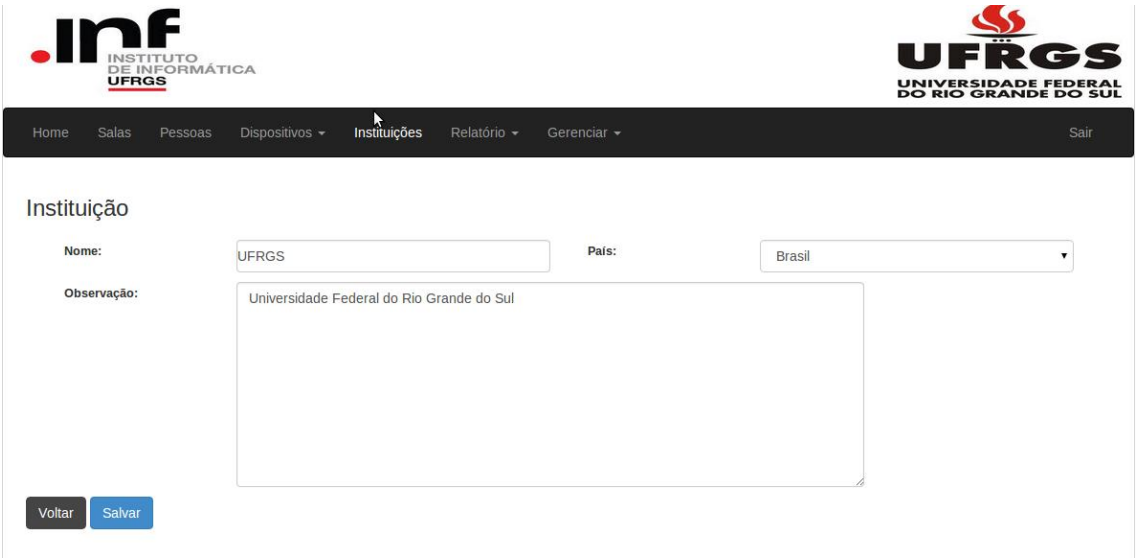


Figura 4.4: Tela de edição de instituições.

4.3.1.4 Telas de pessoas

A tela de listagem de pessoas pode ser acessada através da opção *Pessoas* do menu do sistema. Essa tela segue o mesmo formato das telas anteriores e mostra os seguintes dados de pessoas: nome, nível, papel e instituição.

Como se pode observar na Figura 4.5, a tela de edição de pessoas tem cinco campos, nome, nível, papel, instituição e observação. Os campos nome e observação são campos de texto livre, os campos papel e instituição mostram os dados previamente cadastrados das entidades *Role* e *Institution*, respectivamente. Caso o papel desejado não esteja presente nessa lista, é necessário adicionar um novo papel, ver subseção 4.3.2.1. Já o campo nível indica o qual é o nível de escolaridade da pessoa e mostra opções estáticas que foram definidas nas reuniões com os usuários.

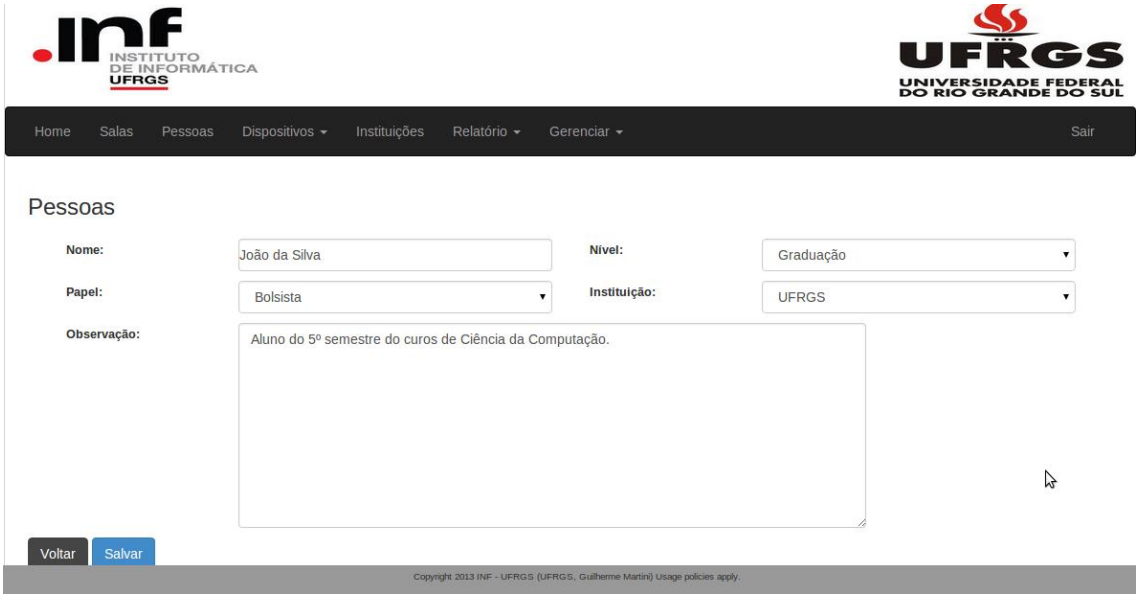


Figura 4.5: Tela de edição de pessoas.

Nessa tela todos os campos, exceto observação, são obrigatórios e em caso de erro de validação dos dados, a tela será exibida novamente para o usuário com uma indicação do erro ocorrido.

É necessário fazer este cadastro previamente uma vez que é necessário escolher uma pessoa, obrigatoriamente, para fazer o cadastro de bolsistas, baias e usuários. Além disso, é possível selecionar uma pessoa no cadastro de salas.

4.3.2 Telas administrativas

As telas administrativas são o conjunto de telas que apenas o administrador do sistema pode acessar. São elas: telas de listagem e edição de usuários, telas de listagem e edição de perfis e telas de listagem e edição de papéis. Essas telas devem ser acessadas apenas pelo administrador do sistema, pois, com acesso total as mesmas, o usuário pode editar dados de usuários já existentes, alterar permissões de perfis, criar

novos perfis e criar novos papéis, o que poderia gerar uma série de problemas para todos os usuários do sistema.

As três telas que compõe o fluxo administrativo são opções que aparecem ao clicar na opção Gerenciar do menu. Essa opção aparece para qualquer usuário que tenha permissão para editar dados de perfis ou papéis.

4.3.2.1 Telas de Papéis

O cadastro de papéis é formado por duas telas, uma de listagem e uma de edição.

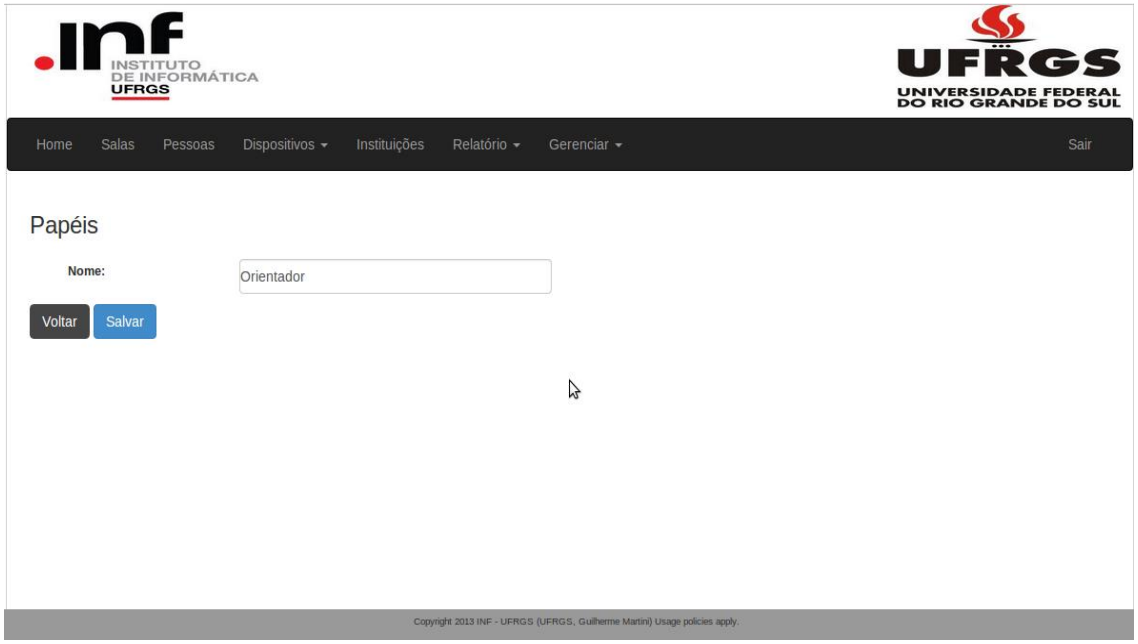


Figura 4.6: Tela de edição de papéis.

Como é possível perceber na Figura 4.6, a tela de edição de papeis é bastante simples, contendo apenas o nome do novo papel a ser criado. Os papéis criados nessa tela serão mostrados como opção de papéis ao criar ou editar uma pessoa (seção 4.3.2), além disso, o papel é uma opção de filtro no relatório por períodos (subseção 4.3.4).

A tela de listagem de papéis é bastante simples e segue o modelo das telas de listagem que serão exibidas a seguir.

4.3.2.2 Telas de perfis

As telas de perfis servem para definir os perfis de usuários do sistema, um usuário associado a determinado perfil terá acesso a todas as funcionalidades que forem escolhidas no momento da criação do perfil.

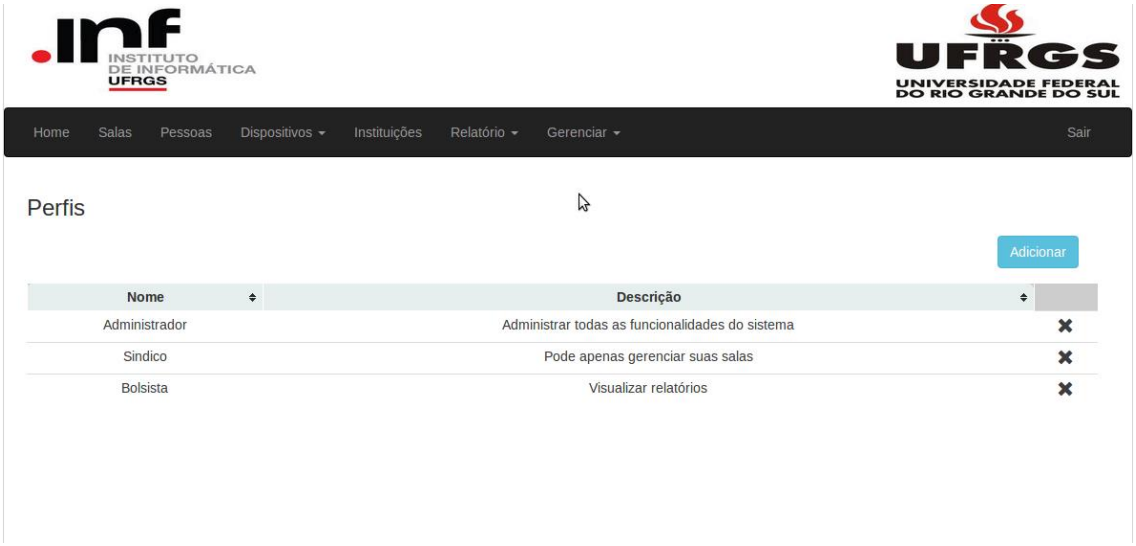


Figura 4.7: Tela de listagem de perfis.

A tela de listagem de perfis lista todos os perfis cadastrados no sistema, mostrando o nome e a descrição dos mesmos. Como podemos ver na Figura 4.7, na tela de listagem a um botão *Adicionar*, ao clicar neste botão será aberto o formulário para criação de um novo perfil. A última coluna da listagem é um *X*, esse *X* é a opção para apagar o registro correspondente àquela linha da listagem. Além disso, ao clicar em uma linha da listagem, o usuário será redirecionado para tela de edição do perfil correspondente a linha clicada.

A tela de edição de perfis possui dois campos de texto livre, para atribuir um nome e uma descrição ao perfil que está sendo criado, como pode ser visto na Figura 4.8. O outro campo que há na tela é uma lista com todas as funcionalidades do sistema, nesse campo o usuário deve selecionar todas as funcionalidades que deseja que os usuários do perfil criado possam acessar.

Na tela de edição, o usuário tem dois botões disponíveis, o botão Salvar e o botão Voltar. Os nomes são bastante explicativos, o botão Salvar salva os dados preenchidos no banco de dados, caso não haja nenhum campo com erro, por exemplo, algum campo obrigatório não preenchido, já o botão Voltar, volta para a tela de listagem de perfis. No formulário de edição de perfis todos os campos são obrigatórios.

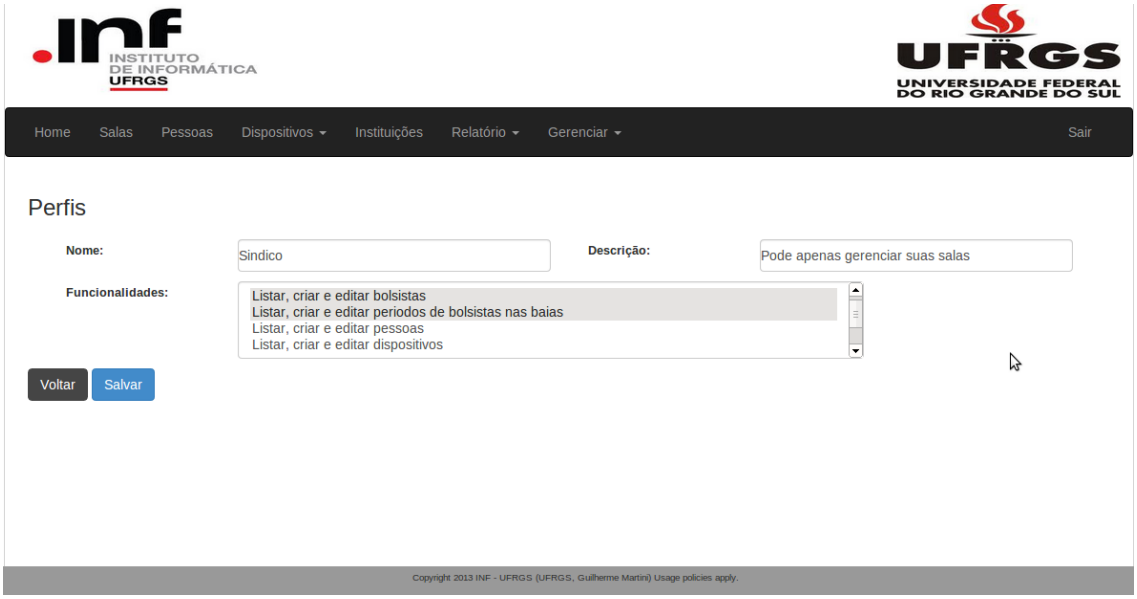


Figura 4.8: Tela de edição de perfis.

4.3.2.3 Telas de usuários

A interface de administração de usuários também é composta por uma tela de listagem e uma tela de edição.

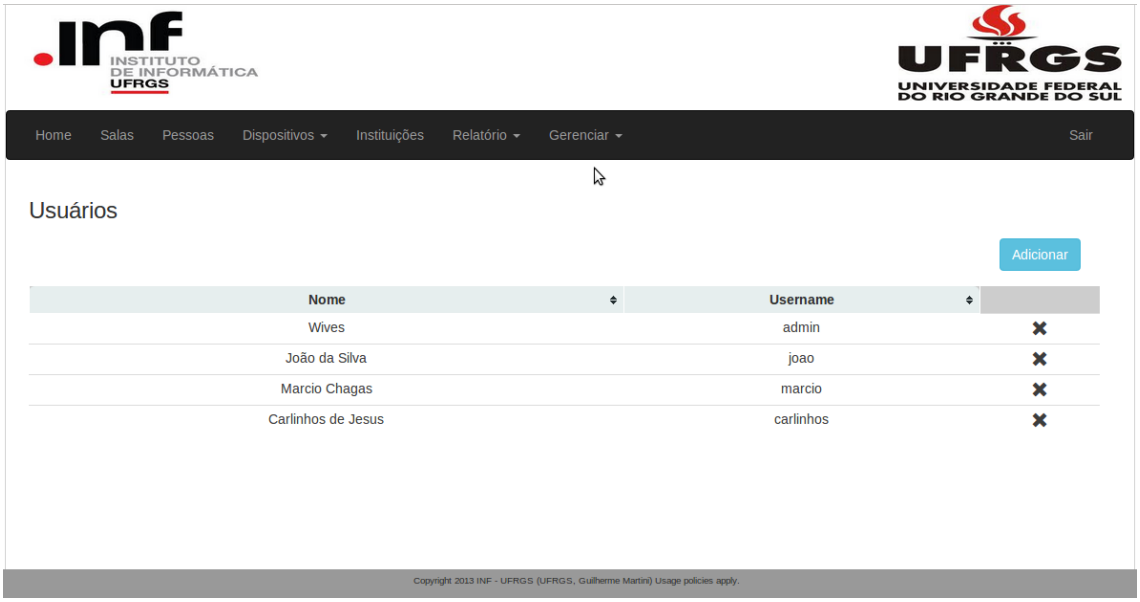


Figura 4.9: Tela de listagem de usuários.

A tela de listagem de usuários mostra o nome e o *username* dos usuários cadastrados, como pode ser visto na Figura 4.9. Assim como na tela de listagem de funcionalidades há a opção de apagar o registro correspondente a uma linha da lista e a opção de adicionar um novo usuário.

A tela de edição de usuários, conforme pode ser visto na Figura 4.10, possui três campos de texto livre, são eles: *username*, senha e confirmação de senha. Além desses, há campo Pessoa e o campo Perfil. No campo Pessoa o usuário deve escolher a pessoa que irá acessar o sistema com o usuário que está sendo criado e no campo Perfil deve

escolher qual o perfil que o usuário terá. Nessa tela todos os campos são obrigatórios e em caso de erro de validação a tela será exibida novamente mantendo os dados preenchidos pelo usuário, exceto as senhas. Outra validação que é feita nessa tela e a necessidade das duas senhas informadas serem iguais.

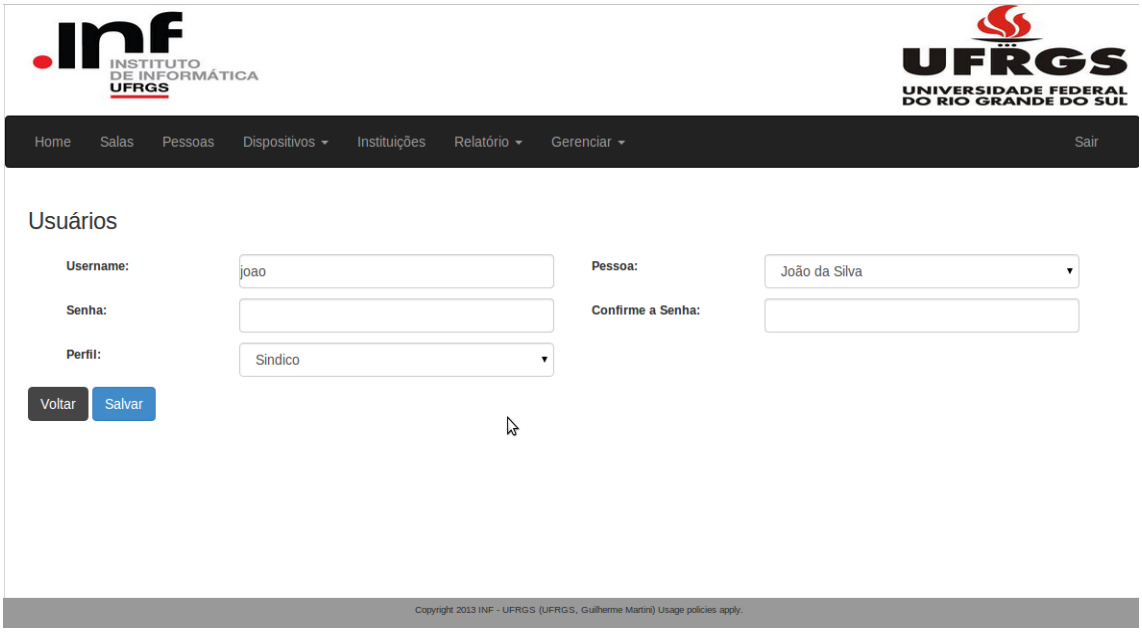


Figura 4.10: Tela de edição de usuários.

Na Figura 4.10, é possível ver que o usuário *joao* tem o perfil *Síndico*, esse perfil não tem permissão para gerenciar usuários, nem perfis do sistema.

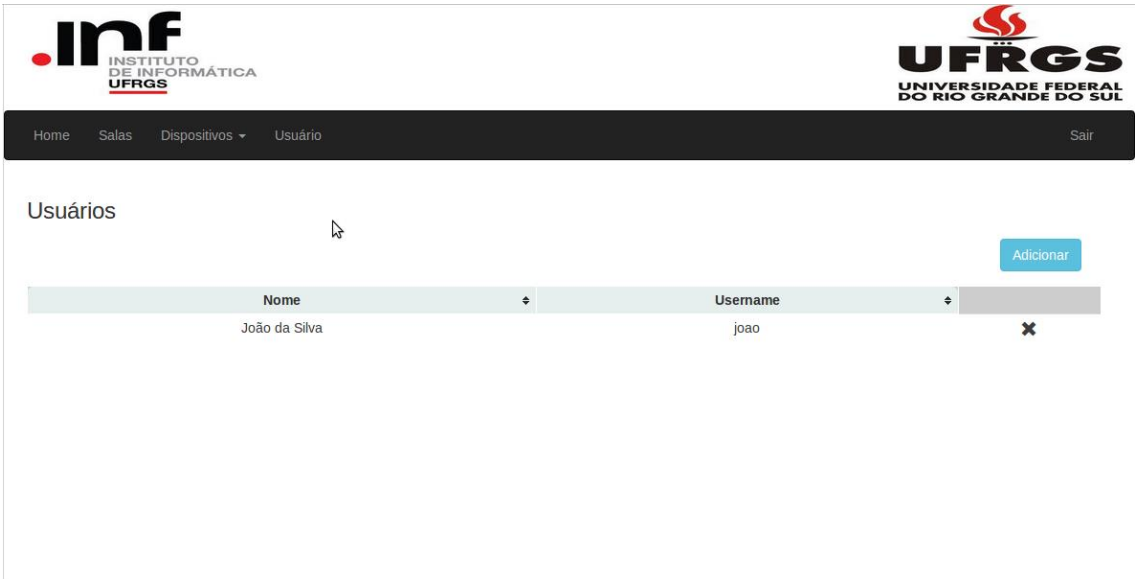


Figura 4.11: Tela de listagem de usuários para usuário com perfil Síndico.

Ao acessar o sistema com o usuário *joao*, como é possível perceber na Figura 4.11, o usuário consegue visualizar menos opções no menu do sistema, também é possível visualizar que a opção Gerenciar não está mais presente no menu, no lugar dela apareceu a opção Usuário. Isso aconteceu porque um usuário do perfil Síndico tem permissão para visualizar e editar apenas o seu usuário e não todos os usuários do

sistema como na Figura 4.9. Por esse motivo, há apenas o seu próprio usuário na listagem de usuários.

4.3.3 Telas de cadastro principais

As telas de cadastro principais correspondem ao cadastro dos dados que são de real interesse para esse trabalho. Dessa forma, esse grupo de telas é composto pelas telas de: salas (subseção 4.3.3.1), baias (subseção 4.3.3.2), bolsistas (subseção 4.3.3.3) e períodos (subseção 4.3.3.4).

4.3.3.1 Telas de salas

O fluxo de cadastro de salas é composto por duas telas, uma para listagem e uma para edição. Para acessar a tela de listagem é necessário clicar na opção *Salas* no menu do sistema. Como é possível perceber na Figura 4.12, a tela de listagem de salas é muito parecida com as outras telas de listagem do sistema. Os dados mostrados na lista são: número, descrição e quantidade de baias. A quantidade de baias é uma contagem de quantas baias foram cadastradas para cada sala.

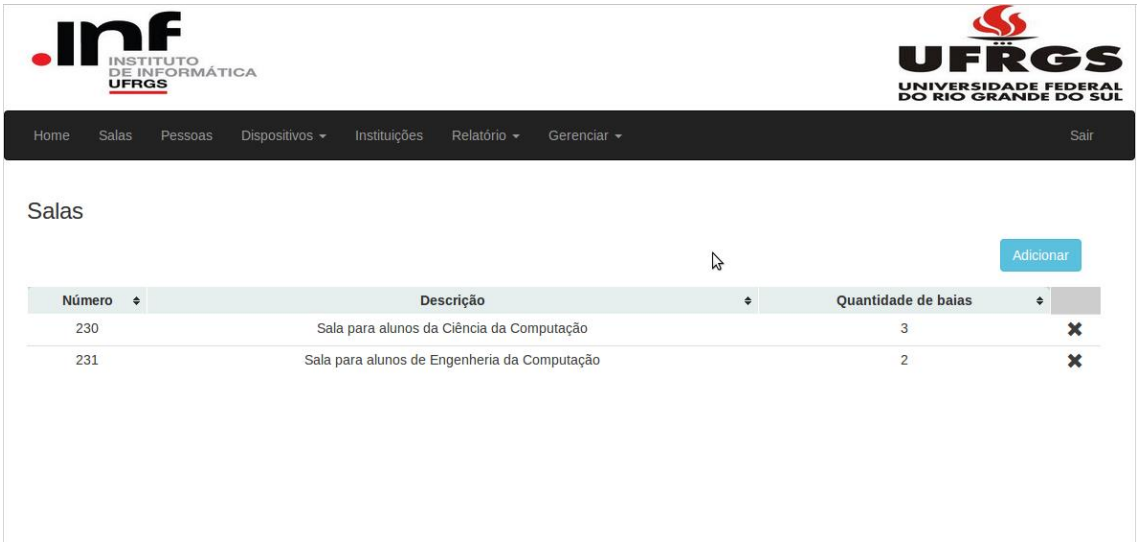


Figura 4.12: Tela de listagem de salas.

A tela de edição de salas é composta por duas partes, a primeira parte é o formulário com os dados básicos da sala, já a segunda é uma lista das baias que estão na sala em questão. Quando o estiver cadastrando uma sala nova, ele conseguirá visualizar apenas a parte do formulário do cadastro. Após preencher os dados básicos da sala e salvar, o usuário passará a visualizar também a lista de baias daquela sala, como pode ser visto na Figura 4.13.

O formulário de salas possui três campos, são eles: número, descrição e síndico. Os campos número e descrição são campos de texto livre enquanto o campo síndico permite que o usuário escolha a pessoa que é responsável por manter os dados daquela sala atualizados. Neste formulário apenas o campo número é obrigatório.

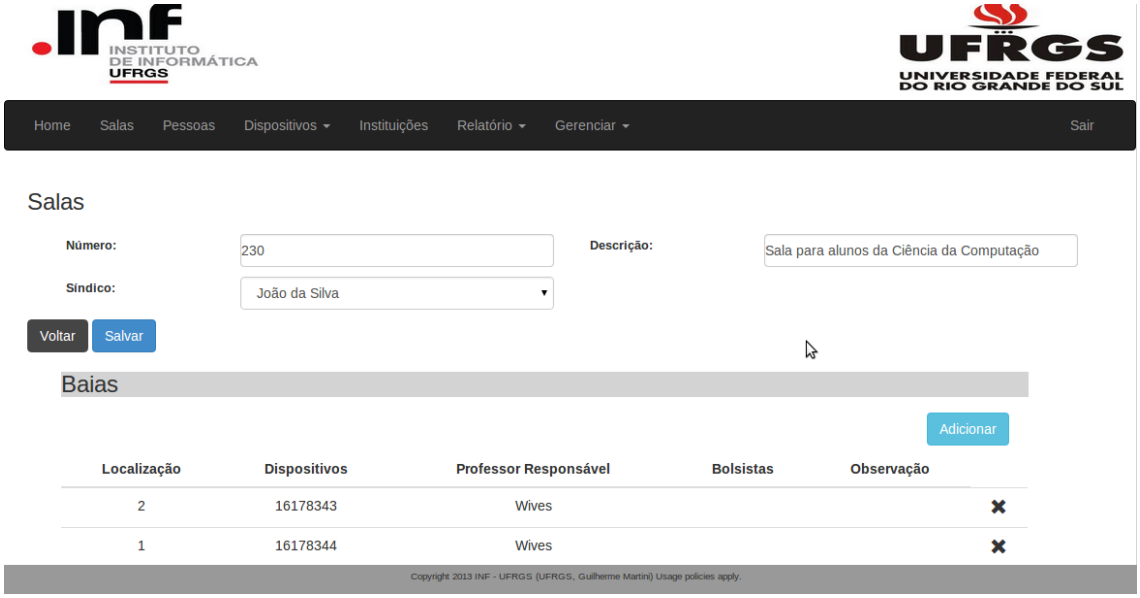


Figura 4.13: Tela de edição de salas.

4.3.3.2 Telas de baias

O fluxo de cadastro de baias começa na tela de edição de salas, onde o usuário pode clicar no botão de adicionar para inserir uma nova baia ou pode clicar em uma das linhas da listagem de baias para editar a baia correspondente a linha. Na listagem de baias são mostrados os seguintes campos: localização, dispositivos, professor responsável, bolsistas e observação.

A tela de edição de baias, da mesma forma que a edição de salas, tem duas partes, um formulário com os dados básicos da baia e uma lista com os bolsistas que usam aquela baia. A lista de bolsistas só é mostrada quando a baia já foi cadastrada no sistema e está sendo editada.

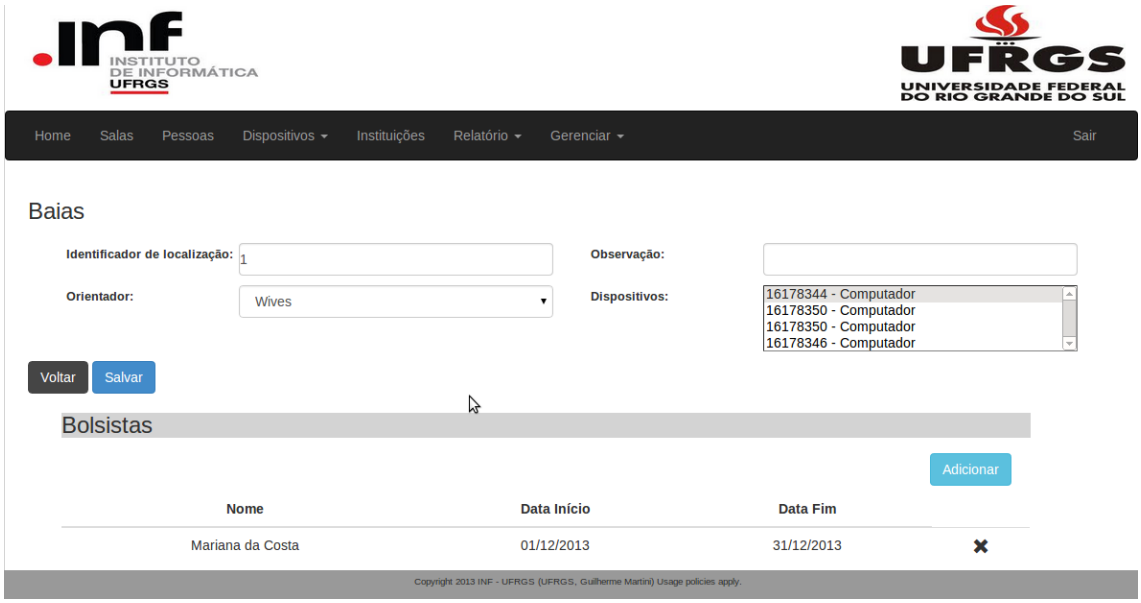


Figura 4.14: Tela de edição de baias.

Os campos presentes no formulário de baias são: identificador de localização, observação, orientador e dispositivos. Os campos identificador de localização e observação são campos de texto livre, já o campo orientador disponibiliza uma lista de pessoas que tem o papel cujo nome é *Orientador* e o campo *dispositivos* traz uma lista com todos os dispositivos que ainda não estão em nenhuma baia, o campo de dispositivos permite escolher quantos dispositivos o usuário desejar.

4.3.3.3 Telas de bolsistas

O fluxo de cadastro de bolsistas começa na tela de edição de baias, como se pode ver na Figura 4.14, onde é mostrada uma lista com todos os bolsistas que usam a baia que está sendo editada. Nessa listagem é mostrado o nome do bolsista e as datas de início e fim de utilização daquela baia pelo bolsista.

A tela de edição de bolsistas, também é formada por duas partes, sendo a primeira o formulário com os dados básicos do bolsista e a segunda uma lista com os períodos em que o bolsista utiliza a baia em que ele está.

Como pode ser visto na Figura 4.15, ao cadastrar um novo bolsista é mostrada uma tela com os campos básicos de bolsista que são: bolsista, data de início e data de fim, onde o bolsista deve ser selecionado de uma lista que contém todas as pessoas cadastradas no sistema e data de início e fim são campos de data que, ao serem clicados, mostram um calendário para o usuário poder escolher a data desejada. Mas, além dos campos básicos, são mostrados campos correspondentes ao período de uso daquele bolsista naquela baia, são os campos: períodos e os dias da semana, onde período é uma lista com os 4 períodos do dia, que são fixos no sistema, e ao lado de cada dia da semana tem um campo que pode ser marcado para indicar que naquele dia o bolsista irá usar aquela baia.

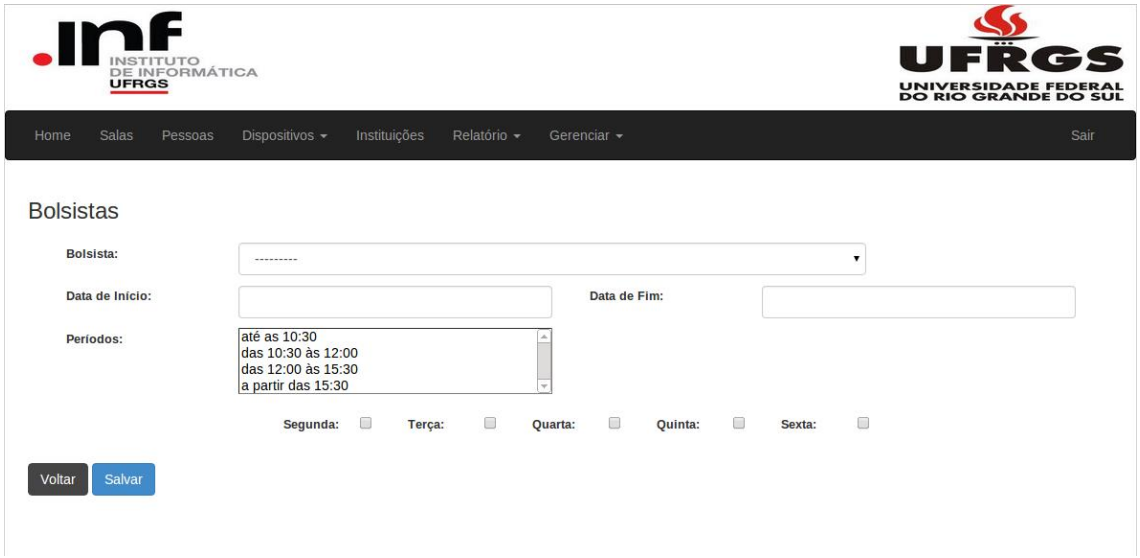


Figura 4.15: Tela de cadastro de novo bolsistas.

Já ao editar um bolsista já existente é mostrado os campo básicos do bolsista preenchidos com os valores que foram previamente cadastrados e as informações sobre o período, que inicialmente estavam junto ao formulário, agora estão em uma lista de períodos, como pode ser observado na Figura 4.16.

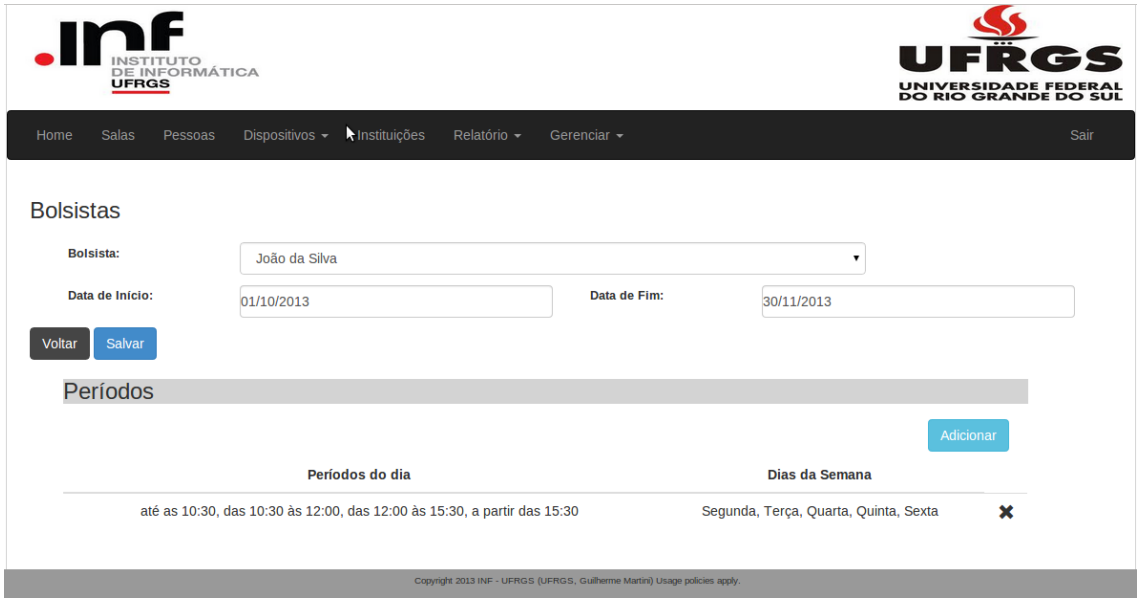


Figura 4.16: Tela de edição de bolsistas.

4.3.3.4 Telas de períodos

O fluxo de cadastro de períodos começa na tela de edição de bolsistas, onde é mostrada a lista de períodos daquele bolsista e o usuário pode editar um período já existente ou criar um novo período.

Conforme Figura 4.17, a tela de edição de períodos é composta pelos mesmos campos de período que são mostrados ao criar um novo bolsista, ver Figura 4.15. Nessa tela é possível escolher os períodos do dia e os dias da semana em que o bolsista irá ocupar a baia.

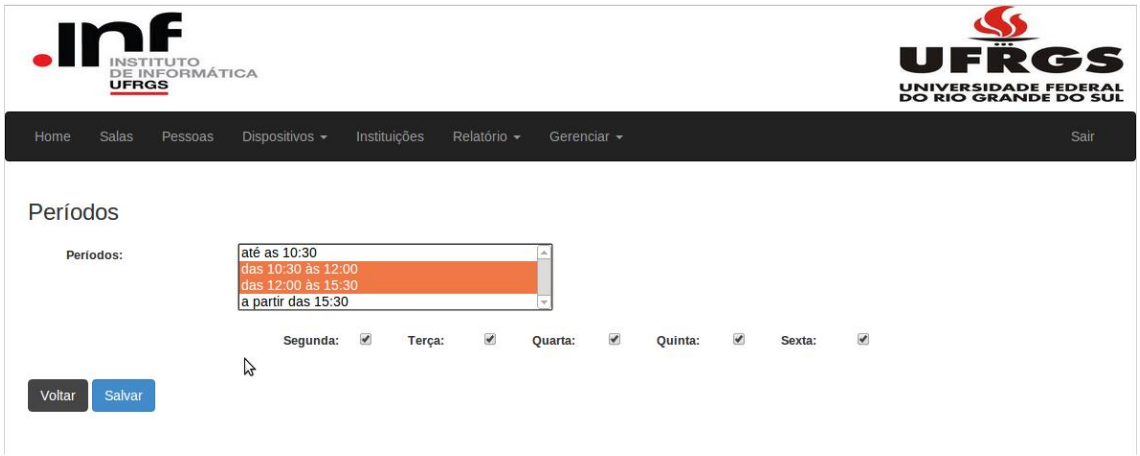


Figura 4.17: Tela de edição de períodos.

4.3.4 Telas de relatórios

O sistema tem dois relatórios, o relatório por período (subseção 4.3.4.1) e o relatório por ocupação (subseção 4.3.4.2). Os relatórios servem para dar uma visão geral e mais personalizada dos dados cadastrados no sistema.

4.3.4.1 Relatório por período

O relatório por período tem como objetivo mostrar os dados referentes a ocupação das baias pelos bolsistas, como pode ser visto na Figura 4.18. Nesse relatório são mostrados: os bolsistas, o período que eles vão ocupar a baia, tanto data inicial e final de ocupação quanto o período do dia e da semana, qual a baia e a sala em que eles estão e, também, qual o orientador do bolsista.

Na Figura 4.18 pode-se observar que é possível fazer filtros sobre os dados apresentados, o campo sobre o qual se deseja fazer o filtro pode ser escolhido na lista de filtros. Nessa figura foram aplicados dois filtros, um sobre a localização da baia, foi feito um filtro pela localização *1* e outro sobre o número da sala, foi feito um filtro pela sala *231*. Nesse caso pode-se perceber que há apenas um bolsista utilizando essa baia, o bolsista *Carlinhos de Jesus*, mas ele usa a baia em horários diferentes nas terças-feiras e quintas-feiras em relação às segundas-feiras, quartas-feiras e sextas-feiras, por isso há duas ocorrências no relatório.

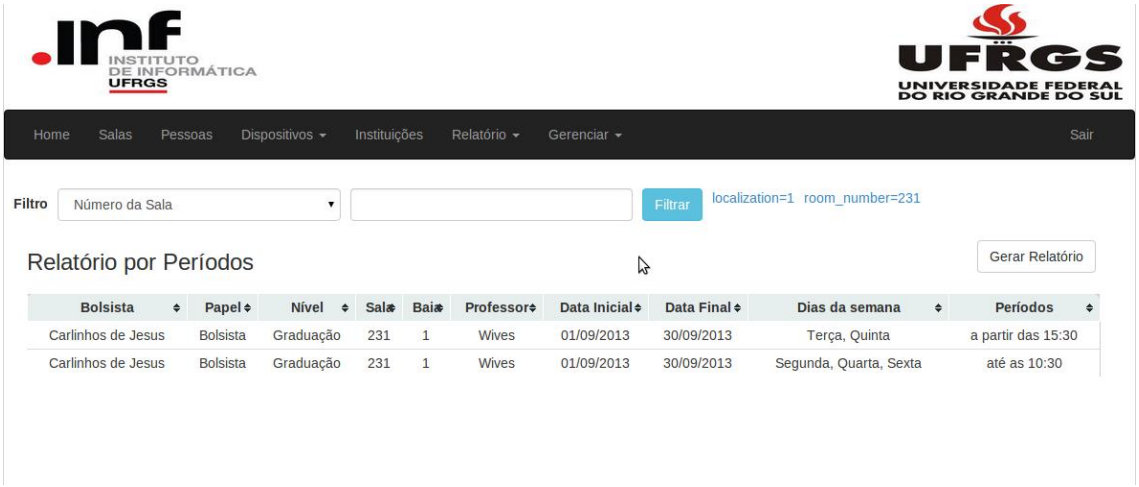
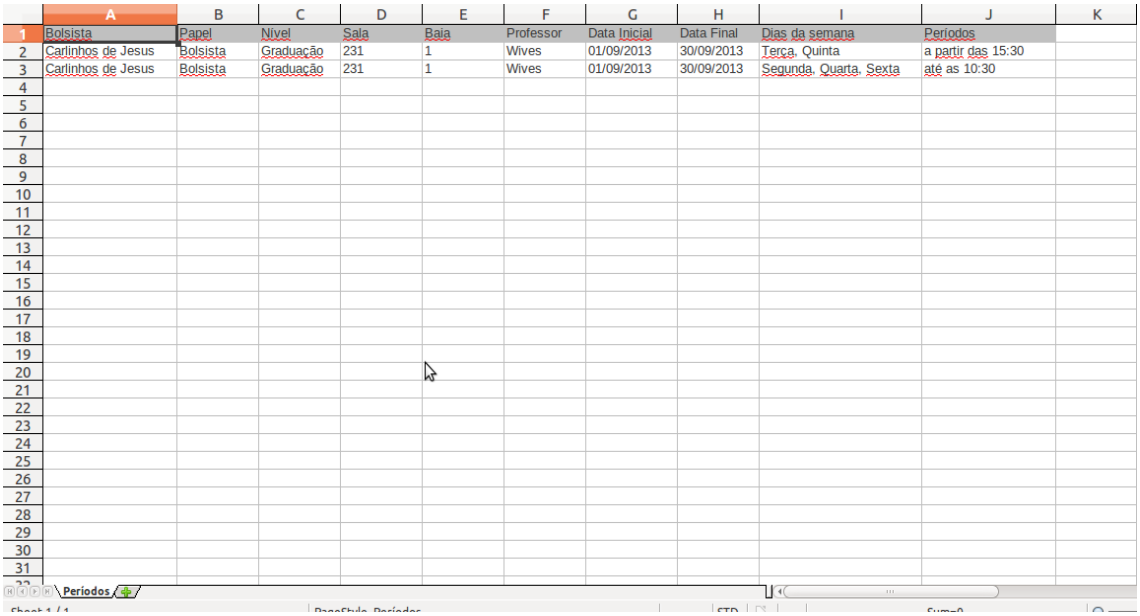


Figura 4.18: Tela de relatório por período.

Os filtros aplicados podem ser removidos, para isso basta clicar sobre o link azul que fica ao lado do botão filtrar. Além disso, há um botão *Gerar Relatório* logo acima da listagem à direita. Esse botão gera e faz *download* de um arquivo Excel contendo os mesmos dados que estão na tela, como pode ser visto na Figura 4.19.



	A	B	C	D	E	F	G	H	I	J	K
1	Bolsista	Papel	Nível	Sala	Baia	Professor	Data Inicial	Data Final	Dias da semana	Períodos	
2	Carlinhos de Jesus	Bolsista	Graduação	231	1	Wives	01/09/2013	30/09/2013	Terça, Quinta	a partir das 15:30	
3	Carlinhos de Jesus	Bolsista	Graduação	231	1	Wives	01/09/2013	30/09/2013	Segunda, Quarta, Sexta	até as 10:30	

Figura 4.19: Excel do relatório por período.

4.3.4.2 Relatório de ocupação

O relatório de ocupação é mais simples que o relatório por período, o objetivo dele é mostrar quantas das baias cadastradas em cada sala são ocupadas por quais professores. Assim como pode ser visto na Figura 4.20, esse relatório tem quatro colunas que são: número da sala, professor, quantidade de baias do professor na sala e total de baias daquela sala.

Assim como no relatório por período, é possível filtrar os dados apresentados. Além disso, também é possível gerar um arquivo Excel com os dados do relatório.

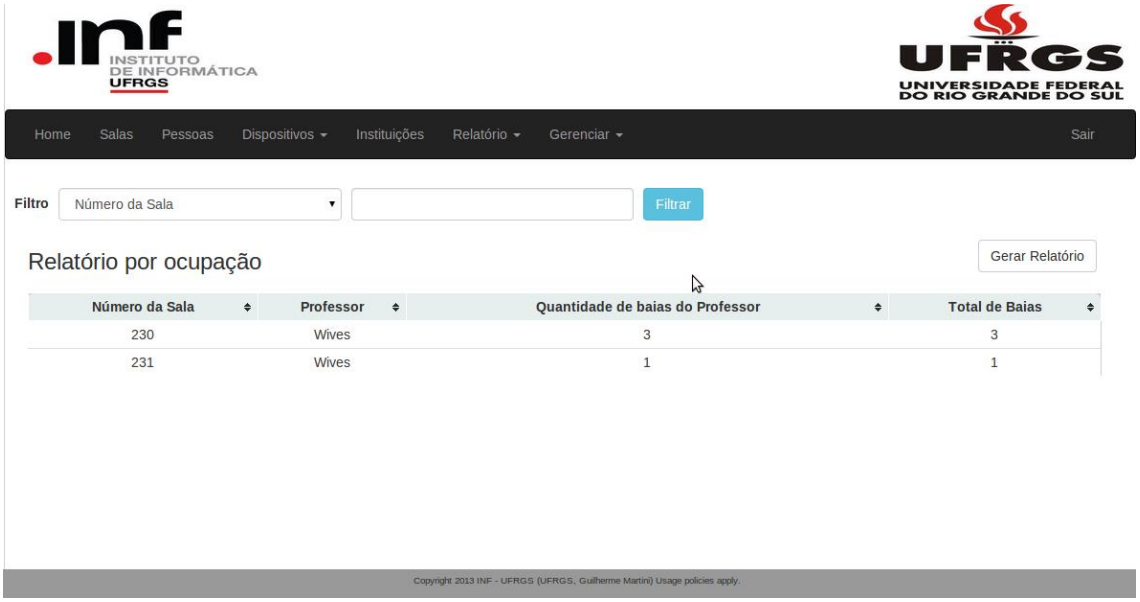


Figura 4.20: Excel do relatório por ocupação.

4.4 Implatação em produção

O projeto foi colocado em produção ao final da quinta *Sprint*, antes de estar totalmente concluído. Essa decisão foi tomada para que os usuários fossem se acustumando a usar o sistema e novas demandas e problemas fossem identificados.

O sistema foi disponibilizado em uma máquina virtual do Grupo de Pesquisa em Sistemas de Informação. Esse fato facilitou muito o processo de implantação, pois foi possível gerar uma máquina virtual remotamente e depois levar apenas o disco rígido dessa máquina virtual até o Instituto de informática para colocá-lo na máquina virtual que realmente serviu como servidor. Este servidor ainda não possui um nome amigável para ser acessado, ou seja, não há nenhum nome de domínio associado a ele, por isso, ele só pode ser acessado por ser endereço IP. Dessa forma, para facilitar o acesso ao servidor, uma das próximas ações a serem feitas será associar este servidor a um nome de domínio.

Após a subida para produção, alguns problemas foram identificados. Para os usuários poderem cadastrar os problemas com maior facilidade e, também, para manter uma lista organizada dos problemas encontrados, foi usado o Github. Ele permite o cadastrar os problemas encontrados, mas também permite que o usuário acompanhe o andamento da resolução do mesmo. O Github possui uma área para cadastro de *issues*, onde é feito o cadastro de problemas, como é possível ver na figura 4.21, a interface é bastante intuitiva, nela há uma lista com os itens já cadastrados, com uma *label* que identifica o tipo de cada *issue*, quem o abriu e o *status* da mesma.

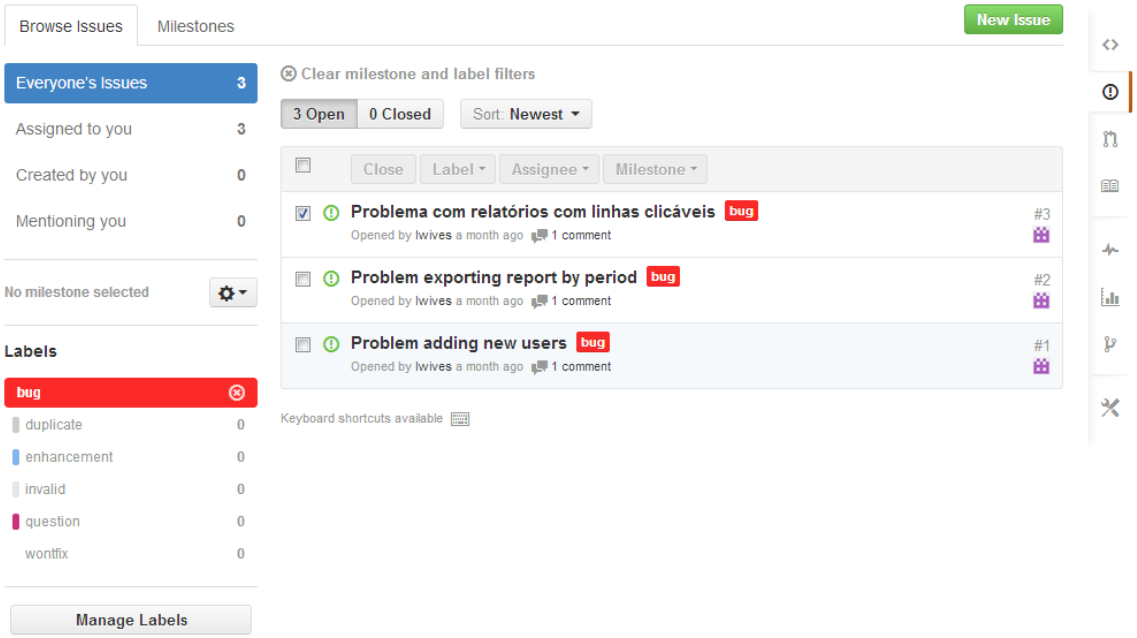


Figura 4. 21: Lista de issues do sistema.

Além disso, ao clicar em um item da lista, o usuário vê os detalhes do item. Como pode se perceber na Figura 4.22, o principal dado mostrado é o histórico de comentários, quem está trabalhando na issue, caso ela esteja aberta e a possibilidade de adicionar um comentário.

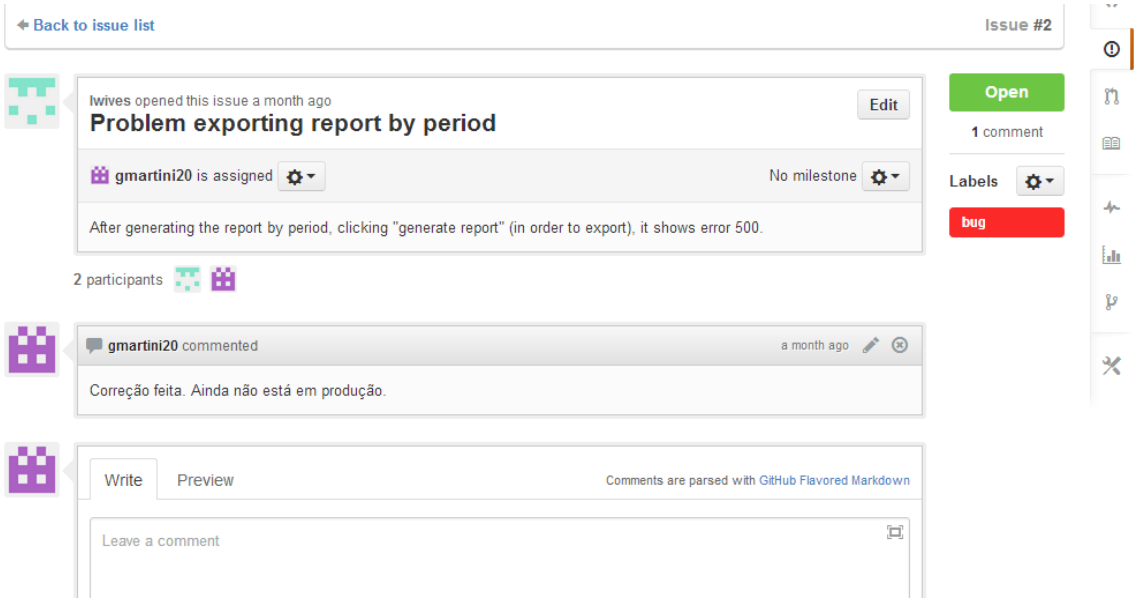


Figura 4.22: Detalhe de uma issue.

Dessa forma foi feito o cadastro, o acompanhamento dos problemas e suas respectivas correções. As correções e novas funcionalidades foram colocadas em produção através do comando git pull do GIT e algumas outras operações necessárias, tais como, o comando do Django *collectstatic* que copia os arquivos estáticos, tais como CSS e JavaScript, para a pasta raiz dos arquivos estaticos, que é definida no arquivo de configuração.

4.5 Avaliação do Sistema

Após a implantação do sistema em produção foi identificado a necessidade de desenvolver algumas melhorias e novas funcionalidades para atender de forma mais completa as necessidades dos usuários. Entre essas melhorias estão: criar uma opção para cadastro rápido de pessoas e dispositivos nas telas de cadastro de bolsistas e de baias, respectivamente; adicionar um filtro por intervalo de datas no relatório por período; adicionar um relatório por baias livres, com a possibilidade de buscar baias por períodos do dia e por salas; no cadastro de salas, adicionar um campo para informar a capacidade de baias que a sala tem, com este dado, na tela de baias, o campo identificador seria um campo de seleção com opções de acordo com o número de baias cadastrado na sala. O levantamento dessas melhorias foi realizado através de conversa com usuários.

Além disso, foi feito um questionário, anexo ao trabalho, para verificar a satisfação dos usuários com o sistema, mas não foram obtidas respostas suficientes para serem levadas em consideração. A sequência do trabalho vai consistir do reenviou deste questionário para os usuários e uma posterior análise das respostas para identificar as melhorias necessárias para o sistema. Após isso, as melhorias que forem consideradas prioritárias serão desenvolvidas e incorporadas ao sistema.

5 CONCLUSÃO

Este trabalho apresentou o processo de desenvolvimento de um sistema Web, apresentando todas as fases de desenvolvimento, desde a coleta de requisitos até o processo de implantação em produção. Neste trabalho, esse processo foi cíclico, pois os requisitos não pararam de ser coletados mesmo após a implantação do sistema em produção. Isso fez com que o resultado final do sistema fosse o mais próximo possível daquilo que o usuário deseja.

O sistema resultante deste trabalho poderia ser adaptado, com alguns ajustes, para servir a outros fins, tais como reserva de salas e recursos para reuniões ou mapeamento de recursos em laboratórios, entre outros. Fazer a adaptação deste sistema a outros fins seria interessante pelo fato dele possuir um sistema de controle de acesso bastante personalizável e já possuir meios de cadastro e associação de salas, recursos e pessoas.

O processo de desenvolvimento desse trabalho ainda não está totalmente concluído, há a necessidade de fazer um cadastro rápido de pessoas na tela de cadastro de bolsistas e um cadastro rápido de dispositivos na tela de cadastro de baias.

Isso mostra como é necessário que o cliente interaja com o sistema durante o processo de desenvolvimento, pois, na maioria das vezes, o cliente não sabe exatamente o que deseja. Assim, quando o cliente interage com o sistema durante o desenvolvimento, o impacto de mudanças de requisitos é amenizado, diminuindo o retrabalho.

Outro fator que ajuda a diminuir o retrabalho e aumentar o envolvimento do cliente no processo de desenvolvimento é o uso de uma ferramenta de gerência de projetos. Assim o cliente pode acompanhar as atividades que estão sendo feitas e entender melhor o trabalho de desenvolvimento, identificando histórias que não estão de acordo com as suas necessidades.

Os resultados obtidos com o desenvolvimento do sistema proposto foram significativos, uma vez que houve uma grande melhoria na forma de gerência dos recursos do Grupo de Pesquisa em Sistema de Informação que facilitou a identificação de fim de períodos de ocupação de baias por bolsistas e aumentou a confiabilidade do processo de cadastro de bolsistas. Além disso, o sistema mantém o histórico de uso das baias, o que permite fazer consultas de uso anteriores das baias.

Ainda há a possibilidade de inserir novas funcionalidades ao sistema em trabalhos futuros, poderia ser feito um aplicativo para android ou IOS para ler a etiqueta de patrimônio da UFRGS e mostrar os dados do bolsista que está trabalhando naquele dispositivo e o professor responsável por ele. Além disso, poderia ser feita uma

melhoria para fazer com que a identificação da baia na sala fosse feita de forma visual e não apenas com um identificador.

REFERÊNCIAS

ABOUT PostgreSQL. **PostgreSQL**. Disponível em: <http://www.postgresql.org/about/>. Acesso em: mar. 2013.

ABOUT Python, **Python**. Disponível em: < http://python.org/about/>. Acesso em: set. 2013.

AJAX Tutorial. w3schools. Disponível em: <http://www.w3schools.com/ajax/default.ASP>. Acesso em: jul 2013

OBE, Regina; HSU, Leo. **PostgreSQL Up and Running.** 1ª ed. Sebastopol: O'Reilly Media, 2012.

BURBECK, Steve. **Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)**. Disponível em: <http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html>. Acesso em: out. 2013

CHACON, Scott. **Pro Git**. 1ª ed. Nova Iorque: Apress, 2009

COCHRAN, David. **Twitter Bootstrap Web Development How-To**. 1ª ed. Birmingham: Packt Publishing, 2012.

COHN, Mike. User Stories Applied: For Agile Software Development. Boston: Addison-Wesley Professional, 2004.

FLANAGAN, David. **JavaScript: The Definitive Guide**. 6ª ed. Sebastopol: O'Reilly Media, 2011

GETTING started. **Bootstrap**. Disponível em: <http://getbootstrap.com/getting-started/>. Acesso em: abr. 2013.

GETTING started with Trello. **Trello Help**. Disponível em: <http://help.trello.com/customer/portal/articles/891656-getting-started-with-trello>. Acesso em: mar. 2013.

HOLOVATY, Adrian; KAPLAN-MOSS, Jacob. **The Definitive Guide to Django: Web Development Done Right (Expert's voice in Web Development)**. 2ª ed. Nova Iorque: Apress, 2009.

JQUERY API, **Jquery**. Disponível em: < http://api.jquery.com/>. Acesso em: abr. 2013.

LAGRONE, Benjamin. **HTML5 and CSS3 Responsive Web Design Cookbook**. Birmingham: Packt Publishing, 2013.

LARSEN, Rob. **Beginning HTML and CSS**. Indianapolis: John Wiley & Sons, 2013.

MVC ARCHITECTURE. **Chrome apps**. Disponível em: <http://developer.chrome.com/apps/app_frameworks.html>. Acesso em: set. 2013

REENSKAUG, Trygve, **The original MVC reports**. Disponível em: <https://www.duo.uio.no/bitstream/handle/10852/9621/ReenskaugMVC.pdf?sequence=1>. Acesso em: out. 2013

SCHWABER, Ken; SUTHERLAND, Jeff. **Scrum Guide**. Disponível em: <https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide.pdf#zoom=100>. Acesso em: jul. 2013

THE DJANGO template language. Django Documentation. Disponível em: <https://docs.djangoproject.com/en/dev/topics/templates/>. Acesso em: jun. 2013.

YORK, Richard. **Beginning JavaScript and CSS Development with JQuery**. Indianapolis: Wiley Publishing, 2009.

WORKING with forms. Django Documentation. Disponível em: <https://docs.djangoproject.com/ en/dev/topics/forms/>. Acesso em: jun. 2013.

ANEXO

Questionário com questões enviadas para os usuários avaliarem o sistema e darem sugestões para melhorias futuras no sistema. Devido às poucas respostas recebidas, este questionário será reenviado para os usuários.

1) Como foi a experiência de utilização do sistema? (Relativo a experiência de uso, fluxo das telas e facilidade para o usuário saber o que deve fazer.)
   a. Muito Satisfeito
   b. Satisfeito
   c. Médio
   d. Insatisfeito
   e. Muito Insatisfeito

2) Em relação a performance do sistema ao realizar as operações, você está?
   a. Muito Satisfeito
   b. Satisfeito
   c. Médio
   d. Insatisfeito
   e. Muito Insatisfeito

3) Em relação as mensagens de aviso (sucessos e erros) do sistema, você está?
   a. Muito Satisfeito
   b. Satisfeito
   c. Médio
   d. Insatisfeito
   e. Muito Insatisfeito

4) Comparado com a utilização de uma planilha compartilhada pelo Google Docs, você acredita que o sistema facilitou e tornou mais organizada a gerência de recursos?
   a. Sim, melhorou muito.
   b. Sim, melhorou um pouco.
   c. Não houve mudança.
   d. Não, piorou um pouco.
   e. Não, piorou muito.

5) O sistema satisfaz as necessidades dos usuários?
    a. Satisfaz completamente.
    b. Satisfaz parcialmente.
    c. Não satisfaz.

6) Você tem sugestões para futuras melhorias no sistema? Liste todas.