Controle Multimídia Universal

Documento de Descrição do Projeto

Versão 2.0



**Histórico da Revisão**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| 02/09/2013 | 1.0 | Atualização da arquitetura | Felipe Lindemberg; Felipe Miranda; Lucas Salvino; Marcos Antônio |
| 16/09/2013 | 1.1 | Criação de um padrão de Backlog | Felipe Lindemberg |
| 01/10/2013 | 1.2 | Documentação das chamadas do servidor. Documentação de Jenkis e Pep8. | Felipe Lindemberg |
| 07/10/2013 | 1.3 | Padronização do documento. | Felipe Lindemberg |
| 23/10/2013 | 1.4 | Formatação/ indexação (WORD) | Felipe Lindemberg |
| 25/10/2013 | 2.0 | Adição de testes de usabilidade, prototipação de tela, revisão das chamadas dos servidores, atualização das referências | Felipe Lindemberg; Felipe Miranda; Marcos Antônio; Lucas Salvino |

**Índice Analítico**

[1. Objetivo 4](#_Toc370805271)

[2. Motivação 4](#_Toc370805272)

[3. Solução 4](#_Toc370805273)

[4. Arquitetura 5](#_Toc370805274)

[5. Protocolo de Bluetooth 6](#_Toc370805275)

[6. Protocolo de Rede 6](#_Toc370805276)

[7. Requisitos Não Funcionais 6](#_Toc370805277)

[8. Requisitos Funcionais 7](#_Toc370805278)

[9. Casos de Uso 8](#_Toc370805279)

[9.1 Fluxos de eventos dos casos de uso 8](#_Toc370805280)

[10. Processo de Desenvolvimento 13](#_Toc370805281)

[10.1. Repositório 13](#_Toc370805282)

[10.2. Integração Contínua 13](#_Toc370805283)

[10.3. Revisão de Código 14](#_Toc370805284)

[10.4. Gerência de Riscos 16](#_Toc370805285)

[11. Cronograma Inicial 17](#_Toc370805286)

[12. Padrões de comandos e métodos dos servidores 18](#_Toc370805287)

[12.1. Chamadas do Servidor Residência 18](#_Toc370805288)

[12.2. Chamadas do Servidor Cômodo 20](#_Toc370805289)

[12.3. Comandos enviados aos equipamentos 22](#_Toc370805290)

[13. Prototipação de Telas 24](#_Toc370805291)

[14. Testes 26](#_Toc370805292)

[14.1. Pep8 e Pylint 26](#_Toc370805293)

[14.2. Testes de Unidade 26](#_Toc370805294)

[14.3. Testes de Usabilidade 28](#_Toc370805295)

[14.3.1 Análise de cobertura dos testes de usabilidade 42](#_Toc370805296)

[14.3.2 Resumo das Estatísticas Gerais dos Testes de Usabilidade 44](#_Toc370805297)

[15. *Bugs* 44](#_Toc370805298)

[16. Anexos 45](#_Toc370805299)

[A - Pep 8 - Python style guide checker 45](#_Toc370805300)

**Documento de Descrição do Projeto**

# 1. Objetivo

Desenvolver um sistema de controle para dispositivos multimídias residenciais (controle de áudio e vídeo dos aparelhos de uma residência).

Tecnologias desejadas: Controle/Simulação de dispositivos, Bluetooth[[1]](#footnote-1), Redes Domésticas (UPnP[[2]](#footnote-2)), Controle do sistema (cliente/servidor).

# 2. Motivação

Visando a integração e a mobilidade do usuário com os sistemas multimídias residenciais, especificamente TV e SOM, este projeto propõe uma nova forma de interação e conectividade na relação ambiente e usuário.

Atualmente, tal interação é bastante limitada, deixando o usuário “preso” ao ambiente para que ele possa acompanhar determinado conteúdo. Para isso, esta proposta dispensa a necessidade de intervenção do usuário para redefinir suas preferências no que diz respeito a configuração dos aparelhos multimídias (TV e SOM) no ponto em que estava sendo acompanhado anteriormente. Dessa forma, tais informações serão transmitidas para aparelhos similares por onde quer que usuário se desloque dentro de sua residência.

# 3. Solução

A proposta de solução para o problema supracitado é desenvolver um controle remoto universal capaz de controlar os equipamentos multimídias de uma residência que estejam interconectados por meio de uma rede doméstica. O controle será capaz de identificar o ambiente (cômodo) no qual esteja inserido, como, por exemplo, uma sala ou quarto, e controlar os aparelhos que fazem parte do mesmo cômodo.

Quando o usuário mudar de cômodo, e.g., sair da sala para o quarto, o sistema deverá reconhecer que o controle se encontra em um novo cômodo, utilizando para isso o sinal de comunicação Bluetooth, e automaticamente ligar os aparelhos deste cômodo, preservando as configurações dos aparelhos do cômodo anterior (volume e/ou canal) no qual o usuário estava, bem como desligar os aparelhos deste cômodo caso não haja pessoas no referido cômodo.

Com o uso desta solução será possível economizar energia, ao desligar os equipamentos dos ambientes, tanto os que não possuir pessoas, como também aqueles que apresentarem um período de inatividade dos usuários, como, por exemplo, por meio de sensores de movimento.

# 4. Arquitetura

A arquitetura proposta para o Controle Multimídia Universal consiste em 3 módulos: Controle, Cômodo e Residência.

* Controle – Permite enviar comandos para um determinado equipamento de um cômodo.
* Residência – Possui um servidor que permite a adição de cômodos para serem controlados. É papel da residência repassar os comandos enviados do controle para os cômodos.
* Cômodo – Possui equipamentos controláveis (Som e TV).



Figura 1 - Simples arquitetura do sistema

O sistema consiste em representar uma Residência com seus respectivos Cômodos que possuem Aparelhos multimídia, e, utilizando o Controle, controlar os equipamentos de um determinado cômodo por meio do envio de comandos usando a rede doméstica.

Os cômodos são aplicações independentes, o que irá possibilitar a criação de várias instâncias deste para uma residência e adicioná-las ao sistema para que seus aparelhos possam ser controlados. O controle também é uma aplicação, cuja responsabilidade é enviar comandos para alterar as configurações dos aparelhos multimídia de um cômodo, além de detectar em qual cômodo o mesmo se encontra.

Para obter um tempo de resposta aceitável para o usuário, em relação ao tempo de enviar um comando para um aparelho e esse comando ser executado, cada cômodo terá um servidor que será responsável por controlar o aparelho (uso do padrão de projeto *Observer[[3]](#footnote-3)*).

O “Servidor Principal” é uma ponte entre o controle e o servidor de um determinado cômodo, sendo o responsável por repassar o comando para o ambiente correto.

A simulação para identificar em qual ambiente o controle se encontra é realizada via Bluetooth. Assim, cada cômodo terá um sensor que fornecerá sua localização (sala, quarto) e caso o controle receba essa informação o mesmo irá atualizar o seu parâmetro de localização, permitindo a partir deste instante realizar ações de controle.

# 5. Protocolo de Bluetooth

O protocolo adotado para a simulação da identificação do ambiente pelo controle foi o RFCOMM[[4]](#footnote-4). Tal protocolo objetiva a comunicação ponto-a-ponto entre dispositivos Bluetooth, atendendo desta forma um requisito necessário para a simulação da localização dos cômodos para o controle.

Por medidas de segurança, todos os dispositivos Bluetooth’s utilizados no sistema estão pré-definidos. Sendo assim, não é permitido o uso de Bluetooth’s externos para utilizar o sistema (deve-se adicionar no código, o MAC do dispositivo desejado) para que seja permitida a comunicação entre o cômodo e o controle.

# 6. Protocolo de Rede

O modelo adotado para controlar os aparelhos por meio da rede foi o TCP/IP[[5]](#footnote-5). Tal modelo foi escolhido para garantir que os comandos enviados pelo controle sejam recebidos pelos aparelhos de um determinado cômodo.

# 7. Requisitos Não Funcionais

|  |  |
| --- | --- |
| **Requisito Não Funcional** | |
| Requisitos mínimos | Os requisitos mínimos para a implementação desse sistema são: dois computadores utilizando o sistema operacional Linux, ambiente de programação Python[[6]](#footnote-6) instalado, acesso à rede e dispositivos para comunicação Bluetooth. |
| Usabilidade | O aplicativo cliente se comunicará com o usuário por meio de uma interface gráfica simples e de fácil utilização. Alguns poucos botões de ação presentes na interface do aplicativo disponibilizarão todas as demais funcionalidades do sistema tais como:   * enviar a localização em que o usuário se encontra para o servidor; * trocar de canal e alterar o volume. |
| Confiabilidade | Em caso de “queda” do sistema elétrico, a aplicação contará com um servidor capaz de salvar as informações do estado atual do cliente em um arquivo. Desse modo, ao se reconectar com o sistema, essas informações serão disponibilizadas ao usuário. Para lidar com a perda de dados e pacotes, a forma de conexão do cliente com o servidor será TCP. Em casos de perda de conexão, falta de alcance e operações que sejam realizadas pelo usuário de forma incorreta, o sistema será capaz de tratar todas as exceções referentes a esses problemas. |
| Desempenho | Após o envio do comando do usuário por meio do aplicativo cliente, o tempo de resposta do servidor para realizar as simulações deve ser abaixo de dois segundos. |

Tabela 1 - Tabela de requisitos não funcionais

# 8. Requisitos Funcionais

|  |  |
| --- | --- |
| **Requisito Funcional** | |
| Identificar o ambiente que o usuário se encontra | Por meio de um dispositivo de comunicação Bluetooth incorporado ao controle será possível identificar em qual cômodo o mesmo se encontra. O servidor também terá informações sobre a existência de pessoas em um cômodo, mesmo após a saída do usuário com o controle de um cômodo para outro. |
| Controlar os aparelhos de acordo com o ambiente do usuário | O controle irá mandar a localização em que se encontra para o servidor, e ao trocar de canal ou alterar o volume será alterado o dispositivo referente àquele ambiente. |
| Identificar se existem pessoas no ambiente | A identificação de pessoas no ambiente será realizada mediante a presença do controle ou de usuários no ambiente. A presença é simulada pelo servidor que terá os métodos para:   * *addPessoa(local{quatro, sala});* * *removerPessoa(local{quarto, sala})*. |
| Ligar automaticamente os aparelhos de acordo com o ambiente do usuário | Por meio da identificação realizada pelo sensor informando o cômodo atual em que o usuário se encontra, se os aparelhos multimídia (SOM ou TV) estiverem desligados sem quaisquer pessoas assistindo, os mesmos serão automaticamente ligados de acordo com os seus respectivos estados quando antes do usuário se deslocar (ligado ou desligado com a devida programação e volume). |
| Desligar os aparelhos caso o usuário não esteja no ambiente ou não seja detectado a presença de pessoas por um determinado intervalo de tempo | Após os usuários se retirarem do ambiente será contabilizado um certo intervalo de tempo em que ocorreu inatividade, ocasionando em seguida o desligamento dos equipamentos do ambiente. |

Tabela 2 - Tabela de requisitos funcionais

# 9. Casos de Uso

A Figura 2 mostra as interações possíveis do usuário com o sistema. São elas: controlar os equipamentos de um cômodo ou mudar de um cômodo para outro. Ao controlar um aparelho, o esperado é que o comando do controle seja refletido no equipamento. Já quando o usuário muda de cômodo, novos casos de uso são criados, como desligar e ligar os equipamentos de um cômodo.

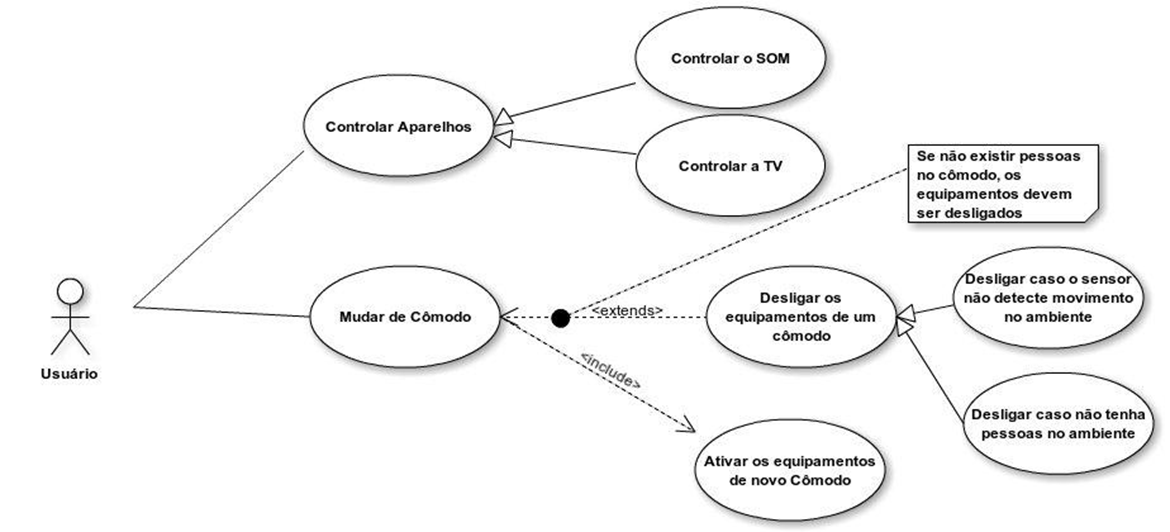


Figura 2 - Diagrama de casos de uso do sistema

## 9.1 Fluxos de eventos dos casos de uso

Casos de uso possuem definições e regras para que sejam realizados com sucesso. Para cada caso foi criado uma tabela explicando o que é e como funciona o caso, além de apresentar as pré-condições (o que é necessário para o sistema executar o caso) e as pós-condições (o que deve ter sido feito após a execução do caso). Também é mostrado um fluxo perfeito de funcionamento do caso, sem considerar imprevistos durante a sua execução.

|  |  |
| --- | --- |
| **Controlar aparelhos** | |
| Descrição | O usuário poderá controlar os aparelhos multimídias do ambiente em que o controle se encontra. |
| Pré-condição | O controle deve estar conectado a uma rede doméstica e saber o cômodo onde se encontra. |
| Pós-condição | Após enviar um comando (aumentar volume, mudar canal) o mesmo deve ser executado no aparelho. |
| Fluxo Principal | 1 - O controle está em algum cômodo identificado por Bluetooth. |
| 2 - O usuário seleciona no controle o aparelho que deseja controlar. |
| 3 - O usuário seleciona a função que pretende configurar (volume, canal, Power). |
| 4 - A função deve ser configurada no aparelho selecionado referente ao cômodo correto. |

|  |  |
| --- | --- |
| **Mudar de cômodo** | |
| Descrição | O usuário poderá se deslocar com o controle de um ambiente para outro e o mesmo será capaz de identificar em qual cômodo se encontra. |
| Pré-condição | O controle deve estar conectado a uma rede doméstica e conhecer o local onde se encontra. |
| Pós-condição | Após mudar de cômodo, o controle deve atualizar o ambiente em que se encontra. |
| Fluxo Principal | 1 - O controle se encontra em algum cômodo identificado por Bluetooth. |
| 2 - O usuário leva o controle para outro cômodo. |
| 3 - O ambiente em que o controle está envia por Bluetooth a atualização para o novo local. |
| 4 - O controle atualiza sua localização. |

|  |  |
| --- | --- |
| **Controlar Som** | |
| Descrição | O usuário poderá controlar o SOM do ambiente em que se encontra. |
| Pré-condição | O controle deve estar conectado a uma rede e conhecer o local onde se encontra. O botão de SOM no controle deve estar selecionado. |
| Pós-condição | Após enviar o comando, o resultado de sua ação deve ser "refletido" no SOM. |
| Fluxo Principal | 1 - O controle se encontra em algum cômodo identificado por Bluetooth. |
| 2 - O usuário seleciona o botão SOM. |
| 3 - O usuário seleciona o comando que deseja enviar para controlar o SOM. |
| 4 – O resultado da ação do comando enviado deve ser refletido no SOM. |

|  |  |
| --- | --- |
| **Controlar TV** | |
| Descrição | O usuário poderá controlar a TV do ambiente em que se encontra. |
| Pré-condição | O controle deve estar conectado a uma rede doméstica e conhecer o local onde se encontra. O botão de TV no controle deve estar selecionado. |
| Pós-condição | Após enviar o comando, o resultado da ação do mesmo deve ser "refletido" na TV. |
| Fluxo Principal | 1 - O controle está em algum cômodo identificado por Bluetooth. |
| 2 - O usuário seleciona o botão TV. |
| 3 - O usuário seleciona o comando que deseja enviar para a TV. |
| 4 - O resultado da ação do comando enviado deve ser refletido na TV. |

|  |  |
| --- | --- |
| **Desligar os equipamentos de um cômodo** | |
| Descrição | Os equipamentos devem ser desligados caso não existam pessoas no ambiente ou não seja detectado movimentos. |
| Pré-condição | Os equipamentos devem estar ligados. Não pode haver pessoas no ambiente ou nem detectado movimento. |
| Pós-condição | Todos os equipamentos do ambiente devem estar desligados. |
| Fluxo Principal | 1 - O controle se encontra em algum cômodo identificado por Bluetooth. |
| 2 - O usuário muda de ambiente e o controle recebe um sinal informando que está em um novo cômodo. |
| 3 - Os equipamentos serão desligados, quando da ausência de pessoas no ambiente em que o usuário anteriormente se encontrava ou não detectado movimento. |

|  |  |
| --- | --- |
| **Desligar os equipamentos de um cômodo caso não seja detectado movimentos no ambiente** | |
| Descrição | Quando o usuário sair de um ambiente, os equipamentos do ambiente em que o usuário estava devem ser desligados se não for detectado movimento num intervalo de tempo de 15 segundos. Mesmo havendo pessoas no ambiente, se não for detectado movimento, o sistema desliga os aparelhos. |
| Pré-condição | Os equipamentos devem estar ligados. O controle deve estar em um outro cômodo e não pode ser detectado movimento por um intervalo de tempo X. |
| Pós-condição | Todos os equipamentos do ambiente devem estar desligados. |
| Fluxo Principal | 1 - O controle se encontra em algum cômodo identificado por Bluetooth. |
| 2 - O usuário muda de ambiente e o controle recebe um sinal informando que está em um novo cômodo. |
| 3 - O sistema verifica se há pessoas no ambiente. |
| 4 - O sistema verifica se ocorre movimentos por um intervalo de X minutos, caso não ocorra, desliga os aparelhos. |

|  |  |
| --- | --- |
| **Desligar os equipamentos de um cômodo caso não tenha pessoas no ambiente** | |
| Descrição | Quando o usuário sai de um ambiente, caso não tenha pessoas no ambiente deixado, todos os aparelhos devem ser desligados. |
| Pré-condição | Os equipamentos devem estar ligados. Não pode haver pessoas no ambiente. |
| Pós-condição | Todos os equipamentos do ambiente devem estar desligados. |
| Fluxo Principal | 1 - O controle está em algum cômodo identificado por Bluetooth. |
| 2 - O usuário muda de ambiente e o controle recebe um sinal informando que está num novo ambiente. |
| 3 - O sistema verifica que não há pessoas no ambiente deixado. |
| 4 - O sistema desliga os aparelhos deste ambiente. |

|  |  |
| --- | --- |
| **Ligar os equipamentos de um cômodo** | |
| Descrição | Ao chegar em um cômodo, os equipamentos devem ser ligados. |
| Pré-condição | Não pode existir pessoas no novo ambiente que o usuário ingressou. |
| Pós-condição | Os equipamentos devem estar ligados |
| Fluxo Principal | 1 - O usuário muda de cômodo. |
| 2 - Ao chegar no novo cômodo o sensor detecta a presença do usuário. |
| 3 - Os equipamentos são ligados, mantendo as configurações dos equipamentos do cômodo anterior. |

Tabela 3 - Tabelas de fluxo de eventos dos casos de uso

# 10. Processo de Desenvolvimento

Para o desenvolvimento do projeto foi utilizado a Metodologia SCRUM[[7]](#footnote-7) com *sprints* semanais. Apesar da utilização de um processo ágil, a documentação não ficou em segundo plano, tendo constantes revisões e atualizações do documento a cada duas semanas. Veja o Product Backlog e Sprint Backlog [aqui](https://docs.google.com/spreadsheet/ccc?key=0AvvvF4CtbWgSdDNXZGlnN1JwUDYwanBRd1k5NjNfU1E&usp=drive_web#gid=0).

A equipe de desenvolvimento foi formada por quatro desenvolvedores, um deles fazendo o papel de Scrum Master (Felipe Lindemberg).

## 10.1. Repositório

Durante o desenvolvimento foi utilizado o Google Code[[8]](#footnote-8) como repositório do sistema. Além disso, utilizamos o SVN[[9]](#footnote-9) para realizar o controle de versão, por ser um sistema open source, simples de usar e bastante eficiente para uma equipe de quatro pessoas.

O sistema possui licença GNU GPL v2[[10]](#footnote-10) e pode ser baixado através deste link: <https://code.google.com/p/controle-multimidia-universal/>

## 10.2. Integração Contínua

No processo de desenvolvimento do Controle Remoto Universal foi utilizado o Jenkins[[11]](#footnote-11) como servidor de integração contínua. O intuito de sua utilização foi minimizar a quantidade de possíveis erros encontrados. Por meio do Jenkins, todo *commit* de código pode ser testado automaticamente, além de possibilitar a verificação se as alterações realizadas pelo *commit* criaram defeitos no código já existente (Figuras 3 e 4).

Em virtude da inexistência de um servidor dedicado para o Jenkins decidiu-se criar um servidor local, em que foram agendados testes automatizados para serem executados a cada 5 minutos quando a máquina 02 do laboratório estiver ligada. Com isso, pôde-se ter um controle do desempenho da equipe e garantir um produto em total funcionamento.

Os testes programados para o Jenkins foram:

* **Testes de Unidade:** Para isso, foi utilizada uma biblioteca de Python chamada Unittest (nativa de Python), que permite criar classes de teste. Como o código desta aplicação depende de diversos fatores, tais como comunicação via rede e Bluetooth, a gama de testes de unidade ficou bastante reduzida. No entanto, foi explorado ao máximo a quantidade de testes possíveis.
* **Testes Estáticos de Código:** Por meio do Pylint[[12]](#footnote-12), verificou-se a estrutura do código, *imports* e regras do Pep8[[13]](#footnote-13) (padrão de desenvolvimento para Python).

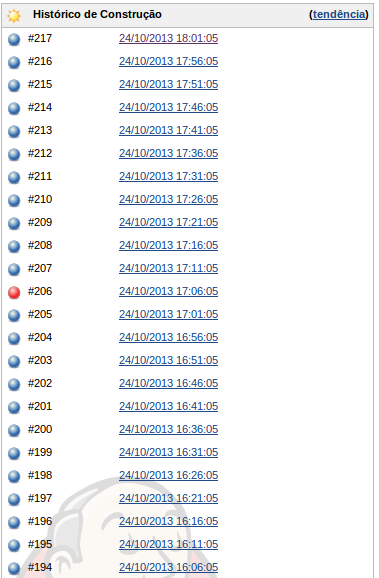




Figura 3 - Histórico de construções Figura 4 - Status do projeto no Jenkins

## 10.3. Revisão de Código

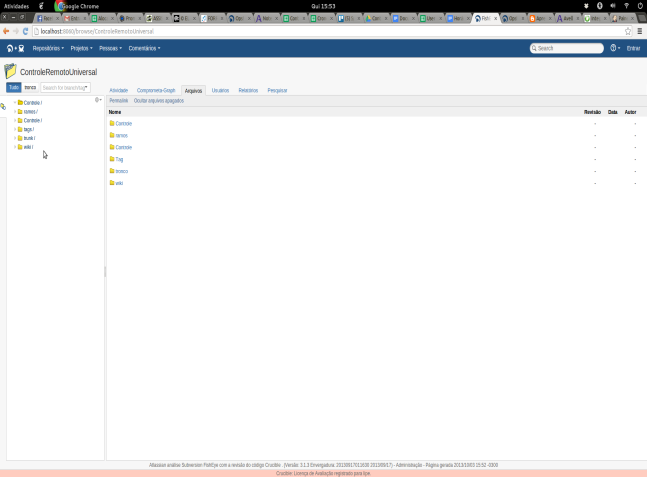
  
 Além do Jenkins, foi utilizado o Crucible[[14]](#footnote-14) como ferramenta para revisão de código. Em cada *commit* realizado pelo desenvolvedor, uma revisão foi criada e direcionada para o *Scrum Master* revisar. Porém, os outros integrantes do projeto participaram da revisão, informando erros de lógica, formas de melhorar o desempenho do sistema ou organizar o código. Isto teve o objetivo de assegurar que mais de um desenvolvedor julgue a eficiência do que foi desenvolvido.

Figura 5 - Raiz do projeto

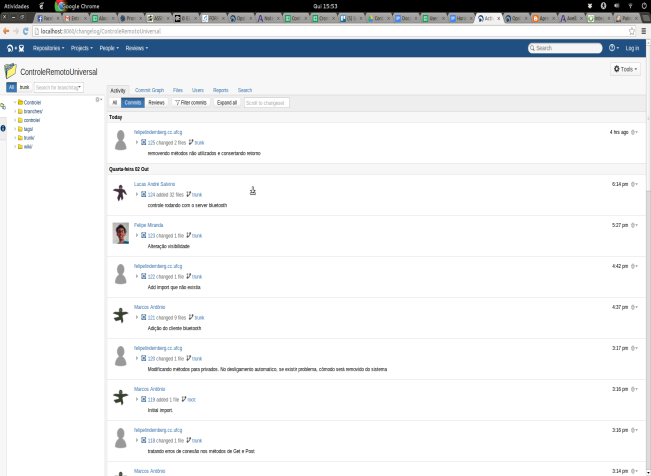


Figura 6 - Histórico de *commits*

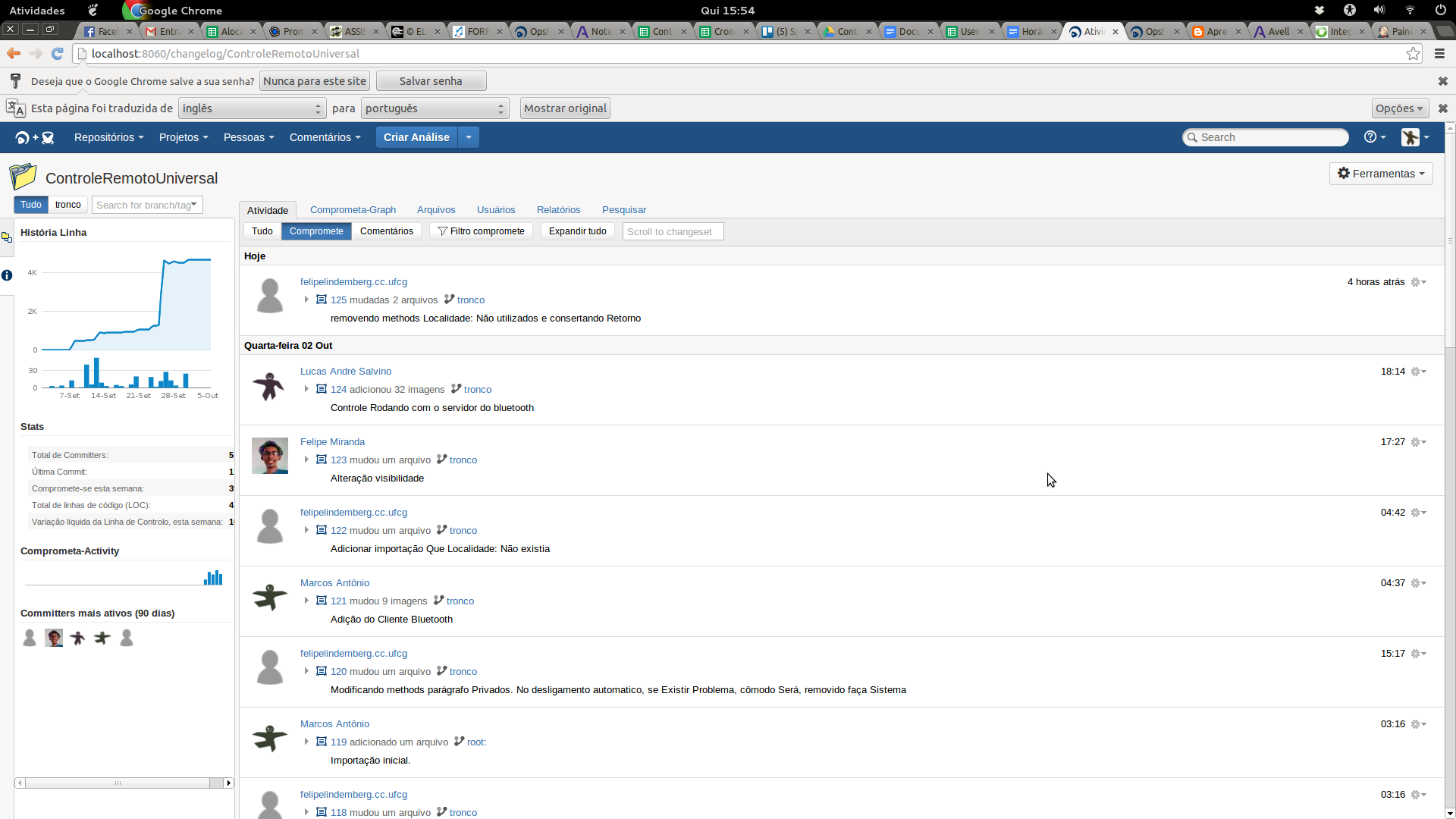


Figura 7 - Tela para criação de revisões

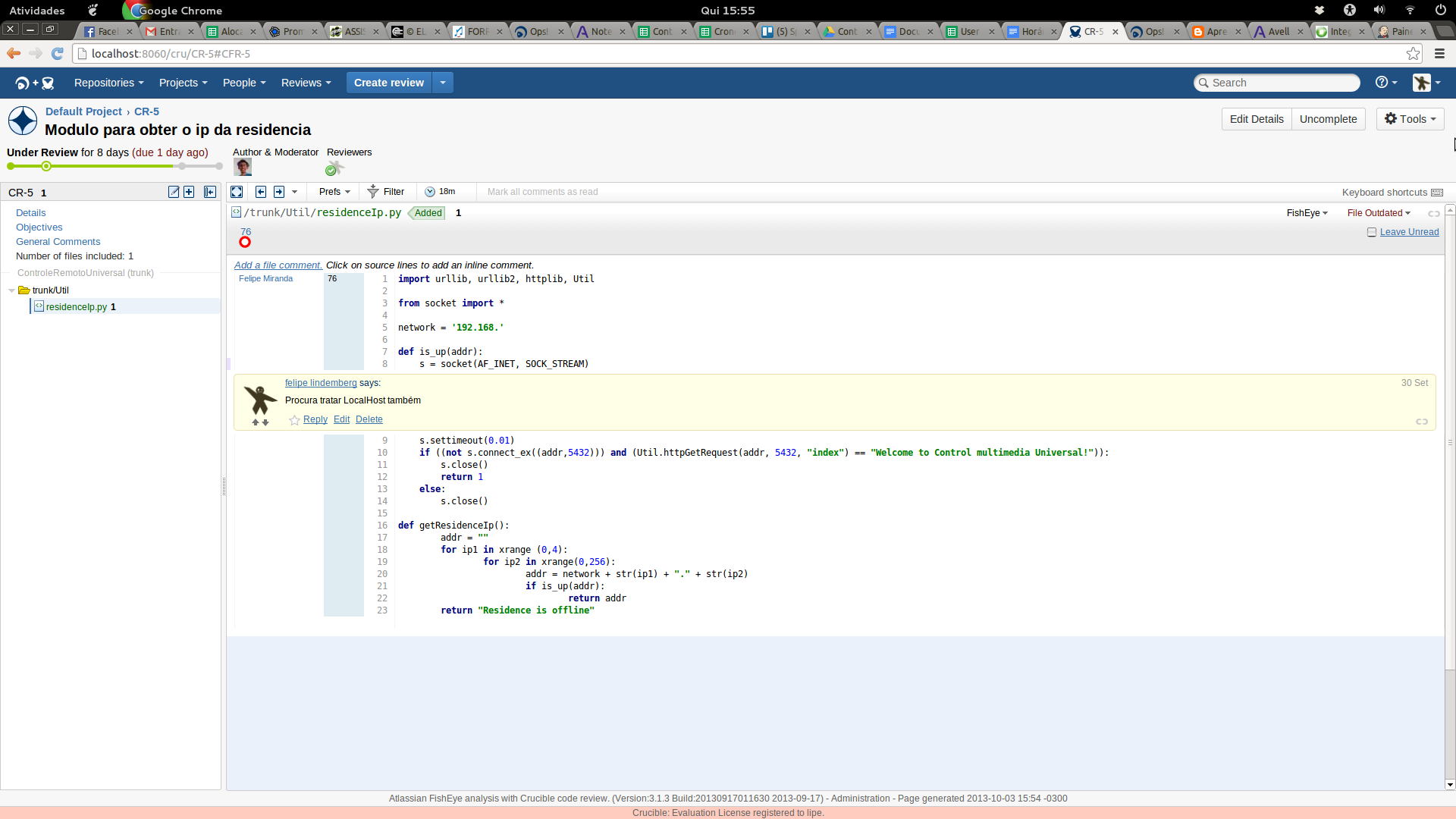


Figura 8 - Revisão feita em uma parte do código

Com o Crucible, o Scrum Master pode ter acesso a toda a estrutura do projeto (Figura 5) para verificar o histórico de *commits* (Figura 6) e criar revisões de partes do código (Figura 7 e 8).

## 10.4. Gerência de Riscos

|  |  |  |
| --- | --- | --- |
| **Classificação do Risco** | **Impacto / Descrição do Risco** | **Estratégia de diminuição e/ou plano de contingência** |
| Alto | Integrante da equipe ficar doente | O membro pode fazer as atividades de casa / repor horário ou outro membro da equipe faz a atividade. |
| Alto | Pouca experiência com uma tecnologia específica | Antes de iniciar a *sprint*, verificar as dificuldades de cada integrante e separar um tempo para estudo. |
| Alto | Estimativa errada para realização de uma atividade | Planejar melhor cada *sprint*, verificar as dificuldades de cada integrante e alocar mais integrante para finalizar a atividade. |
| Médio | Complexidade da interface gráfica | Dividir a atividade com mais membros da equipe e procurar formas alternativas |

Tabela 3 – Riscos durante o desenvolvimento do sistema

Durante o desenvolvimento ocorreram vários imprevistos relacionados a linguagem de desenvolvimento adotada, no caso Python. A versão atual da biblioteca PyGame[[15]](#footnote-15) utilizada para a criação da interface gráfica do controle foi alterada e muitas das suas funções padrões foram removidas. Além disso, ela possuía erros ao executar o áudio e o vídeo. Para contornar esses erros houve a tentativa de buscar bibliotecas alternativas, que também não trouxeram sucesso.

Para executar o vídeo, a biblioteca requer exclusividade no hardware de som, e com isso não era possível reproduzir o volume do SOM e o áudio da TV ao mesmo tempo. A solução foi não reproduzir o áudio do vídeo, já que por se tratar de um ambiente simulado, considerou-se a reprodução do vídeo mais importante do que o seu áudio.

# 11. Cronograma Inicial

|  |  |
| --- | --- |
| **Data** | **Atividades** |
| **02/09 à 13/09** | **Implementação Inicial + Reunião** |
|  | Reunião inicial (definir *Scrum Master*, criar *BugReport*) |
|  | Criar servidor |
|  | Criar cliente |
|  | Desenvolver interface inicial dos cômodos da sala (Servidor) |
|  | Desenvolver interface inicial do controle (Cliente) |
| **16/09 à 27/09** | **Testes + Implementação + Documentação** |
|  | Revisar documento e fazer alterações necessárias |
|  | Criar um plano de testes de usabilidade |
|  | Criar testes de unidade (se possível) |
|  | Implementar simulação dos sensores que detectam em qual cômodo o controle se encontra |
|  | Criar conexão Bluetooth entre os sensores e o controle |
|  | Resolução de *Bugs* |
| **30/09 à 11/10** | **GUI + Implementação** |
|  | Implementar lógica para detectar pessoas num ambiente |
|  | Implementar o desligamento automático dos aparelhos quando não houver pessoas no ambiente |
|  | Implementar a ligação automática de aparelhos de um cômodo |
|  | Implementar GUI para as novas funcionalidades |
|  | Resolução de *Bugs* |
| **14/10 à 25/10** | **Testes + Implementação + Documentação** |
|  | Implementar lógica para detectar movimentos num ambiente |
|  | Implementar o desligamento automático dos aparelhos quando não detectar movimentos |
|  | Manter configurações dos aparelhos ao trocar de cômodo |
|  | Criar plano de testes para as novas funcionalidades |
|  | Resolução de *Bugs* |
| **28/08 à 31/10** | **Revisar Documento + Testes + Revisar Produto** |
|  | Fazer mais testes de usabilidade |
|  | Revisar todas as funcionalidades do sistema |
|  | Resolução de BUGS |
|  | Atualizar documento para versão final (adicionar dificuldades enfrentadas, soluções, conclusão) |

Tabela 4 - Tabela de cronograma

# 12. Padrões de comandos e métodos dos servidores

## 12.1. Chamadas do Servidor Residência

**Descrição:** Método que exibe a mensagem de boas-vindas.

**Método:** index()

**Tipo:** GET

**Retorno:** String: {"Welcome to Control Multimedia Universal!"}

**Exemplo:**

method = "index"

retorno = Network.httpGetRequest("192.168.2.0", 5432, method)

**Descrição:** Método que adiciona um novo cômodo ao sistema (residência).

**Método:** addRoom(nameRoom, ip, port)

**Tipo:** POST

**Retorno:** String: {"Comodo adicionado", "Comodo já existe"}

**Exemplo:**

params = {"nameRoom": "sala", "ip": "192.168.2.6", "port": 8989}

method = "addRoom"

retorno = Network.httpPostRequest("192.168.2.0", 5432, method, params)

**Descrição:** Método que desconecta o controle do cômodo caso o controle não esteja presente no mesmo.

**Método:** disconnectControl(nameRoom)

**Tipo:** POST

**Retorno:** String: {"OK"}

**Exemplo:**

params = {"nameRoom": "quarto"}

method = "disconnectControl"

retorno = Network.httpPostRequest("192.168.2.0", 5432, method, params)

**Descrição:** Método que remove um novo cômodo do sistema (residência). Inicialmente criado para ser chamado quando um cômodo é encerrado.

**Método:** removeRoom(nameRoom)

**Tipo:** POST

**Retorno:** String: {"Comodo removido", "Comodo nao existe"}

**Exemplo:**

params = {"nameRoom": "quarto"}

method = "removeRoom"

retorno = Network.httpPostRequest("192.168.2.0", 5432, method, params)

**Descrição:** Método que envia comandos ("command”) para o aparelho ("equipment") de um cômodo específico ("nameRoom").

**Método:** sendCommand(nameRoom, equipment, command)

**Tipo:** POST

**Retorno:** String: {"ok", "Comodo nao existe"}

**Exemplo:**

**Comandos para equipamentos específicos (TV, SOM):**

**POWER:** Liga ou Desliga o aparelho selecionado

**UPVOLUME:** Aumenta o volume

**DOWNVOLUME:** Diminui o volume do aparelho

**UPCHANNEL:** Muda de canal progressivamente (TV) ou aumenta a frequência (rádio)

**DOWNCHANNEL:** Muda de canal regressivamente (TV) ou diminui a frequência (rádio)

params = {"nameRoom": "quarto", "equipment": "tv", "command": "downcanal"}

method = "sendCommand"

retorno = Network.httpPostRequest("192.168.2.0", 5432, method, params)

**Comandos para todos os equipamentos de um cômodo:**

**ALL:** Comando aplicado a todos os equipamentos

**POWEROFF:** Desliga todos os aparelhos do cômodo

params = {"nameRoom": "quarto", "equipment": "all", "command": "poweroff"}

method = "sendCommand"

retorno = Network.httpPostRequest("192.168.2.0", 5432, method, params)

**Descrição:** Altera no sistema o cômodo onde o controle se encontra.

**Método:** setRoomOfControl(nameRoom)

**Tipo:** POST

**Retorno:** String: {"ok", "Comodo nao existe”, “Local alterado"}

**Exemplo:**

params = {"nameRoom": "quarto"}

method = "setRoomOfControl"

retorno = Network.httpPostRequest("192.168.2.0", 5432, method, params)

**Descrição:** Acessa no sistema o cômodo onde o controle se encontra.

**Método:** getRoomOfControl()

**Tipo:** GET

**Retorno:** String: {"nameRoom": nomeComodo}

**Exemplo:**

method = “getRoomOfControl”

retorno = Network.httpGetRequest("192.168.2.0", 5432, method)

**Descrição:** Retorna os cômodos que foram adicionados no sistema.

**Método:** getRooms()

**Tipo:** GET

**Retorno:** Json: {“nomeComodo": {“ipComodo", portaComodo}}

**Exemplo:**

method = "getRooms"

retorno = Network.httpGetRequest("192.168.2.0", 5432, method)

## 12.2. Chamadas do Servidor Cômodo

**Descrição:** Método acessador do status da conexão com residência.

**Método:** getConnected()

**Tipo:** GET

**Retorno:** Booleano: {"connected": Boolean}

**Exemplo:**

method = "getConnected"

retorno = Network.httpGetRequest("192.168.2.0", 5432, method)

**Descrição:** Método modificador da presença do controle no cômodo.

**Método:** controlIsFound(isFound)

**Tipo:** POST

**Retorno:** String: {"OK"}

**Exemplo:**

params = {“isFound”: "True"}

method = "controlIsFound"

retorno = Network.httpPostRequest("192.168.2.0", 5432, method, params)

**Descrição:** Método que faz a conexão de cômodo com a residência.

**Método:** Connect()

**Tipo:** GET

**Retorno:** None

**Exemplo:**

method = "Connect"

retorno = Network.httpGetRequest(“192.168.2.0”, 5432, method)

**Descrição:** Método que **r**etorna o nome do cômodo.

**Método:** index()

**Tipo:** GET

**Retorno:** String: {"Cômodo - nameRoom"}

**Exemplo:**

method = "index"

retorno = Network.httpGetRequest("192.168.2.0", 5432, method)

**Descrição:** Método quecontrola o equipamento do cômodo.

**Método:** controlEquipment(equipment, command)

**Tipo:** POST

**Retorno:** String: {"ok", "Aparelho não esta ligado"}

**Exemplo:**

params = {"equipment": "tv", "command": "power"}

method = "controlEquipment”

retorno = Network.httpPostRequest("192.168.2.0", 5432, method, params)

**Descrição:** Método quemodifica a configuração do aparelho do cômodo.

**Método:** setNewConfiguration(configuration)

**Tipo:** POST

**Retorno:** None

**Exemplo:**

params = {"configuration": "{"tv": {"state": "Boolean", "volume": "Inteiro", "channel": "Inteiro"}, "sound": {"state": "Boolean", "volume": "Inteiro", "channel": "Inteiro"}}"

method = "setNewConfiguration"

retorno = Network.httpPostRequest(“192.168.2.0”, 5432, method, params)

**Descrição:** Método quemodifica o número de pessoas presentes no cômodo.

**Método:** setNumberOfPeoples(number)

**Tipo:** POST

**Retorno:** String: {"Add"}

**Exemplo:**

params = {"number": 5}

method = "setNumberOfPeoples"

retorno = Network.httpPostRequest("192.168.2.0", 5432, method, params)

**Descrição:** Método que retorna o número de pessoas presentes no cômodo.

**Método:** getNumberOfPeoples()

**Tipo:** GET

**Retorno:** String: {"number": "5"}

**Exemplo:**

method = "getNumberOfPeoples"

retorno = Network.httpGetRequest("192.168.2.0", 5432, method)

**Descrição:** Método que finaliza a comunicação de residência com o cômodo.

**Método:** exit()

**Tipo:** GET

**Retorno:** None

**Exemplo:**

method = "exit"

retorno = Network.httpGetRequest("192.168.2.0", 5432, method)

**Descrição:** Método queretorna os comandos que foram enviados aos aparelhos do cômodo.

**Método:** getRoomConfiguration()

**Tipo:** GET

**Retorno:** Json: {“tv": {"status": Boolean, "channel": Inteiro, "volume": Inteiro}, "som": {"status": Boolean, "channel": Inteiro, "volume": Inteiro}}

**Exemplo:**

method = "getRoomConfiguration“

retorno = Network.httpGetRequest("192.168.2.0", 5432, method)

## 12.3. Comandos enviados aos equipamentos

**Comandos e funções referentes aos equipamentos presentes em Cômodo**

**Equipamento:** TV

**Função:** Permite enviar comandos para controlar especificamente o aparelho multimídia TV.

**Equipamento:** SOM

**Função:** Permite enviar comandos para controlar especificamente o aparelho multimídia Som.

**Equipamento:** ALL

**Função:** Permite enviar comandos para controlar todos os aparelhos de multimídia presentes no cômodo. Neste caso, TV e Som.

**Comandos e funções para equipamentos multimídia do tipo TV ou SOM**

**Comando:** POWER

**Função:** Ligar caso o aparelho multimídia esteja desligado ou desligar caso o aparelho multimídia esteja ligado. Os demais métodos só serão possíveis caso o aparelho multimídia que se deseja controlar esteja ligado.

**Comando:** UPCHANNEL

**Função:** Incrementa o canal do aparelho multimídia em um.

**Comando:** DOWNCHANNEL

**Função:** Decrementa o canal do aparelho multimídia em um.

**Comando:** UPVOLUME

**Função:** Incrementa o volume do aparelho multimídia em um.

**Comando:** DOWNVOLUME

**Função:** Decrementa o volume do aparelho multimídia em um.

**Comando:** MUTE

**Função:** Coloca o volume do aparelho multimídia no modo MUDO.

**Comandos e funções para equipamentos multimídia do tipo ALL**

**Comando:** POWEROFF

**Função:** Desliga todos os aparelhos multimídia presentes no cômodo, caso os mesmos estejam ligados.

**Comando:** MUTE

**Função:** Coloca o volume de todos os aparelhos multimídia presentes no cômodo no modo MUDO.

# 13. Prototipação de Telas

No início do projeto, a prioridade de desenvolvimento foram as funcionalidades do produto de software. Assim, foram criados os servidores para as classes cômodo e residência e a lógica para simular os equipamentos multimídia. Com isso a interface gráfica era a mais simples possível, como se pode observar nas Figuras 9 e 10.

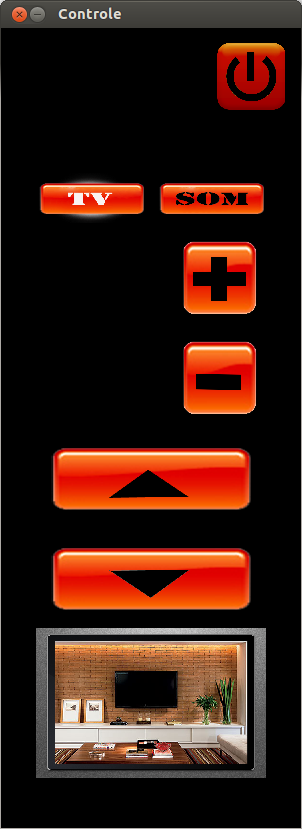
 

Figura 9 - Versão inicial do cômodo Figura 10 - Controle com funcionalidades básicas

Após implementação total das funcionalidades, as interfaces gráficas do ambiente simulado e do controle foram evoluídas, conforme apresentado nas Figuras 11 e 12.

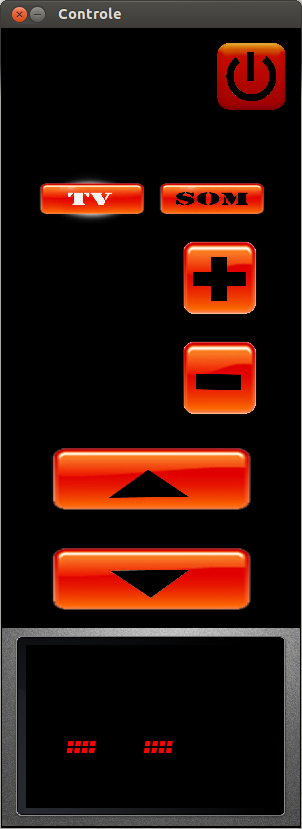
 

Figura 11 - Cômodo com o número de pessoas no ambiente Figura 12 – Controle exibindo local em que se encontra

Por fim, foi criada uma interface gráfica simples para a execução de algumas outras funcionalidades do sistema. Além disso, foi adicionada uma nova funcionalidade no controle (MUTE) e as interfaces gráficas de cômodo e controle foram melhoradas.

Também foram adicionados botões que identificam se o dispositivo Bluetooth está conectado no computador, que permite simular a troca de cômodo.

Figura 13 – Novo controle Figura 14 – Identifica controle no cômodo e permite adicionar pessoas

# 14. Testes

Testes de usabilidade da aplicação foram realizados, de forma a verificar as funcionalidades que fazem uso das conexões de rede e Bluetooth para funcionar. Para isso, foi criada uma planilha com um plano de testes detalhado informando o passo-a-passo para realizar determinada atividade e o resultado esperado para cada uma delas.

## 14.1. Pep8 e Pylint

Pep8 é uma convenção que define regras para o desenvolvimento de códigos eficientes e organizados em Python. Para o Controle Remoto Universal, algumas das convenções do Pep8 são ignoradas por não serem relevantes no *design* do código e nem implicarem no entendimento do mesmo.

Para verificar se o código segue as regras do Pep8 utilizou-se uma biblioteca de Python chamada Pylint, que faz a análise estática do código. Pylint faz mais do que verificar as convenções do Pep8, ela assegura a qualidade do projeto procurando problemas nos códigos e gerando relatórios. Variáveis não utilizadas, *imports* não utilizados, ausência de *imports*, possíveis *bad blocks* são alertados pelo Pylint.

Todos os erros ignorados pelo Controle Multimídia Universal estão descritos no [**Anexo A**.](#_16.1._Pep_8)

Julga-se que muitos desses testes são inviáveis para o projeto, como por exemplo, limitar 79 caracteres por linha de código, uma vez que hoje em dia qualquer monitor é capaz de mostrar mais de 80 caracteres por linha. Outra restrição removida foi o teste de linhas similares, pois para métodos do servidor, existem muitas linhas similares que tem funcionalidades diferentes. Contudo foram mantidos muitos testes, como verificação dos *imports* não utilizados, de espaçamentos entre funções e declarações de classes, de linhas em brancos. Quando estas convenções são seguidas à risca, melhoram o desempenho da aplicação e facilita a leitura do código.

## 14.2. Testes de Unidade

Testes de unidade são muito importantes no processo de desenvolvimento de um software, pois garante o funcionamento correto das funções do sistema após as diversas modificações do código.

No Controle Remoto Universal, apesar das principais funções necessitarem de servidores rodando e serem bem mais complexas de testar, foi contemplado testes de unidades. As principais funções dos objetos controláveis, de cômodo e de residência foram contempladas com testes utilizando a biblioteca Python Unittest[[16]](#footnote-16). Esses testes são executados tanto pelos desenvolvedores durante o processo de desenvolvimento como também pelo Jenkins no processo de integração contínua para garantir que sempre o último *commit* do repositório esteja funcionando corretamente.



Figura 15 - Testes de unidade executados pelo Jenkins

## 14.3. Testes de Usabilidade

Um Teste de Usabilidade tem por objetivo verificar a facilidade que o software possui de ser claramente compreendido e manipulado pelo usuário” (WIKIPEDIA, 2013)[[17]](#footnote-17).

Esses testes também permitem verificar o funcionamento do produto em condições reais e são bem úteis para prover *feedback* aos desenvolvedores no que pode ser melhorado e no que deve ser corrigido.

Para isso, criou-se uma planilha de testes que mostra um passo-a-passo do que deve ser feito e de como o sistema deve se comportar.

**Teste de usabilidade**

|  |  |
| --- | --- |
| **TESTE – Abrir Tutorial de uso** | |
| **Modo de execução** | **Exemplo** |
| 1.Inicie o run.py  2. Clique em “ajuda” no canto superior esquerdo a tela  3. Clique em tutorial |  |
| **Resultado esperado** |
| Após esses passos, o aplicativo deve exibir um arquivo PDF em uma nova janela. Nesse arquivos estão descritos todos os passos necessários para utilizar de maneira correta o Controle Multimídia Universal. |

**Teste de usabilidade**

|  |  |
| --- | --- |
| **TESTE – Abrir créditos do Controle Multimídia Universal** | |
| **Modo de execução** | **Exemplo** |
| 1.Inicie o run.py  2. Clique em “ajuda” no canto superior esquerdo a tela  3. Clique em “Sobre” |  |
| **Resultado esperado** |
| Após esses passos, o aplicativo deve exibir uma nova janela com uma descrição sucinta do Controle Multimídia Universal. |

|  |  |
| --- | --- |
| **TESTE – Criar duas residências na mesma rede** | |
| **Modo de execução** | **Exemplo** |
| 1. Execute o run.py em uma janela do “explorer” do Ubuntu[[18]](#footnote-18) e clique em “Iniciar Residência”.  2. Em outra janela do “explorer” do Ubuntu clique em run.py.  3. Clique em “Iniciar residência” |  |
| **Resultado esperado** |
| Não deve ser permitido criar uma nova residência, pois já existe uma rodando na rede. O botão de criar residência irá informar que a residência já foi iniciada na rede. |

**Teste de usabilidade**

|  |  |
| --- | --- |
| **TESTE – Interface do Menu: Pré inicialização do sistema** | |
| **Modo de execução** | **Exemplo** |
| 1. Inicie o run.py |  |
| **Resultado esperado** |
| Após esse passo, o aplicativo deve exibir um menu com quatro botões. São eles: **Iniciar residência**, **Iniciar controle**, **Iniciar cômodo** e **Sobre**. |

|  |  |
| --- | --- |
| **TESTE – Execução do controle com a residência offline** | |
| **Modo de execução** | **Exemplo** |
| 1. Certifique-se que a residência está offline  2. Inicie o run.py  3. Clique em “Iniciar controle” |  |
| **Resultado esperado** |
| Após esses passos, o aplicativo irá pesquisar a residência na rede e como ela está offline será exibida uma tela informando o problema e após 5 segundos a tela é fechada. |

|  |  |
| --- | --- |
| **TESTE – Execução do controle com a residência online** | |
| **Modo de execução** | **Exemplo** |
| 1. Certifique-se que a residência está online  2. Inicie o run.py  3. Clique em “Iniciar controle” |  |
| **Resultado esperado** |
| Após esses passos, o aplicativo irá pesquisar a residência na rede e como ela está online, o controle será iniciado normalmente e exibirá no visor central um “load” de espera de conexão Bluetooth. |

**Teste de usabilidade**

|  |  |
| --- | --- |
| **TESTE – Criar residência quando não existe residência na rede** | |
| **Modo de execução** | **Exemplo** |
| 1. Inicie o run.py  2. Clique em “Iniciar residência” |  |
| **Resultado esperado** |
| Após clicar em “Iniciar Residência”, o sistema irá pesquisar na rede se já existe uma residência rodando e caso não tenha irá criar a residência. |

**Teste de usabilidade**

|  |  |
| --- | --- |
| **TESTE – Criação de cômodo** | |
| **Modo de execução** | **Exemplo** |
| 1. Inicie o run.py  2. Digite o nome do cômodo (o nome deve ter mais de 3 dígitos) no campo de texto ao lado esquerdo do botão “Iniciar cômodo”.  3. Clique em “Iniciar cômodo” |  |
| **Resultado esperado** |
| Ao tentar criar um cômodo o mesmo vai pesquisar a residência na rede e exibir uma mensagem com o nome “Conectando”. Caso a conexão seja feita com sucesso o cômodo será exibido e se não encontrar a janela será fechada automaticamente. |

**Teste de usabilidade**

|  |  |
| --- | --- |
| **TESTE – Cômodo identificar se existe dispositivo Bluetooth no computador** | |
| **Modo de execução** | **Exemplo** |
| 1. Execute o run.py.  2. Digite um nome para um cômodo  3. Clique em “Iniciar cômodo”  4. Conecte e desconecte o dispositivo Bluetooth na USB do computador. |  |
| **Resultado esperado** |
| Ao conectar o Bluetooth no computador o botão Bluetooth de cômodo ficará azul e permitirá clique e ao desconectar ficará cinza e não irá permitir cliques. |

|  |  |
| --- | --- |
| **TESTE – Alterar a quantidade de pessoas de um cômodo** | |
| **Modo de execução** | **Exemplo** |
| 1. Execute o run.py.  2. Inicie a residência  3. Inicie o cômodo  4. Clique nos botões “+” e “-“ do Cômodo. |  |
| **Resultado esperado** |
| Ao clicar no botão “+” a quantidade de pessoas será aumentada, caso clique no “-“ a quantidade será diminuída. Caso o controle esteja no cômodo, a quantidade de pessoas será incrementada em um e o número mínimo de pessoas nesse cômodo também será um. |

**Teste de usabilidade**

|  |  |
| --- | --- |
| **TESTE – Presença / troca de cômodos** | |
| **Modo de execução** | **Exemplo** |
| *Teste deve ser feito em 2 computadores com Bluetooth*  1. Execute o run.py.  2. Crie uma residência  3. Crie um cômodo com o nome “sala” e outro com o nome “quarto”  4. Execute o controle em outra máquina  5. Identifique na interface do cômodo que o Bluetooth está conectado  6. Clique no botão Bluetooth da “sala” e veja o controle atualizar o nome do cômodo.  7. Faça o passo 6 para o “quarto” |  |
| **Resultado esperado** |
| Ao clicar no botão Bluetooth de um cômodo, o seu nome deve ser enviado para o controle e na interface do cômodo, deve ser exibido que o controle se encontra nele. Ao trocar de cômodo, não é permitido que o controle esteja em 2 cômodos ao mesmo tempo. |

**Teste de usabilidade**

|  |  |
| --- | --- |
| **TESTE – Encerrar controle** | |
| **Modo de execução** | **Exemplo** |
| *Teste deve ser feito em 2 computadores com Bluetooth*  1. Execute o run.py.  2. Crie uma residência  3. Crie um cômodo com o nome “sala”  4. Execute o controle em outra máquina  5. Identifique na interface do cômodo que o Bluetooth está conectado  6. Clique no botão Bluetooth da “sala” e veja o controle atualizar o nome do cômodo.  7. Encerre o controle com “ESC” ou fechando |  |
| **Resultado esperado** |
| Ao clicar no botão Bluetooth, a interface do cômodo deve ser atualizada para mostrar que o controle se encontra nele. Ao encerrar o controle, deve-se atualizar novamente informando que o controle não está mais. |

**Teste de usabilidade**

|  |  |
| --- | --- |
| **TESTE – Selecionar qual equipamento será controlado** | |
| **Modo de execução** | **Exemplo** |
| 1. Inicie o run.py  2. Inicie o controle  3. Pressione o botão “tv”  4. Pressione o botão “som”  5. Pressione os botões “tv” e “som” repetidamente e alternadamente |  |
| **Resultado esperado** |
| Ao clicar no botão “tv” o mesmo ficará em destaque e qualquer outro botão que for pressionado posteriormente controlará a TV do cômodo em que se encontra. Ao clicar no botão “som”, o foco de controle passará para o “SOM” do cômodo e todos os comandos serão enviados para ele. Não será possível selecionar os dois equipamentos ao mesmo tempo. |

**Teste de usabilidade**

|  |  |
| --- | --- |
| **TESTE – Clicar em botões do controle quando o mesmo não sabe em qual cômodo se encontra.** | |
| **Modo de execução** | **Exemplo** |
| 1. Execute o run.py.  2. Crie uma residência  3. Crie um cômodo  4. Crie um controle e veja que o seu visor exibe “- - -“  5. Pressione os botões do controle |  |
| **Resultado esperado** |
| Após esses passos verifique que os equipamentos do cômodo não vão ser controlados. |

|  |  |
| --- | --- |
| **TESTE – Ligar/Desligar equipamento** | |
| **Modo de execução** | **Exemplo** |
| 1. Execute o run.py.  2. Crie uma residência  3. Crie um cômodo  4. Crie um controle  5. Clique no botão Bluetooth do cômodo.  6. Selecione um equipamento e clique no botão power. |  |
| **Resultado esperado** |
| Se o equipamento estiver desligado, ele deverá ligar. Caso esteja ligado, o equipamento deve desligar. |

**Teste de usabilidade**

|  |  |
| --- | --- |
| **TESTE – Mudar canal/Controlar volume de um equipamento** | |
| **Modo de execução** | **Exemplo** |
| 1. Execute o run.py.  2. Crie uma residência  3. Crie um cômodo  4. Crie um controle  5. Clique no botão Bluetooth do cômodo.  6. Selecione um equipamento e ligue-o.  7. Clique nos botões de volume e canal. |  |
| **Resultado esperado** |
| Controlar volume não deve interferir em nada, apenas no volume. Mudar canal não deve interferir no volume, apenas no canal. |

**Teste de usabilidade**

|  |  |
| --- | --- |
| **TESTE – Desligar/Ligar equipamentos automaticamente** | |
| **Modo de execução** | **Exemplo** |
| Teste deve ser feito em 2 computadores com Bluetooth  1. Execute o run.py.  2. Crie uma residência  3. Crie um cômodo com o nome “sala” e outro com o nome “quarto”  4. Execute o controle em outra máquina  5. Clique no botão Bluetooth da “sala” e veja o controle atualizar o nome do cômodo.  7. Ligue os equipamentos da sala em qualquer canal e volume  8. Clique no Bluetooth do “quarto” para simular que você foi para o quarto. |  |
| **Resultado esperado** |
| Ao realizar esses procedimentos, os equipamentos da sala devem ser desligados e os equipamentos do quarto devem ligar automaticamente no mesmo canal e volume do cômodo anterior. |

**Teste de usabilidade**

|  |  |
| --- | --- |
| **TESTE – Desligamento automático de equipamentos quando não há pessoas no cômodo** | |
| **Modo de execução** | **Exemplo** |
| Teste deve ser feito em 2 computadores com Bluetooth  1. Execute o run.py.  2. Crie uma residência  3. Crie um cômodo com o nome “sala”  4. Execute o controle em outra máquina  5. Clique no botão Bluetooth da “sala” e veja o controle atualizar o nome do cômodo.  7. Ligue os equipamentos da sala em qualquer canal e volume  8. Pressione ESC no controle para encerrar o mesmo |  |
| **Resultado esperado** |
| Ao encerrar o controle, o usuário não está mais em cômodo nenhum. O sistema, de 15 em 15 segundos, verifica se há pessoas no cômodo e caso não tenha, desliga todos os equipamentos. Então, ao sair do controle, os equipamentos da sala devem ser desligados. |

|  |  |
| --- | --- |
| **TESTE – Mute nos equipamentos** | |
| **Modo de execução** | **Exemplo** |
| Teste deve ser feito em 2 computadores com Bluetooth  1. Execute o run.py.  2. Crie uma residência  3. Crie um cômodo com o nome “sala”  4. Execute o controle em outra máquina  5. Clique no botão Bluetooth da “sala” e veja o controle atualizar o nome do cômodo.  7. Ligue a TV da sala no volume 15  8. Pressione Mute várias vezes |  |
| **Resultado esperado** |

**Teste de usabilidade**

|  |  |
| --- | --- |
| **TESTE – Removendo um cômodo do sistema** | |
| **Modo de execução** | **Exemplo** |
| Teste deve ser feito em 2 computadores com Bluetooth  1. Execute o run.py.  2. Crie uma residência  3. Crie um cômodo com o nome “sala” e outro com o nome “quarto”  4. Execute o controle em outra máquina  5. Clique no botão Bluetooth da “sala” e veja o controle atualizar o nome do cômodo.  7. Controle a sala  8. Feche o cômodo da sala |  |
| **Resultado esperado** |
| Mesmo o controle exibindo que se encontra na sala, ele não vai mais controlar os equipamentos da mesma. Além disso, não vai ser possível controlar os equipamentos de outro cômodo até que o controle vá para um cômodo controlado. |

|  |  |
| --- | --- |
| **TESTE – Criar cômodos com o mesmo nome** | |
| **Modo de execução** | **Exemplo** |
| 1. Execute o run.py.  2. Crie uma residência  3. Crie um cômodo com o nome “sala”  4. Crie outro cômodo com o nome “sala” |  |
| **Resultado esperado** |
| Não será possível criar 2 cômodos com o mesmo nome. Será mostrado uma mensagem informando que o cômodo já existe. |

**Teste de usabilidade**

|  |  |
| --- | --- |
| **TESTE – Finalizar o Controle Multimídia Universal** | |
| **Modo de execução** | **Exemplo** |
| 1. Inicie o run.py  2. Inicie a residência  3. Inicie o Controle  4. Inicie o Cômodo  2. Finalize o run.py (botão fechar) |  |
| **Resultado esperado** |
| Após esses passos, todos os processos interligados ao Controle Multimídia universal serão finalizados. |

## 14.3.1 Análise de cobertura dos testes de usabilidade

Foram criados diversos testes de usabilidade, e para cada teste foi descrito as funções que o mesmo executa. Com isso, temos uma quantidade estimada de funções do software que foram chamadas e funcionam corretamente. Através desses dados pode-se obter diversas estatísticas sobre cobertura dos testes.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **NOME DO TESTE** | **Passa em X funções do pacote Cômodo** | **Passa em X funções do pacote Residência** | **Passa em X funções do pacote Controle** | **Passa em X funções do pacote** **Util** | **Interface** | **Total de funções que o teste executa** |
| **Criar duas residências na mesma rede** | **0** | **0** | **0** | **1** | **2** | **3** |
| **Interface do Menu: Pré inicialização do sistema** | **1** | **1** | **0** | **0** | **0** | **2** |
| **Execução do controle com a residência offline** | **0** | **0** | **0** | **2** | **2** | **4** |
| **Execução do controle com a residência online** | **0** | **0** | **17** | **0** | **2** | **19** |
| **Criação de cômodo** | **18** | **2** | **0** | **2** | **4** | **26** |
| **Criar residência quando não existe residência na rede** | **0** | **3** | **0** | **0** | **4** | **7** |
| **Cômodo identificar se existe dispositivo Bluetooth no computador** | **39** | **2** | **0** | **2** | **4** | **47** |
| **Presença / troca de cômodos** | **22** | **7** | **43** | **2** | **8** | **82** |
| **Alterar a quantidade de pessoas de um cômodo** | **40** | **2** | **0** | **2** | **4** | **48** |
| **Encerrar controle** | **5** | **6** | **3** | **3** | **0** | **17** |
| **Selecionar qual equipamento será controlado** | **0** | **0** | **2** | **1** | **0** | **3** |
| **Ligar/Desligar equipamento** | **5** | **2** | **5** | **2** | **0** | **14** |
| **Clicar em botões do controle quando o mesmo não sabe em qual cômodo se encontra.** | **0** | **0** | **2** | **0** | **0** | **2** |
| **Mudar canal/Controlar volume de um equipamento** | **14** | **2** | **16** | **0** | **0** | **32** |
| **Desligar/Ligar equipamentos automaticamente** | **1** | **2** | **0** | **1** | **0** | **4** |
| **Desligamento automático de equipamentos quando não há pessoas no cômodo** | **2** | **0** | **0** | **0** | **0** | **2** |
| ***Mute* nos equipamentos** | **1** | **1** | **3** | **0** | **0** | **4** |
| **Abrir tutorial** | **0** | **0** | **0** | **0** | **1** | **1** |
| **Abrir informativos sobre o controle** | **0** | **0** | **0** | **0** | **1** | **1** |
| **Finalizar o Controle Multimídia Universal** | **0** | **0** | **0** | **0** | **2** | **2** |

Tabela 5 – Número de funções testadas executadas em cada teste

## 14.3.2 Resumo das Estatísticas Gerais dos Testes de Usabilidade

*Número de testes de usabilidade:* 19

*Número total de funções do projeto:* 143

*Número de funções testadas:* 139

*Número de funções não testadas:* 4

*Cobertura de testes:* 97,2%

# 15. *Bugs*

Para reportar os *bugs*, criou-se uma planilha onde a equipe poderá adicionar os *bugs* e se possível atribuir um dono ao *bug*. A planilha tem campos para detalhar o problema, classificar o *bug* (simples, importante, crítico), explicar uma forma de reproduzi-lo, atribuir um dono para resolução do problema.

Todos os *bugs* possuem um estado (fechado ou aberto) e para destacar a relevância de cada *bug* foi utilizada cores (vermelho, amarelo, azul), facilitando assim a visualização dos seus estado (vide Tabela 6).

|  |  |  |
| --- | --- | --- |
| **BUG: O servidor não funciona** | | |
| Status | Dono | Relevância |
| Aberto | Felipe | Crítica |
| Descrição |  |  |
| Reproduzir Bug |  |  |
|  | | |
| **BUG: O servidor não funciona** | | |
| Status | Dono | Relevância |
| Aberto | Marcos | Importante |
| Descrição |  |  |
| Reproduzir Bug |  |  |
|  | | |
| **BUG: O servidor não funciona** | | |
| Status | Dono | Relevância |
| Fechado | Felipe | Simples |
| Descrição |  |  |
| Reproduzir Bug |  |  |

Tabela 6 - Tabela para registrar os bugs

[Clique aqui para visualizar os bugs do Controle Multimida Universal!](https://docs.google.com/spreadsheet/ccc?key=0ArbaYflCukPadHdJejFvckRWc1BwRlRDcnBOV3R1S1E&usp=drive_web#gid=0)

# 16. Anexos

## A - Pep 8 - Python style guide checker

Segue a lista de erros que são ignorados no Pylint para o Controle Remoto Universal:

1. [**C0102**](http://pylint-messages.wikidot.com/messages:c0102)**:** Black listed name "%s"
2. [**C0103**](http://pylint-messages.wikidot.com/messages:c0103)**:** Invalid name "%s" (should match %s)
3. [**C0111**](http://pylint-messages.wikidot.com/messages:c0111)**:** Missing docstring
4. [**C0112**](http://pylint-messages.wikidot.com/messages:c0112)**:** Empty docstring
5. [**C0121**](http://pylint-messages.wikidot.com/messages:c0121)**:** Missing required attribute "%s"
6. [**C0202**](http://pylint-messages.wikidot.com/messages:c0202)**:** Class method should have "cls" as first argument
7. [**C0203**](http://pylint-messages.wikidot.com/messages:c0203)**:** Metaclass method should have "mcs" as first argument
8. [**C0301**](http://pylint-messages.wikidot.com/messages:c0301)**:** Line too long (%s/%s)
9. [**C0302**](http://pylint-messages.wikidot.com/messages:c0302)**:** Too many lines in module (%s)
10. [**C0321**](http://pylint-messages.wikidot.com/messages:c0321)**:** More than one statement on a single line
11. [**C0322**](http://pylint-messages.wikidot.com/messages:c0322)**:** Operator not preceded by a space
12. [**C0323**](http://pylint-messages.wikidot.com/messages:c0323)**:** Operator not followed by a space
13. [**C0324**](http://pylint-messages.wikidot.com/messages:c0324)**:** Comma not followed by a space
14. [**R0001**](http://pylint-messages.wikidot.com/messages:r0001)**:** Messages by category
15. [**R0002**](http://pylint-messages.wikidot.com/messages:r0002)**:** % errors / warnings by module
16. [**R0003**](http://pylint-messages.wikidot.com/messages:r0003)**:** Messages
17. [**R0004**](http://pylint-messages.wikidot.com/messages:r0004)**:** Global evaluation
18. [**R0101**](http://pylint-messages.wikidot.com/messages:r0101)**:** Statistics by type
19. [**R0201**](http://pylint-messages.wikidot.com/messages:r0201)**:** Method could be a function
20. [**R0401**](http://pylint-messages.wikidot.com/messages:r0401)**:** Cyclic import (%s)
21. [**R0401**](http://pylint-messages.wikidot.com/messages:r0401)**:** External dependencies
22. [**R0402**](http://pylint-messages.wikidot.com/messages:r0402)**:** Modules dependencies graph
23. [**R0701**](http://pylint-messages.wikidot.com/messages:r0701)**:** Raw metrics
24. [**R0801**](http://pylint-messages.wikidot.com/messages:r0801)**:** Duplication
25. [**R0801**](http://pylint-messages.wikidot.com/messages:r0801)**:** Similar lines in %s files
26. [**R0901**](http://pylint-messages.wikidot.com/messages:r0901)**:** Too many ancestors (%s/%s)
27. [**R0902**](http://pylint-messages.wikidot.com/messages:r0902)**:** Too many instance attributes (%s/%s)
28. [**R0903**](http://pylint-messages.wikidot.com/messages:r0903)**:** Too few public methods (%s/%s)
29. [**R0904**](http://pylint-messages.wikidot.com/messages:r0904)**:** Too many public methods (%s/%s)
30. [**R0911**](http://pylint-messages.wikidot.com/messages:r0911)**:** Too many return statements (%s/%s)
31. [**R0912**](http://pylint-messages.wikidot.com/messages:r0912)**:** Too many branches (%s/%s)
32. [**R0913**](http://pylint-messages.wikidot.com/messages:r0913)**:** Too many arguments (%s/%s)
33. [**R0914**](http://pylint-messages.wikidot.com/messages:r0914)**:** Too many local variables (%s/%s)
34. [**R0915**](http://pylint-messages.wikidot.com/messages:r0915)**:** Too many statements (%s/%s)
35. [**R0921**](http://pylint-messages.wikidot.com/messages:r0921)**:** Abstract class not referenced
36. [**R0922**](http://pylint-messages.wikidot.com/messages:r0922)**:** Abstract class is only referenced %s times
37. [**R0923**](http://pylint-messages.wikidot.com/messages:r0923)**:** Interface not implemented
38. **F0401:** Unable to import %r (%s)
39. **E1121:** Too many positional arguments for function call
40. **E0213:** Method should have "self" as first argument
41. **W0232:** Class has no \_\_init\_\_ method
42. **W0614:** Unused import %s from wildcard import
43. **W0403:** Relative import %r
44. **W0401:** Wildcard import %s
45. **W0212:** Access to a protected member %s of a client class

1. [http://www.bluetooth.com](http://www.bluetooth.com/) [↑](#footnote-ref-1)
2. <http://www.upnp.org/> [↑](#footnote-ref-2)
3. <http://www.oodesign.com/observer-pattern.html> [↑](#footnote-ref-3)
4. <https://developer.bluetooth.org/TechnologyOverview/Pages/RFCOMM.aspx> [↑](#footnote-ref-4)
5. <http://technet.microsoft.com/pt-BR/library/cc786900(v=ws.10).aspx> [↑](#footnote-ref-5)
6. <http://www.python.org> [↑](#footnote-ref-6)
7. Scrum – <http://desenvolvimentoagil.com.br/scrum/> [↑](#footnote-ref-7)
8. Google Code - <https://code.google.com/intl/pt-BR/> [↑](#footnote-ref-8)
9. SVN - <http://subversion.tigris.org/> [↑](#footnote-ref-9)
10. GNU GPL V2 - <http://www.gnu.org/licenses/old-licenses/gpl-2.0.html> [↑](#footnote-ref-10)
11. Jenkins – <http://www.jenkins-ci.org/> [↑](#footnote-ref-11)
12. Pylint - <http://www.pylint.org/> [↑](#footnote-ref-12)
13. Pep8 – <https://pypi.python.org.pypi/pep8> [↑](#footnote-ref-13)
14. Crucible – <https://www.atlassian.com/software/crucible/> [↑](#footnote-ref-14)
15. Pygame - <http://www.python.org.br/wiki/PyGame> [↑](#footnote-ref-15)
16. Python Unittest - <http://docs.python.org/2/library/unittest.html> [↑](#footnote-ref-16)
17. **Teste de usabilidade**. Disponível em: <http://pt.wikipedia.org/wiki/Teste\_de\_usabilidade>. Acesso em: 23 Out. 2013. [↑](#footnote-ref-17)
18. Ubuntu - <http://www.ubuntu-br.org/> [↑](#footnote-ref-18)