



UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE CIÊNCIAS EXATAS E DA TERRA
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO



Roomie: uma aplicação para prédios inteligentes baseada na infraestrutura de Internet das Coisas

Viviane Costa Pinheiro

Natal-RN
Maio 2017

Viviane Costa Pinheiro

**Roomie: uma aplicação para prédios inteligentes
baseada na infraestrutura de Internet das Coisas**

Monografia de Graduação apresentada ao
Departamento de Informática e Matemática
Aplicada do Centro de Ciências Exatas e
da Terra da Universidade Federal do Rio
Grande do Norte como requisito parcial para
a obtenção do grau de bacharel em Ciência
da Computação.

Orientador(a)

Prof. Me. Itamir de Moraes Barroca Filho

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE – UFRN
DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA – DIMAP

Natal-RN

Maio 2017

Monografia de Graduação sob o título *Roomie: uma aplicação para prédios inteligentes baseada na infraestrutura de Internet das Coisas* apresentada por Nome do aluno e aceita pelo Departamento de Informática e Matemática Aplicada do Centro de Ciências Exatas e da Terra da Universidade Federal do Rio Grande do Norte, sendo aprovada por todos os membros da banca examinadora abaixo especificada:

Titulação e nome do(a) orientador(a)

Orientador(a)

Departamento

Universidade

Titulação e nome do membro da banca examinadora

Co-orientador(a), se houver

Departamento

Universidade

Titulação e nome do membro da banca examinadora

Departamento

Universidade

Titulação e nome do membro da banca examinadora

Departamento

Universidade

Natal-RN, data de aprovação (por extenso).

Homenagem que o autor presta a uma ou mais pessoas.

Roomie: uma aplicação para prédios inteligentes baseada na infraestrutura de Internet das Coisas

Autor: Viviane Costa Pinheiro

Orientador(a): Prof. Me. Itamir de Barrocas Filho

RESUMO

Com os avanços tecnológicos dos últimos anos e a popularização da Internet das Coisas (IoT), hoje em dia é possível desenvolver novas aplicações que resolvem velhos problemas do nosso cotidiano que antigamente não possuíam uma solução viável. Uma das áreas que está sendo bastante explorada nesse novo contexto são as aplicações específicas para prédios inteligentes. Dentro dos problemas de gerenciamento de prédios, um dos problemas mais recorrentes é o mau uso dos espaços, em especial das salas de reunião. No entanto os softwares tradicionais utilizados para reservas de salas não conseguem resolver todas as dificuldades relacionadas a gerência desses espaços, principalmente por não se adaptarem ao ambiente no qual estão inseridos e ao comportamento dos usuários em tempo real. Este trabalho tem como principal objetivo descrever o desenvolvimento de uma aplicação voltada para reserva de salas de reunião, a aplicação utiliza infraestrutura de IoT para a incorporação de dados de sensores na aplicação de reserva para permitir a adaptação do sistema ao comportamento dos usuários e estados das salas. *Palavras-chave:* Internet das Coisas, Prédios Inteligentes, Sistemas de Reserva de Sala.

Listas de figuras

1	Raspberry Pi 3, Modelo B	p. 14
2	Leitor de Cartão RFID RC522	p. 15
3	Tag RFID de padrão Mifare	p. 15
4	Sensor PIR	p. 16
5	Visão Geral do sistema	p. 19
6	Diagrama de Implementação	p. 24
7	Diagrama de Casos de Uso	p. 26
8	Diagrama de Classes Roomie	p. 28
9	BPM de Fluxo de reserva do Sistema Roomie	p. 38
10	Arquitetura Kaa	p. 39
11	Arquitetura Kaa	p. 39
12	Código de inicialização cliente Kaa	p. 40
13	Código de anexação do usuário	p. 40
14	Código de registro de recebimento de eventos	p. 40
15	Código de envio de eventos	p. 41
16	Diagrama de Classes RoomieRaspberry	p. 41
17	Mapeamento dos pinos do Raspberry Pi 2 e 3	p. 42
18	Diagrama de classes RoomieWeb	p. 43
19	Tela de Login RoomieWeb	p. 44
20	Tela de Login RoomieWeb	p. 45
21	Listagem de reuniões do usuário	p. 46
22	Diagrama de Classes RoomieController	p. 47

Lista de abreviaturas e siglas

IoT – Internet of Things

GPIO – General Purpose Input/Output

PIR – Passive-Infrared

Sumário

1	Introdução	p. 10
1.1	Definição do Trabalho	p. 10
1.2	Objetivos	p. 11
1.3	Organização do Trabalho	p. 12
2	Fundamentação Teórica	p. 13
2.1	Java	p. 13
2.2	Raspberry Pi	p. 13
2.3	Sensor RFID MFRC522	p. 14
2.4	Sensor PIR	p. 14
2.5	Kaa	p. 16
2.6	Pi4J	p. 16
2.7	MySQL	p. 17
2.8	Spring MVC	p. 17
3	Trabalhos Relacionados	p. 18
3.1	Soluções para prédios inteligentes	p. 18
3.1.1	A Smart Room Scheduling and Management System with Utilization Control and Ad-hoc Support Based on Real-Time Occupancy Detection	p. 18
3.1.2	iSense: A Wireless Sensor Network Based Conference Room Management System	p. 19
3.2	Comparativo entre soluções	p. 20

4 Rommie: uma aplicação para prédios inteligentes	p. 23
4.1 Concepção e Design da Aplicação	p. 23
4.1.1 Arquitetura da Aplicação	p. 23
4.1.2 Requisitos da Aplicação	p. 25
4.1.2.1 Requisitos Funcionais	p. 25
4.1.2.2 Modelagem de Casos de Uso	p. 26
4.1.2.3 Requisitos Não Funcionais	p. 26
4.1.3 Diagrama de Classes	p. 28
4.1.4 Modelo de Processo de Negócio	p. 28
4.2 Implementação e Testes da Aplicação	p. 30
4.2.1 Desenvolvimento da aplicação no Middleware Kaa	p. 30
4.2.1.1 Configuração da Instalação do Kaa	p. 30
4.2.1.2 Visão Geral da Arquitetura e funcionamento do Kaa	p. 30
4.2.1.3 Eventos	p. 31
4.2.1.4 Códigos Conexão Kaa - Java	p. 32
4.2.2 Desenvolvimento da aplicação RoomieRaspberry	p. 34
4.2.2.1 Diagrama de Classes	p. 34
4.2.2.2 Configuração dos sensores no Raspberry Pi	p. 34
4.2.3 Desenvolvimento do RoomieWeb	p. 35
4.2.3.1 Diagrama de Classes	p. 35
4.2.4 Principais Telas da Aplicação	p. 35
4.2.4.1 Desenvolvimento do RoomieController	p. 36
4.2.5 Execução Final da Aplicação	p. 36
5 Considerações finais	p. 48
5.1 Trabalhos Futuros	p. 48

Referências	p. 49
Apêndice A – Primeiro apêndice	p. 51
Anexo A – Primeiro anexo	p. 52

1 Introdução

No panorama atual da Computação, um novo paradigma que surge com cada vez mais força nos últimos anos, é a Internet das Coisas ou em inglês, Internet of Things(IoT). A IoT pode ser definida como uma rede mundial de objetos interconectados e unicamente endereçados, baseado em protocolos tradicionais de comunicação (ATZORI, 2010). Esse novo paradigma traz um novo tipo de relacionamento que antes era baseado apenas em homem-máquina e passa também a ser máquina-máquina (TAN; WANG, 2010). A IoT traz uma possibilidade de dezenas de novas aplicações para o futuro, focando nas mais diversas áreas, como: transportes, logística, ambientes inteligentes, saúde e aplicação pessoais entre outras.

Uma das aplicações que nos últimos anos tem ganhado cada vez mais força graças ao IoT, são as aplicações para smart buildings ou prédios inteligentes. Uma das definições fornecidas para Smart Building pelo (HARRIS, 2012) , é que um smart building é um prédio que possui características mutáveis e possui a habilidade de se adaptar a mudanças em ambientes externos e internos a fim de endereçar problemas de conforto, financeiro e de consumo de energia. Grande parte das aplicações de smart building se concentram em proporcionar um gerenciamento mais inteligente dos espaços físicos dos prédios, proporcionando redução do consumo de energia e também uma melhor adaptação do prédio a condições externas como iluminação e temperatura.

1.1 Definição do Trabalho

Um dos grandes problemas em grandes prédios comerciais hoje em dia é a má utilização dos espaços, em especial das salas de reunião. E apesar de já existirem diversos sistemas que abordam o tema do controle de espaços, poucos possuem a habilidade de serem sensíveis ao ambiente em que estão inseridos, ou seja, poucos capturam dados e informações do ambiente que estão inseridos para uma tomada futura de decisão. A falta dessa característica implica na não detecção de alguns problemas comuns na utilização

desses espaços. Os problemas comumente não detectados são:

- *No-show.* Acontece quando a sala é reservada, no entanto a reunião acaba sendo cancelada mas a reserva não. Esse comportamento não é tão incomum quanto parece, pois aproximadamente 40% das reservas acabam em no-show (DUNCAN, 2016).
- Reservas maiores que reuniões. Um comportamento comum dos usuários que utilizam sistemas de reservas de salas é reservar um tempo maior que o necessário, para que não corram riscos de ficar sem sala. No entanto normalmente a reunião dura menos tempo que o reservado mas sala continua com o status de ocupada no sistema, pois não é possível a identificação do fim da reunião (TRAN et al., 2016).
- O “roubo” de salas. Essa situação ocorre quando alguns usuários desconsideram as reservas previamente feitas para a sala e utilizam o espaço sem ter realizado alguma reserva. Como normalmente não existe um mecanismo de autenticação para identificar o usuário que realizou a tarefa essa é uma situação que acaba sendo bastante comum.

Esses problemas de mau uso e inadequação dos sistemas a realidade dos prédios, acaba saindo caro para as empresas, pois serão necessárias um número muito superior de salas de reunião para suprir essa lacuna de salas sendo usadas inefficientemente.

1.2 Objetivos

Este trabalho tem o objetivo geral descrever o desenvolvimento de uma aplicação de Smart Building utilizando a infraestrutura da Internet das Coisas, aplicação é denominada Roomie. Os objetivos específicos que querem ser alcançados ao final do texto são:

- Relatar o desenvolvimento de uma aplicação baseada na arquitetura de IoT, desde sua documentação até sua implementação;
- Realizar uma comparação das tecnologias atuais que possam ser usadas para desenvolver esse tipo de aplicação.
- Identificar as possíveis dificuldades e limitações fornecidas pelas ferramentas atuais.

1.3 Organização do Trabalho

O trabalho é organizado em seções, que estão organizadas da seguinte maneira:

- Na seção 2 são apresentadas as tecnologias que foram utilizadas no desenvolvimento da aplicação,
- Na seção 3 temos comparação com outras aplicações que tem finalidade similares e abordam o tema de controle de espaços em prédios.
- Na seção 4 é descrita o desenvolvimento da ferramenta, requisitos, arquitetura. implementação de testes da aplicação.
- A seção 5, retrata as considerações finais do trabalho e abre espaço para discussão sobre possíveis trabalhos futuros baseados no presente trabalho de conclusão.

2 Fundamentação Teórica

Para o desenvolvimento da aplicação Roomie diversas tecnologias e sistemas de hardware diferentes foram utilizados para promover escalabilidade, uma maior facilidade no desenvolvimento e fácil acoplação de novas funcionalidades para futuras aplicações. Nas próximas subseções são descritas em mais detalhes as tecnologias e os dispositivos utilizados no desenvolvimento da aplicação.

2.1 Java

A tecnologia Java é a base para praticamente todos os tipos de aplicações em rede hoje em dia sendo um padrão global para o desenvolvimento e distribuição de aplicações móveis, jogos, conteúdo baseado na Web e softwares corporativos. Atualmente existe uma comunidade grande de desenvolvedores do Java, mais de 9 milhões de pessoas em todo mundo. Uma das características mais importantes do Java é a possibilidade de reuso de aplicações portáteis de alto desempenho nos mais diversos dispositivos, ou seja, o software que é feito em uma plataforma pode ser executado em diversas outras plataformas com pouco esforço, permitindo-se assim fornecer mais serviços a um custo mais baixo tanto para empresa quanto para o consumidor final (ORACLE, 2016).

2.2 Raspberry Pi

O Raspberry Pi é um computador de propósito geral com o tamanho de um cartão de crédito, normalmente usado com o sistema operacional Linux e que possui a habilidade de rodar múltiplos programas (JUSTO, 2016).

Além da sua função como um computador, o Raspberry Pi também lida com computação física, que no caso do Pi, pode ser descrito como a interface entre o dispositivo e o ambiente exterior. Através dessa conexão o Pi pode controlar e monitorar elementos

externos que estejam conectados eletronicamente a ele. A interface de conexão entre o Pi e esses dispositivos externos é feita através da fileira pinos de propósito geral de entrada/saída ou em inglês General Purpose Input/Output(GIPO) que se encontram na parte superior na placa. A versão do Raspberry Pi utilizada nesse projeto é a versão 3, modelo B, mostrado na figura 1.

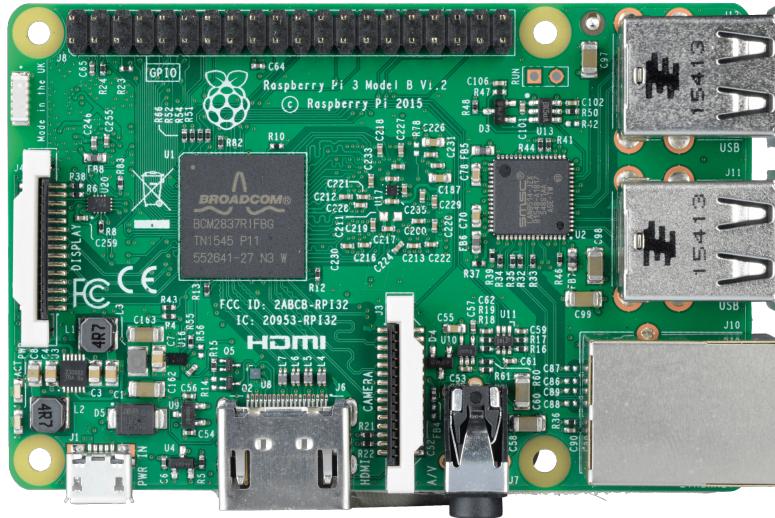


Figura 1: Raspberry Pi 3, Modelo B.

Fonte: (XBIAN COMMUNITY, 2017)

2.3 Sensor RFID MFRC522

O MFRC522 mostrado na figura 2 é um leitor com poder de leitura e escritas para comunicação sem necessidade de contato que opera em uma frequência de 13.56 MHz. O leitor se comunica com cartões que possuem os padrões ISO/IEC 14443 e A/MIFARE cards. O MFRC522 permite a comunicação através das interfaces (NXP SEMICONDUCTORS, 2017)

Neste trabalho é utilizada a interface SPI para a comunicação com o leitor, e são utilizados cartões e tags RFID com padrão Mifare como as mostradas na figura 3.

2.4 Sensor PIR

O sensor PIR (mostrado na figura 4) é utilizado comumente para a identificação de pessoas em um ambiente no raio de alcance do sensor. A sigla PIR vem do inglês Passive-Infrared e é chamado dessa maneira por utilizar sensores piroelétricos para detectar níveis



Figura 2: Leitor de Cartão RFID RC522

Fonte: (BOX ELECTRÓNICA, 2017)



Figura 3: Tag RFID de padrão Mifare

Fonte: (BOX ELECTRÓNICA, 2017)



Figura 4: Sensor PIR

Fonte: (EASY ELECTRONYX, 2017)

de radiação infravermelha. Esse tipo de sensor não identifica a distância em que o corpo está do sensor, apenas se há algum corpo no ambiente. O alcance do sensor é de aproximadamente 6 metros e ângulo de $110^\circ \times 70^\circ$ (ADAFRUIT, 2017).

2.5 Kaa

O Kaa é uma plataforma aberta de Middleware de IoT, que permite a construção de aplicações completas desse paradigma, além de fornecer outras funcionalidades embutidas importantes para o desenvolvimento desse tipo de aplicação. Segundo KAAIOT TECHNOLOGIES (2016) "O Kaa permite um gerenciamento de dados para os objetos conectados e a sua infraestrutura através de componentes para o servidor e SDKs para os endpoints. Os SDKs são embutidos nos objetos conectados e implementam troca de dados bi-direcional com o servidor." O Kaa possibilita o uso de aplicações de larga-escala em IoT, promovendo os serviços de: comunicação entre objetos conectados, consistência de dados, segurança, interoperabilidade e uma conectividade a prova de erros (KAAIOT TECHNOLOGIES, 2016).

2.6 Pi4J

O Pi4J Project é um projeto idealizado para fornecer uma interface orientada a objetos mais amigável para programadores Java utilizarem os recursos de entrada e saída do Raspberry Pi. Através da abstração de funções de baixo nível o Pi4J proporciona essa maior facilidade de uso de E/S (PI4J, 2017).

2.7 MySQL

Segundo Oracle([citeyearoracle-mysql](#)) MySQL é o sistema de gerenciamento de banco de dados aberto mais popular atualmente. É mantido pela Oracle Corporation. O MySQL é lida com banco de dados relacional, comumente chamado de SQL que é a sigla para Structured Query Language (Linguagem Estruturada de Consultas). O MySQL foi inicialmente criado para lidar grande quantidades de dados de uma forma mais rápida que as ferramentas existentes naquela mesma época. Mas também acabou se popularizando no desenvolvimento de pequenas aplicações uma vez que pode ser executado tanto em computadores pessoais como em servidores-web, que possuem mais recursos dedicados (Oracle, 2017).

2.8 Spring MVC

O Spring MVC é uma implementação do framework Spring para aplicações web baseado no modelo arquitetural model-view-controller. O Framework Spring é uma plataforma Java que permite o desenvolvimento de aplicações com suporte focado para a infraestrutura da aplicação, o framework auxilia permitindo um desacoplamento das diferentes partes da aplicação. O modelo MVC está focado na divisão das camadas de uma aplicação em visualização(*view*) (que são os elementos que o usuário interage), modelo(*model*) que é a representação em código dos objetos do mundo real e controles(*controllers*), que faz a lógica da aplicação e permite a comunicação dos dois anteriores. O Spring MVC permite o desacoplamento das camadas que compõem uma aplicação web e oferece as funcionalidades do framework Spring (JOHNSON et al., 2017).

3 Trabalhos Relacionados

3.1 Soluções para prédios inteligentes

3.1.1 A Smart Room Scheduling and Management System with Utilization Control and Ad-hoc Support Based on Real-Time Occupancy Detection

Previamente ao desenvolvimento da aplicação Roomie foram pesquisados outros trabalhos científicos e softwares que possuíam foco no tema de controle de espaços físicos em prédios. Essas referências possibilitaram a aquisição de um maior conhecimento sobre o problema e o desenvolvimento de uma aplicação que tentasse resolver problemas que as soluções anteriores não resolviam.

O principal objetivo do trabalho de TRAN et al. (2016) é a construção de um sistema inteligente de reserva de salas que endereça o problema de subutilização de salas de reunião. Essa subutilização segundo o autor é causada principalmente por reuniões que não acontecem mas possuem reservas no sistema e por sistemas que não suportam disponibilidade em tempo real de novas reservas. A solução proposta para combater o problema reportado é através de uma aplicação que permite a ciência sobre status de ocupação das salas em tempo-real e que promove a integração dessa informação com sistema de reserva.

O sistema descrito no trabalho é composto por sensores de movimento (PIR), cada sala possui um sensor conectado a um sistema embarcado que por sua vez possui uma conexão ethernet. Os sensores são utilizados para detecção de presença e os sinais captados são enviados via protocolo UDP(User Datagram Protocol) pelo sistema embarcado para um servidor que controla as reservas e que disponibiliza uma interface para os usuários gerenciarem suas reservas. A organização do sistema é ilustrada na figura 5.

Principais funcionalidades do sistema:

- Cancelamento de reserva automático por não detecção de presença;

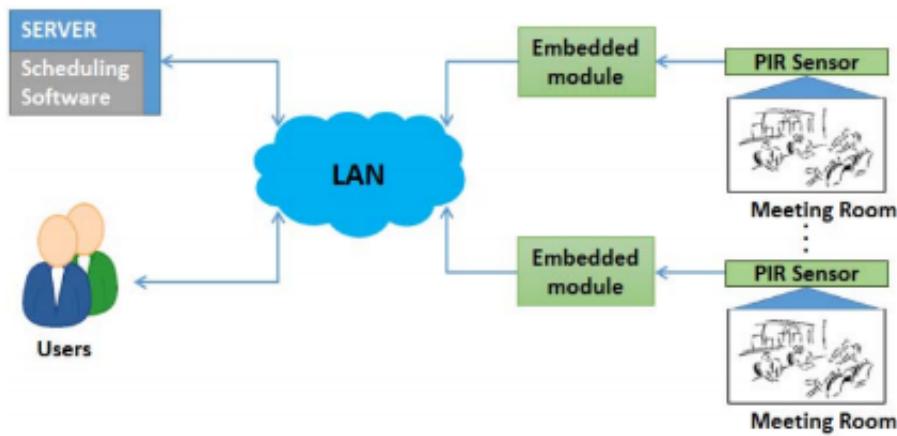


Figura 5: Visão Geral do sistema

Fonte: (TRAN et al., 2016)

- Gerenciamento de reservas em uma Interface Web própria;
- Fornecimento da informação em tempo real sobre a disponibilidade das salas;

3.1.2 iSense: A Wireless Sensor Network Based Conference Room Management System

O trabalho de PADMANABH et al. (2009) é em um sistema de reservas de salas que permite a detecção de presença e status dos aparelhos eletrônicos utilizados na sala. O foco do estudo é o melhor uso dos espaços através de espaços com maior consciência sobre sua ocupação e sobre a utilização de aparelhos eletrônicos na sala. O sistema descrito no trabalho (chamado de iSense) é composto de um microcontrolador com sensores (sensor de presença, temperatura e luminosidade) conectados a ele, um transceptor sem fio, um gateway, uma aplicação de controle e um banco de registro de dados. Todos os dados coletados dos sensores são enviados para um gateway conectado a um computador e interpretados por uma aplicação de controle. Os sensores são utilizados usados para detecção de pessoas (microfone e PIR sensor) e os outros sensores (luz e temperatura) são utilizados para a medição do consumo de energia. O funcionamento do iSense segue da seguinte maneira:

1. Usuário solicita através do iSense uma reserva;
2. iSense checa se a sala no horário estabelecido está livre no MS Outlook, caso esteja confirma a reserva senão fornece a opção do usuário entrar em uma fila para a reserva específica;

3. O iSense monitora o uso das salas através dos sensores, caso uma sala for indicada não ocupada por 5 minutos após o tempo inicial, o dono da reserva recebe uma mensagem requisitando o cancelamento da reserva.
4. Caso o usuário cancele a reserva, a próxima reserva que estava na lista de espera é alocada para o espaço da reserva cancelada.

O iSense foi utilizado no campus da empresa que os autores trabalhavam e obtiveram como resultado final, um aumento de 27Principais funcionalidades do iSense:

- Sistema de fila para reservas;
- Conexão com Microsoft Outlook para efetuar reservas;
- Disponibilização das informações sobre os aparelhos eletrônicos ligados na sala (ar-condicionado e luzes).
- Sistema de detecção de presença;

3.2 Comparativo entre soluções

Na tabela 3.2 é feita uma comparação entre as soluções encontradas na literatura descritas na seção 3.1, baseado nas funcionalidades que os sistemas disponibilizam. E abaixo segue uma legenda para a tabela 3.2.

1. A Smart Room Scheduling and Management System with Utilization Control and Ad-hoc Support Based on Real-Time Occupancy Detection
2. iSense: A Wireless Sensor Network Based Conference Room Management System

Tabela 1: Comparativo de soluções de reserva de salas

Funcionalidade	1	2
Detecção de Presença	Sim	Sim
Status de ocupação de salas em tempo real	Sim	Não
Autenticação na entrada da sala	Não	Não
Sistema de Gerenciamento de Reservas Próprio	Sim	Não
Simplicidade no acoplamento de novos sensores	Não	Não
Status dos equipment eletrônicos	Não	Sim
Cancelamento automático de reservas por Inatividade	Sim	Não

S

4 Rommie: uma aplicação para prédios inteligentes

Poucos exemplos de aplicações que lidam com o tema de reserva de espaços em prédios foram encontrados na literatura, os exemplos mais próximos encontrados foram os trabalhos citados na seção 3. Esses trabalhos possuem focos diferentes e atendem a alguns requisitos similares como exemplificado no quadro 01, mas ambos não suportam autenticação para o uso da salas e não dão suporte a uma maneira simples e fácil de acoplamento de novos sensores e dispositivos. Em consequência dessas funcionalidades não serem encontradas nas soluções anteriores, foi realizado o desenvolvimento da aplicação Roomie, que foi focada principalmente na incorporação do IoT. Nas próximas seções será descrito a aplicação Roomie em termos de arquitetura, requisitos e desenvolvimento das aplicações.

4.1 Concepção e Design da Aplicação

4.1.1 Arquitetura da Aplicação

O sistema Roomie é composto por 3 diferentes aplicações. Cada uma com seu propósito específico. Sendo a primeira uma aplicação Web que tem como principal objetivo fornecer uma interface amigável para os usuários gerenciarem suas reservas; a segunda é a aplicação que roda no Raspberry Pi que é focada em trazer informações vindas de sensores para compor o sistema, e a terceira a chamada aplicação de controle que tem como principal objetivo ajudar na coordenação de todo o sistema Roomie. A arquitetura é dividida na estrutura apresentada na Figura 6.

Abaixo segue uma descrição detalhada da arquitetura, onde os números romanos representam os dispositivos e as letras os softwares ou aplicações que rodam nesses dispositivos.

1. Desktop Ubuntu

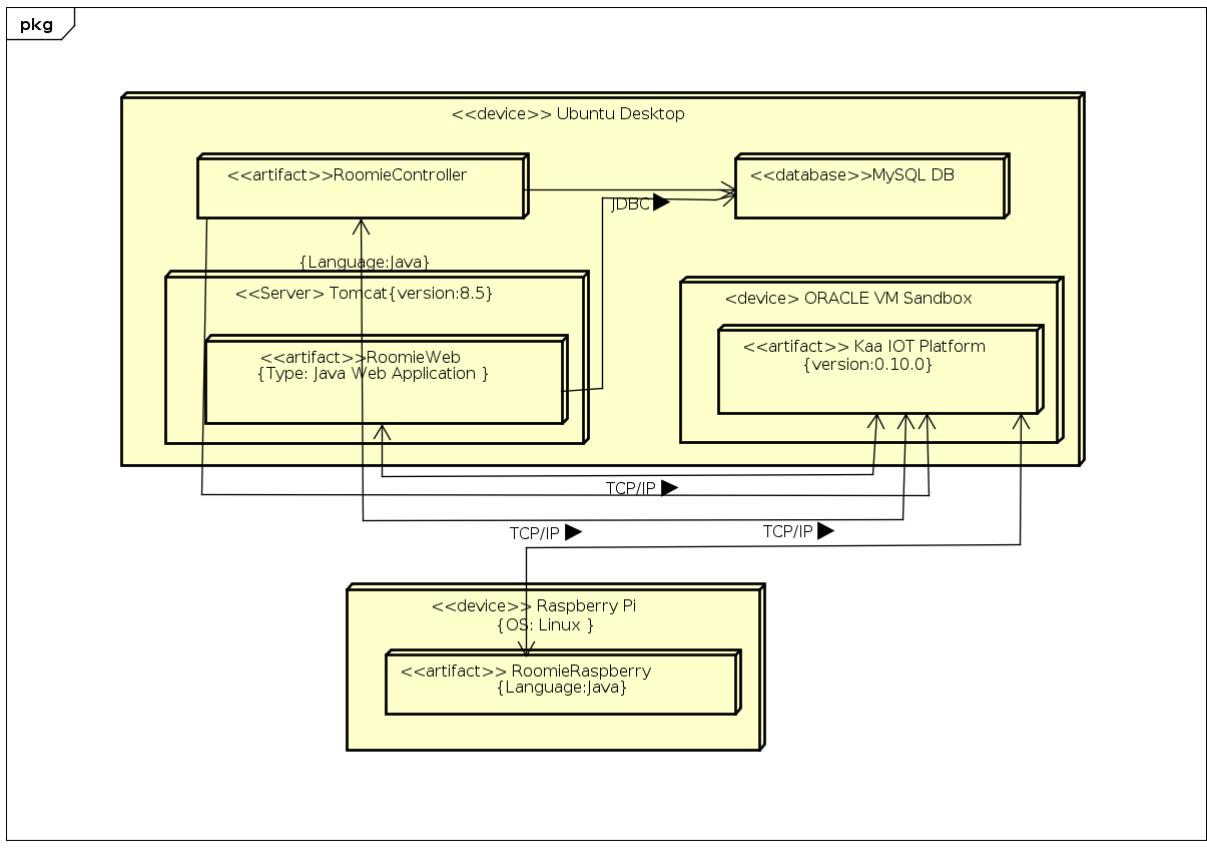


Figura 6: Diagrama de Implementação

Fonte: Próprio Autor

- RoomieController: Aplicação de controle desenvolvida na linguagem Java, tem como principal objetivo fazer o controle de disparos de eventos para as outras aplicações que estão rodando em outros dispositivos e o controle da aplicação no geral.
 - RoomieWeb: Aplicação Web em Spring MVC que permite o usuário gerenciar suas reservas em uma interface amigável.
 - Plataforma Kaa IOT: é o *middleware* de IoT responsável pela conexão das aplicações distintas, essa comunicação é baseada através do uso de eventos.
 - Banco de Dados MySQL: Banco de dados local responsável pelo armazenamento de dados de usuário e das reservas.
- Desktop Ubuntu
 - RoomieRaspberry: aplicação Java responsável pela captação e interpretação dos dados coletados pelos sensores e envio por meio de eventos para o Kaa.

Código	Nome	A Third Head
RF01	Efetuar Login	O usuário deve efetuar login no RoomieWeb para gerenciar suas reservas
RF02	Cadastrar Reservas	Permite o usuário cadastrar novas reservas RoomieWeb.
RF03	Excluir Reservas	Permite o usuário excluir suas reservas no RoomieWeb.
RF04	Visualizar Reservas	Permite o usuário visualizar suas reservas no RoomieWeb
RF05	Acessar sala com cartão RFID	O Acesso a sala reservada deve ser feito apenas aos usuários cadastrados na reserva , através da utilização do cartão RFID cadastrado

Tabela 2: Lista de requisitos funcionais

A comunicação entre as aplicações e o Middleware Kaa é realizada através do envio de eventos, que são disparadas pelas diferentes aplicações enviadas para o Kaa e depois para as aplicações de destino através do protocolo TCP/IP. Enquanto que a comunicação das aplicações com o banco local é realizada através da API do Java de comunicação com o banco de dados, o JDBC.

4.1.2 Requisitos da Aplicação

Um software é construído para resolver um problema ou vários problemas levantados por alguma organização ou indivíduo, no entanto essa é uma visão bastante ampla do que o sistema precisa realizar, por isso é preciso descrever mais detalhadamente as funcionalidades e comportamento esperados. Uma descrição exata do que o sistema deve fazer pode ser expressada através de requisitos, requisitos são as funções e restrições da aplicação. Os requisitos do Roomie serão divididos em requisitos funcionais e não-funcionais., enquanto o outro é utilizado para descrever características mais genéricas do sistema e normalmente não de funcionalidades (SOMMERVILLE, 2006).

4.1.2.1 Requisitos Funcionais

Os requisitos funcionais (RF) são utilizados para descrever as funcionalidades ou serviços promovidos pelo sistema (SOMMERVILLE, 2006). Na tabela 2, são especificados os requisitos da aplicação Roomie.

4.1.2.2 Modelagem de Casos de Uso

Uma forma de representar esses requisitos e os personagens que participam nele é através de um diagrama de casos de uso, esse modelo tem como principal finalidade modelar as funcionalidades do sistema (BEZERRA, 2007). Na figura 7 tem-se o diagrama de casos de uso para o sistema Roomie.

As descrições completas dos casos de uso da figura 7 se encontram no Apêndice A. A tabela 3 contém a descrição dos atores (qualquer elemento externo que interaja com o sistema).

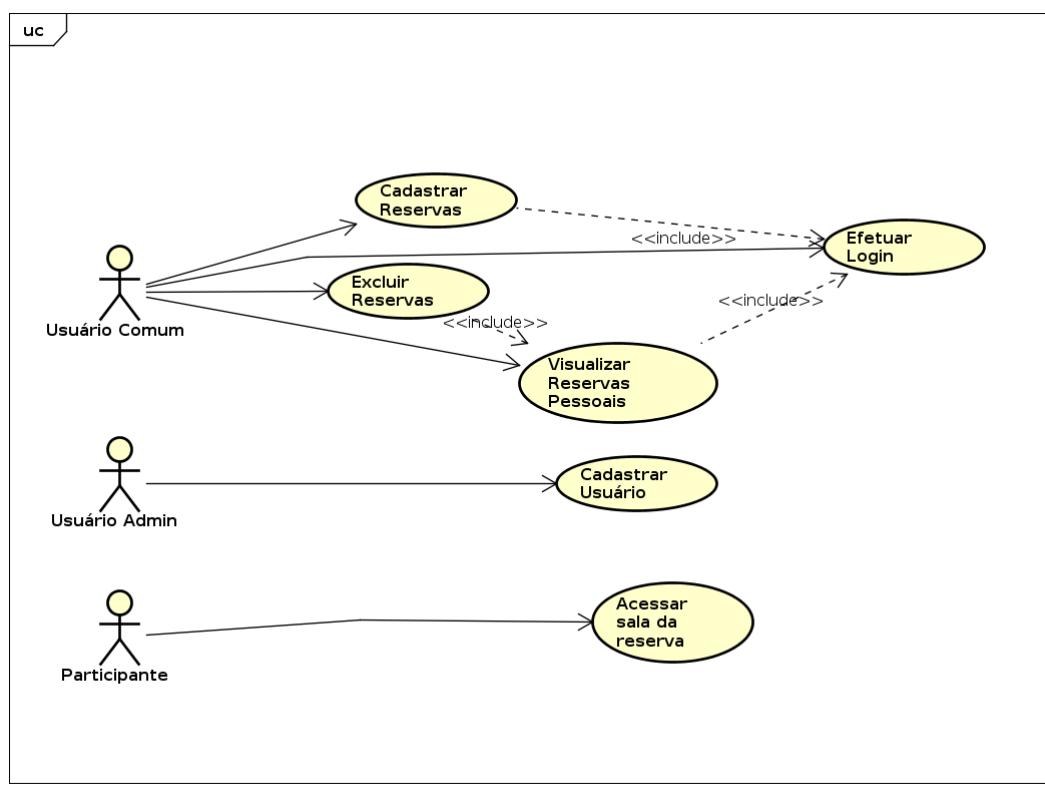


Figura 7: Diagrama de Casos de Uso

Fonte: Próprio Autor

4.1.2.3 Requisitos Não Funcionais

Os requisitos não funcionais (RNF) por sua vez são utilizados para descrever características mais genéricas do sistema e normalmente não de funcionalidades específicas (SOMMERVILLE, 2006). A tabela 4, são especificados os requisitos não funcionais da aplicação Roomie.

Ator	Descrição
Usuário Comum	Usuário que está logado na aplicação RoomieWeb.
Usuário Admin	Usuário que possui o acesso ao RoomieRaspberry.
Participante	É o usuário que possui cadastro no banco de dados, mas não necessariamente reservou uma sala. Mas está incluído na reserva como participante de uma reunião.

Tabela 3: Atores

CódigoNome	Descrição	
RNF01	Tempo de Resposta	O tempo de resposta das aplicações conectadas devem ser near real-time.
RNF02	Segurança na Aplicação Web	O sistema deve permitir apenas aos usuários logados permissão para gerenciar reservas.
"RNF03	Interface	As aplicações com interfaces com o usuário devem ser claras e objetivas.
RNF04	Armazenamento de Senhas	Apenas a senha no formato SHA256 será armazenada no banco de dados;
RF05	Acessar sala com cartão RFID	O Acesso a sala reservada deve ser feito apenas aos usuários cadastrados na reserva, através da utilização do cartão RFID cadastrado.

Tabela 4: Lista de requisitos não funcionais

4.1.3 Diagrama de Classes

Diagrama de classes são utilizados para representar o aspecto estrutural e estático do sistema. Estático por não descrever o comportamento dos objetos do sistema ao longo do tempo e estrutural por focar no relacionamento entre as partes do sistema (BEZERRA, 2007). Na Figura 8 é mostrado o Diagrama de classes do sistema Roomie, focado em um nível geral das diferentes aplicações e os pacotes que compõem as aplicações. Na seção 4.2 serão incluídos outros diagramas de classes com mais detalhes estruturais das diferentes aplicações.

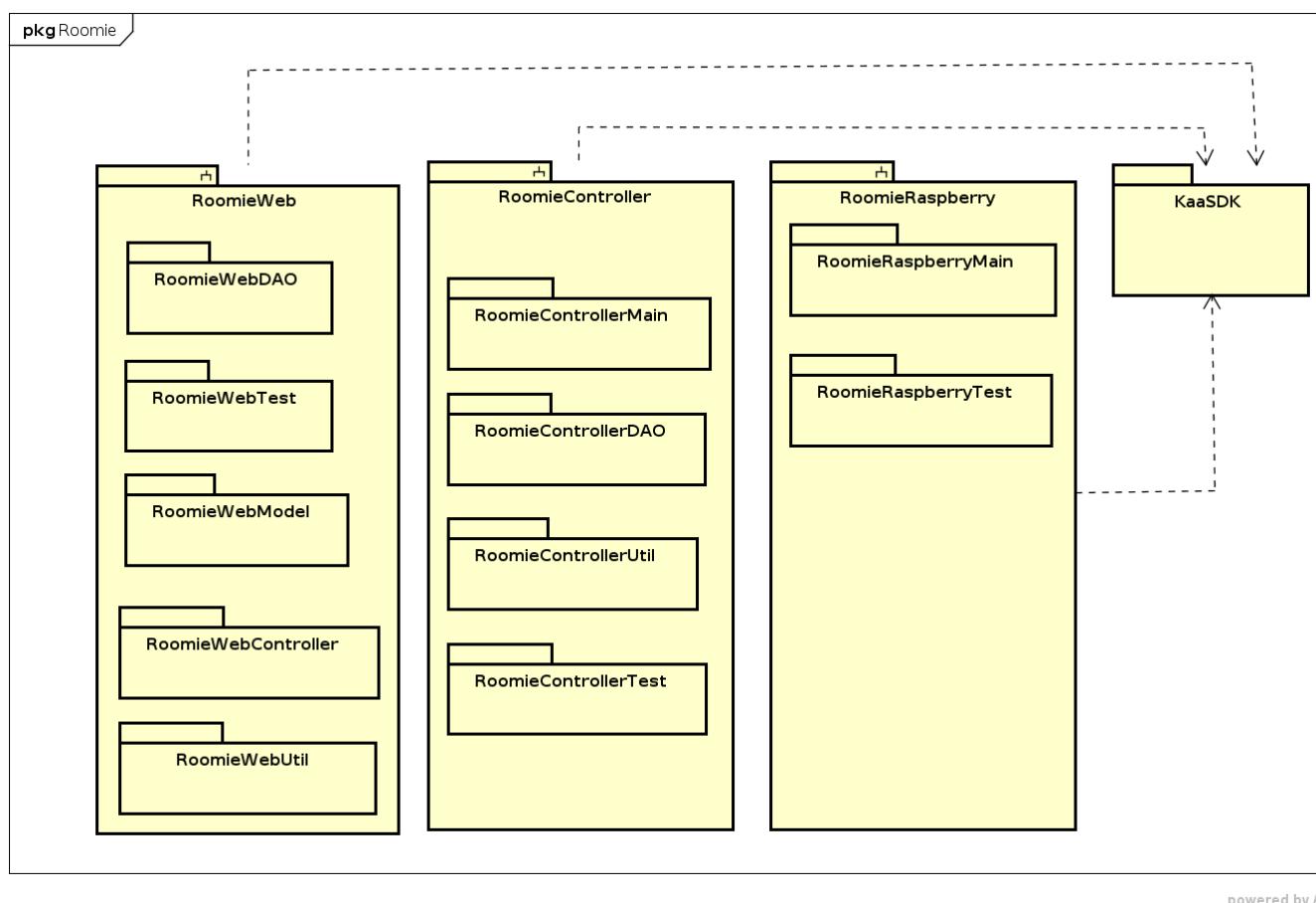


Figura 8: Diagrama de Classes Roomie

Fonte: Próprio Autor

4.1.4 Modelo de Processo de Negócio

Nesta seção é utilizado o modelo de processo de negócios ou mais conhecido como BPM (*Business Process Model*) para mostrar o comportamento e procedimentos mais importantes dentro dos sistemas Roomie. Esse tipo de diagrama consegue capturar bem fluxos

de sequências e mensagens entre diferentes processos, algo que é bastante importante na aplicação Roomie, uma vez que os diferentes sistemas se comunicam constantemente através de envio de mensagens. O modelo da figura 9 descreve o fluxo de uma reserva realizado pela aplicação Roomie.

Descrição do BPM

1. RoomieWeb

- (a) Gerenciar Reserva: Inicia quando o usuário modifica ou cria uma nova reserva e, logo após ocorre a gravação da mudança no banco de dados, caso essa reserva aconteça na hora corrente é enviada uma mensagem de atualização para o RoomieController, caso a reserva não aconteça na hora corrente o fluxo se encerra na atualização do banco.
- 2. RoomieController: como descrito anteriormente o RoomieController tem como principal funcionalidade coordenar o sistema como um geral. Como mostrado no BPM existem 3 fluxos que podem acontecer nessa instância do sistema:

- (a) Interrupção de reserva: inicia quando a um recebimento de uma mensagem informando que uma reserva foi interrompida, a mensagem é enviada pelo Roomie Raspberry. Após isso ocorre uma gravação do cancelamento da reunião no banco dados e o fluxo termina.
- (b) Programação de reservas: de hora em hora um tarefa de programação para recuperar as reservas da hora seguinte é executada e são agendados os envio de mensagens de início das reservas para serem disparadas no horário da reserva. Quando o horário da reserva chega o sistema envia uma mensagem de início da reserva para o Roomie Raspberry e o fluxo se encerra.
- (c) Atualização de Reservas: Quando alguma reserva da hora corrente é modificada é recebida uma mensagem indicando atualização na reserva, logo após isso são feitos os passos de atualizar as reservas que foram recuperadas para a hora corrente.

3. RoomieRaspberry

- (a) Caso o cartão fornecido seja igual o de algum participante da reserva, o acesso a sala é liberado. Logo após o sistema entra no modo de detecção de presença, caso a sala fique com inatividade por mais de 5 minutos é enviada uma mensagem

para o cancelamento da reserva. Caso não for registrado inatividade o sistema continua em detecção da presença até o fim do tempo da reserva e o fluxo finaliza.

- (b) Caso contrário o sistema continua em modo de leitura até o que o cartão válido seja fornecido, caso se passe 20% do tempo da reserva é enviado um email alertando o usuário do possível cancelamento. Caso chegue a marca de 30% do tempo total da reserva e nenhum cartão seja identificado a reserva é cancelada e é enviada uma mensagem para o cancelamento da reunião para o RoomieController.

4.2 Implementação e Testes da Aplicação

4.2.1 Desenvolvimento da aplicação no Middleware Kaa

Para trazer a aplicação de reservas Roomie para o mundo de IoT foi utilizado o middleware Kaa, o Kaa possibilita a troca de mensagens entre as aplicações de maneira simples e com pequeno esforço. A escolha do Kaa como middleware partiu pelo fato de ser uma plataforma de código-aberto, que é independente de plataforma e por ser atualmente estar sendo atualizada constantemente. Propiciando um espaço mais seguro e simples para o desenvolvimento da aplicação.

4.2.1.1 Configuração da Instalação do Kaa

Para a implementação do Roomie, o Kaa foi instalado em uma máquina virtual, a Oracle VM Virtual Box - 5.1.2, com 2GB reservados para a máquina virtual e com uso de 2 processadores.

4.2.1.2 Visão Geral da Arquitetura e funcionamento do Kaa

A arquitetura do Kaa segundo (CYBERVISION INC., 2016a) é formada basicamente por três partes distintas:

- Servidor Kaa : é a parte back-end da plataforma, é responsável pelo gerenciamento dos elementos que são acoplados a ferramenta como as aplicações, dispositivos e elementos internos.

- Extensões Kaa: são softwares independentes do Kaa, que ajudam o Kaa em funções específicas. Exemplos dessas extensões como mostrado na figura 08 é o Zookeeper Apache e bancos de dados SQL e NoSQL.
- *Endpoints SDKs*: Os endpoints SDKs são bibliotecas instaladas nos dispositivos a serem conectados no Kaa. Esse sdk permite uma comunicação fácil e padronizada com o Kaa através de uma API.

Existem dois tipos de mensagens de comunicação entre o servidor kaa e os *endpoints sdfs*, são as notificações e os eventos. As notificações são utilizadas para envio de mensagens do servidor Kaa para os endpoints que estão cadastrados no mesmo tópico. Enquanto que os eventos são utilizados para o envio de informações de *endpoints* para outros *endpoints*, e esse envio sendo gerenciado pelo servidor do Kaa. Na aplicação Roomie utilizamos apenas eventos como meio de comunicação dos endpoints.

4.2.1.3 Eventos

A geração de eventos através da plataforma Kaa são suportados de maneira quase que tempo-real, os eventos são gerados por endpoints gerados pelo mesmo usuários tratados e enviados pelo kaa para os endpoints que estão cadastrados para o recebimento do evento específico. Os dados enviados nos eventos são configuráveis no próprio Kaa, cada evento possuindo esquema diferente para classe que representa um evento (CYBERVISION INC., 2016c). Cada evento possui uma classe de evento associada a ele, não podendo assim existir dois eventos com a mesma classe. Classes de evento (EC) são organizadas em famílias de classes de eventos (ECF). Cada *endpoint* cobra para o recebimento de uma classe de família, se o *endpoint* recebe um evento de uma família específica deve ser configurado para receber todos os eventos da família específica. O envio de eventos pode ser feito para um *endpoint* específico (unicast) ou para todos os *endpoints* cadastrados para o recebimento para aquele evento (multicast) (CYBERVISION INC., 2016c).

Classificando *endpoints* segundo a ECF Os endpoints podem enviar, receber e enviar/receber eventos de uma ECF. Para distinguir os papéis do endpoints na configuração do endpoint podemos classificar *endpoint* segundo a uma ECF de três maneiras:

- source: apenas envia eventos mas não os recebe;
- sink : apenas recebe eventos;

Família de Eventos	Evento	Descrição
RegisterUserECF	RegisterUserEvent	Evento de registro de novo usuário
ConfirmationEventECF	ConfirmationEvent	Evento de confirmação de cadastro de usuário
StartMeetingECF	StartMeetingEvent	Sinaliza o início de uma reunião enviando as informações da reunião
InterruptingMeetingECF	InterruptMeetingEvent	Evento de Cancelamento de Reserva
MeetingChangedECF	MeetingChangedEvent	Sinaliza que uma reserva na hora atual foi modificada

Tabela 5: Lista de requisitos não funcionais

Evento	Dados enviados	Quem envia	Quem recebe
RegisterUserEvent	user: User user:User	RoomieRaspberry	RoomieController
ConfirmationEvent	status:String email:String isRegistered:Boolean	RoomieController	RoomieRaspberry
StartMeetingEvent	meeting:Meeting	RoomieController	RoomieRaspberry
InterruptMeetingEvent	meetingId:Integer interruptReason:String	RoomieRaspberry	RoomieController
MeetingChangedEvent	meetingId:Integer	RoomieWeb	RoomieController

Tabela 6: Descrição detalhada dos eventos

- both: recebe e envia evento da ECF específica;

Eventos no Roomie

Para realizar a trocas de informações entre as diferentes aplicações Roomie são utilizado os eventos do Kaa. Na Figura 11 é mostrado um diagrama ilustrando os eventos utilizados no Roomie. Na tabela 5 se encontra descrição de todos os eventos utilizados pelo sistema Roomie. E na tabela 6 uma descrição eventos e dos dados que são enviados por cada evento, a tabela 7 é usado como apoio para a descrição de objetos.

4.2.1.4 Códigos Conexão Kaa - Java

Nesta seção são apresentados códigos básicos utilizados para a conexão com a plataforma Kaa.

Código de Inicialização do Kaa No trecho da figura 12, um cliente Kaa é inicial-

Objeto	Descrição dos campos
User	userName: String email: String hashedPassword: String isOwner:boolean rfidCode: String;
"Meeting	users: Array Users startTime:String endTime:String meetingName:String meetingId:Integer roomLocation:String roomId:Integer roomName:String

Tabela 7: Descrição dos objetos suporte

izado, e são definidos comportamentos para serem executados depois da inicialização e para quando o Kaa for pausado. Como as chamadas do Kaa são assíncronas, ou seja, o código continua a ser executado e não espera o fim da execução da chamada, é utilizado um mecanismo para pausar a thread principal até a chamada ser finalizada e o fim dela ser sinalizado.

Código de Anexação da cliente ao usuário

A anexação de um cliente a um usuário é necessária para habilitar o envio/recebimento de eventos, vindo desse cliente, o código na figura 13 ilustra a operação de anexação.

Registro para recebimento de eventos O código na figura 14 representa o cadastramento do cliente para o recebimento do evento, nesse caso InterruptMeetingEvent e apresenta o comportamento após o recebimento do evento.

Envio de Eventos

O código da figura 15 é utilizado para o envio de um evento. Anteriormente ao envio de um evento é necessário saber quais endpoints estão cadastrados para receber o evento específico, para isso é utilizado o método findEventListeners. Logo após identificado os endpoints pode se enviar o evento para um endpoint específico (unicast) ou para todos os endpoints cadastrados para o recebimento do evento (broadcast). No código apresentado na figura 15 o evento é enviado em modo broadcast.

4.2.2 Desenvolvimento da aplicação RoomieRaspberry

A aplicação no Raspberry Pi foi feita na linguagem Java e para a interpretação dos dados que partiam dos sensores foi utilizada a biblioteca Pi4J. O Raspberry Pi foi escolhido pela a liberdade no uso de linguagens de programação e pelo seu maior poder computacional se comparando com outras placas de prototipação semelhantes. As principais funcionalidades presentes no RoomieRaspberry são a leitura dos dados de sensores RFID e de presença; o controle no fluxo de execução da reserva como mostrado no BPM da figura 9 e o recolhimento dos dados para cadastro do usuário na plataforma Roomie.

4.2.2.1 Diagrama de Classes

A estrutura do RoomieRaspberry é bem mais simples se comparado com as outras aplicações Roomie, basicamente as classes utilizadas são para realizar a comunicação com Kaa e para a leitura do sensor RFID. As classes de testes são de teste unitários das classes da aplicação.

4.2.2.2 Configuração dos sensores no Raspberry Pi

Os sensores utilizados para a captação dos dados e para saída de dados foram os seguintes:

- Sensor de Presença PIR;
- Leitor cartão RFID RC522;
- Leds

Para a conexão com sensores o Raspberry Pi tem uma série de pinos de entrada/saída, os chamados GPIO, na figura 17 é mostrado o mapeamento dos pinos. Na tabela ?? segue o mapeamento dos pinos que foram usados para a conexão com os sensores.

Os Leds Coloridos são utilizados no fluxo de execução da reserva, um led por vez é acesso representando um estágio no fluxo de reserva.

- Vermelho: Não existe reserva no momento.
- Amarelo: Reserva identificada e sala pronta para a identificação do RFID.
- Verde: indica que o cartão RFID foi liberado e que a sala está liberada.

Sensor/Pino	Numeração Pino Raspberry	Pino GPIO
RC522/SDA	24	GPIO8
RC522/SCK	23	GPIO11
RC522/MOSI	19	GPIO10
RC522/MISO	21	GPIO9
RC522/GND	20	GND
RC522/3.3V	1	3V3
PIR/VCC	2	5V3
PIR/GND	6	GND
PIR/OUT	16	GPIO(23)
Led Vermelho	7	GPIO 04
Led Verde	11	GPIO 17
Led Amarelo	13	GPIO 27

Tabela 8: Mapeamento de pinos para o Raspberry Pi

4.2.3 Desenvolvimento do RoomieWeb

A aplicação RoomieWeb foi desenvolvida utilizando a linguagem Java e o framework Spring MVC. O principal objetivo da aplicação é fornecer uma interface amigável para os usuários gerenciarem suas reservas.

4.2.3.1 Diagrama de Classes

A Figura 18 apresenta o diagrama de classes do RoomieWeb, como mencionado anteriormente é utilizado o modelo arquitetural MVC com adição de um pacote de Objeto de Acesso a Dados também chamado DAO (Data Access Object) para promover o desacoplamento das classes que lidam com banco de dados. As classes de teste no diagrama são classes utilizadas para testes unitários da aplicação.

4.2.4 Principais Telas da Aplicação

Login A figura 19 representa a tela inicial de login da aplicação RoomieWeb, nessa tela usuários que já estão cadastrado no sistema Roomie podem ter acesso a plataforma.

Criação de nova reunião

A figura 20 representa a tela da criação de nova reservas na aplicação RoomieWeb, nessa tela o usuário pode criar uma nova reserva no RoomieWeb. Os campos de salas e usuários são populados diretamente do banco de dados.

Visualização/Edição de Reservas A figura 21 representa a tela de listagem de todas as reuniões que o usuário logado no momento participa, caso o usuário seja o criador da reserva também é possível realizar a edição/exclusão da reserva.

4.2.4.1 Desenvolvimento do RoomieController

A Aplicação RoomieController tem como principal objetivo gerenciar o sistema como um todo, ou seja, é a entidade responsável por diversas atividades que permitem que aplicação funcione corretamente. A aplicação RoomieController recupera as informações necessárias para o funcionamento da aplicação RoomieRaspberry e também gerenciar o disparo de eventos de início de reuniões. Na Figura 22 é mostrado o diagrama de classes da aplicação.

4.2.5 Execução Final da Aplicação

Para o teste final do sistema Roomie, foi feito um experimento em uma sala de reunião durante 2 horas. O roteiro do teste foi feito para que grande parte das possibilidades e situações diferentes fossem testadas. Foram utilizados três personagens no cenário: Lauren, Alex e Camila. Os três tinham diferentes cartões rfids e cadastros separados na aplicação. A tabela 9 contém as reuniões que foram criadas e mais informações sobre cada reunião.

Vídeo de execução da aplicação

Nome da Reunião	Horário	Participantes	Descrição do Cenário
Lauren - Individual	09:00-09:15	Lauren	<p>09:03 - Lauren recebe um email informando que caso não compareça a sala sua reserva será cancelada.</p> <p>09:03:30 Lauren entra na sala com seu cartão RFID e tem acesso permitido.</p>
Plano de Ação	09:20 - 09:45	Lauren, Alex	<p>Lauren convida Camila para a reunião mas não edita a reserva.</p> <p>09:20 - Camila chega na sala, mas não consegue se autenticar com seu cartão RFID.</p> <p>09:22 - Alex chega na reunião com seu cartão RFID e tem acesso liberado.</p>
1:1 Camila e Alex	10:00 -10:30	Camila, Alex	<p>10:00 - Ambos chegam na reunião as 10:00 após 15 min a reunião é finalizada, mas os participantes não cancelam a reserva no sistema.</p> <p>10:20 - Reunião é automaticamente cancelada</p>
Camila - Individual	10:40-11:10	Camila	<p>10:46 - Camila recebe email avisando que sua reserva será cancelada caso não compareça até às 10:49.</p> <p>10:50 - Camila tenta entrar nas sala mas não consegue autenticar com seu cartão.</p>

Tabela 9: Tabela de casos de teste

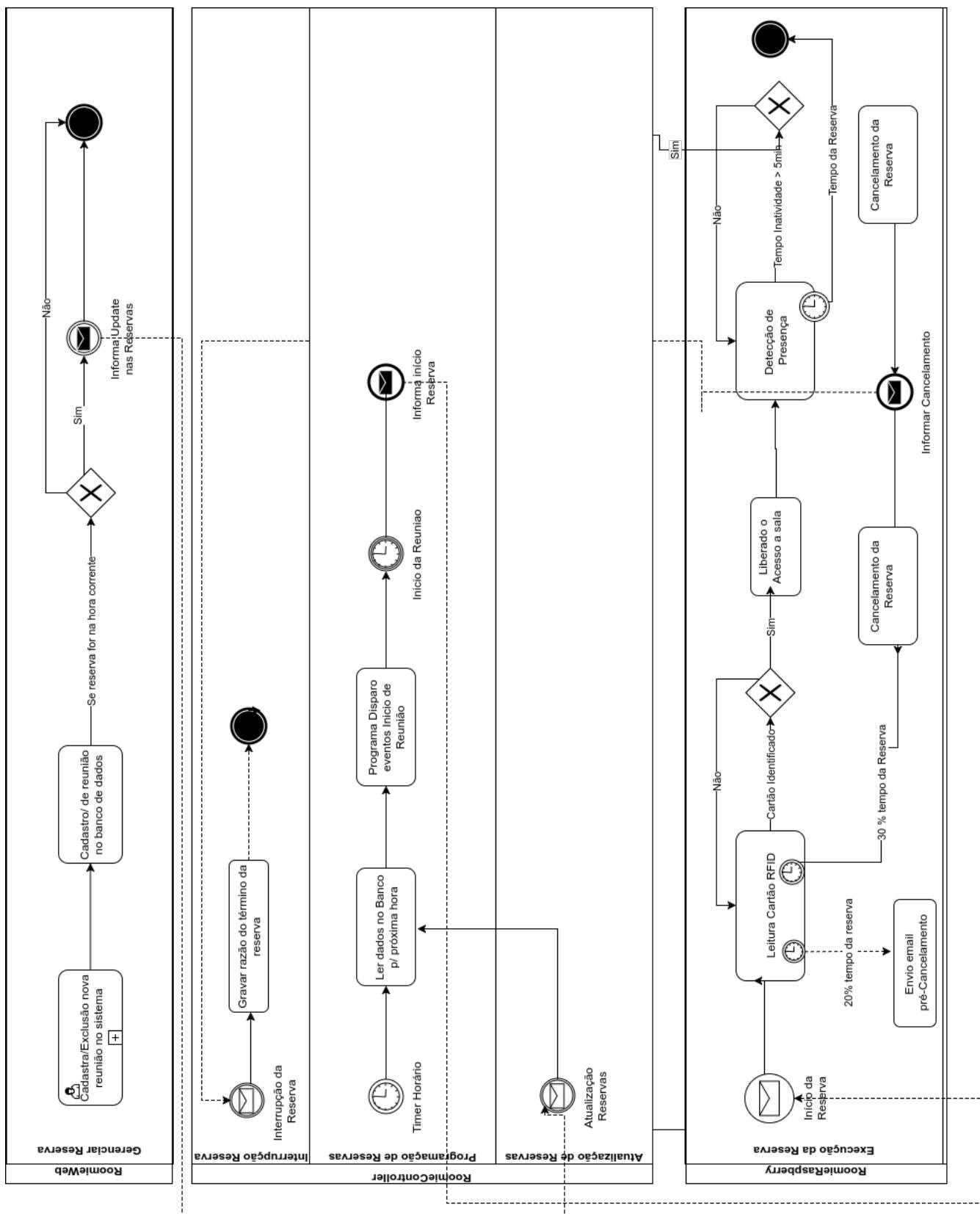


Figura 9: BPM de Fluxo de reserva do Sistema Roomie
Fonte: Próprio Autor

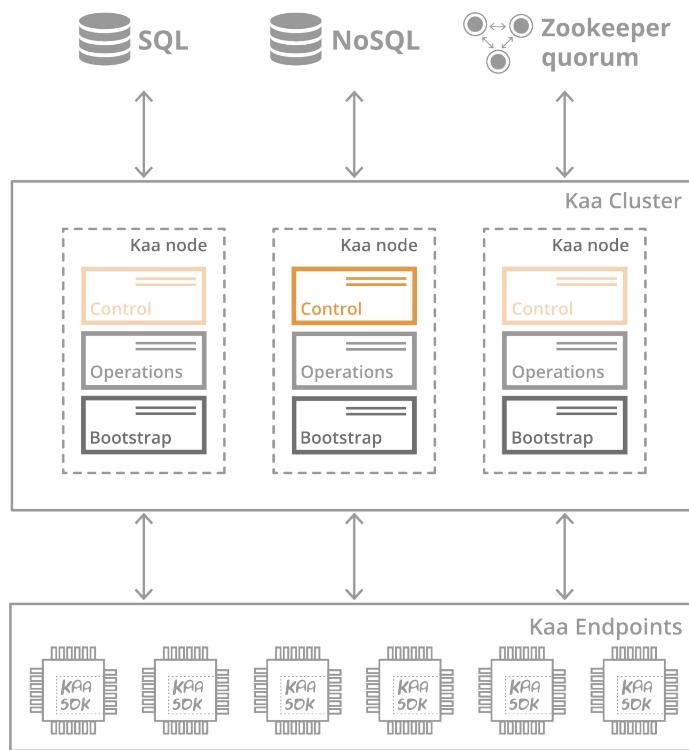


Figura 10: Arquitetura Kaa
Fonte: (CYBERVISION INC., 2016b)

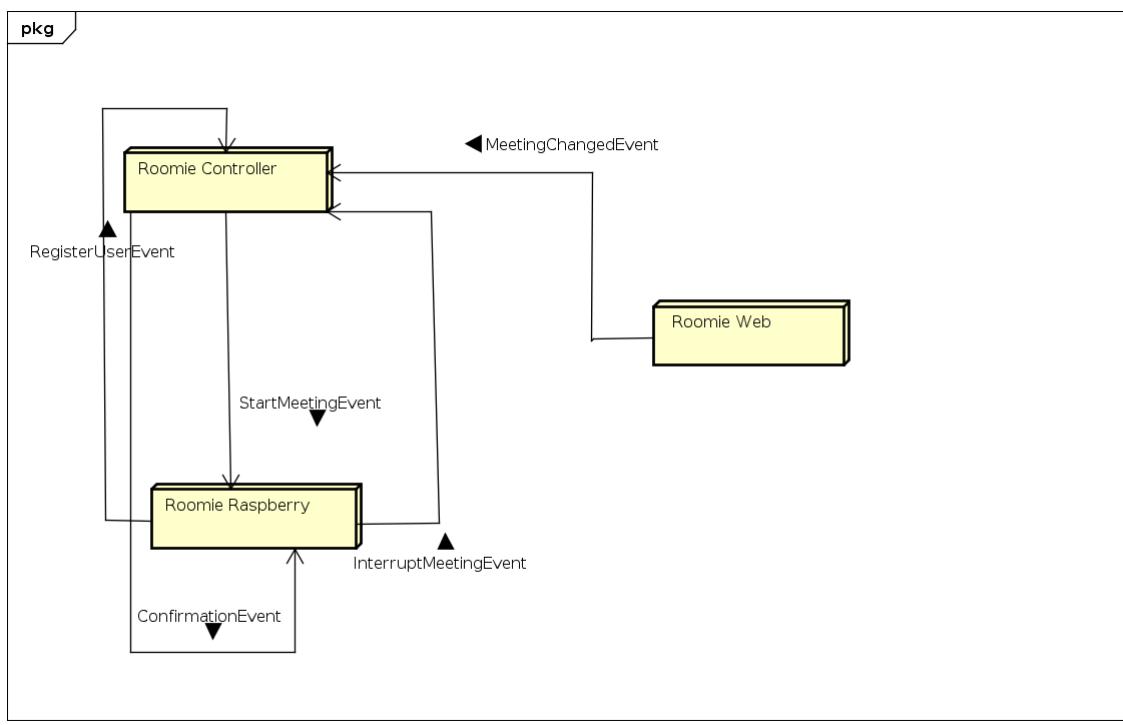


Figura 11: Arquitetura Kaa
Fonte: Próprio Autor

```

57 CountDownLatch startUpLatch = new CountDownLatch(1);
58 KaaClientProperties endpointProperties = new KaaClientProperties();
59 endpointProperties.setWorkingDirectory(KEYS_DIR);
60
61 // Create the Kaa desktop context for the application
62 DesktopKaaPlatformContext desktopKaaPlatformContext = new DesktopKaaPlatformContext(endpointProperties);
63 // new Desktop
64 KaaClient kaaClient = Kaa.newClient(desktopKaaPlatformContext, new SimpleKaaClientStateListener() {
65     //defines behaviour for after kaa had started
66     @Override
67     public void onStart() {
68         System.out.println("Kaa client started");
69         //makes the code after await to be executed
70         startUpLatch.countDown();
71     }
72     //defines behaviour for after kaa had stopped
73     @Override
74     public void onStop() {
75         System.out.println("Kaa client stopped");
76     }
77 },true);
78
79 kaaClient.start();
80 // code waits until countDown is called
81 startUpLatch.await();

```

Figura 12: Código de inicialização cliente Kaa

Fonte: Próprio Autor

```

84 final CountDownLatch attachLatch = new CountDownLatch(1);
85 kaaClient.attachUser("trustful", "92119438580922591122", new UserAttachCallback()
86 {
87     //makes the code after await to be executed
88     @Override
89     public void onAttachResult(UserAttachResponse response) {
90         System.out.println("Attach response" + response.getResult());
91         attachLatch.countDown();
92     }
93 });
94 // code waits until countDown is called
95 attachLatch.await();

```

Figura 13: Código de anexação do usuário

Fonte: Próprio Autor

```

interruptMeetingECF.addListener(new InterruptMeetingECF.Listener(){
    //define behaviour for when an event is executed
    @Override
    public void onEvent(InterruptMeetingEvent event, String source) {
        try {
            new MeetingDao().UpdateMeetingInterruptReason(event);
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
});

```

Figura 14: Código de registro de recebimento de eventos

Fonte: Próprio Autor

```

172     List<String> FQNs = new LinkedList<String>();
173     FQNs.add(StartMeetEvent.class.getName());
174     final CountDownLatch startMeetingLatch = new CountDownLatch(1);
175     //find all listeners for a specific event
176     kaaClient.findEventListeners(FQNs, new FindEventListenersCallback(){
177         //defines the behaviour after listeners are received
178         @Override
179         public void onEventListenersReceived(List<String> eventListeners) {
180             System.out.println("Event listeners received " + eventListeners.size());
181             for( int i =0 ; i<eventListeners.size(); i++)
182                 System.out.println(eventListeners.get(i));
183             //signals main thread waiting to continue
184             startMeetingLatch.countDown();
185         }
186         @Override
187         public void onRequestFailed() {
188             System.out.println("Failed ");
189         }
190     });
191     //signals main thread to wait
192     startMeetingLatch.await();
193     StartMeetECF startECF = eventFamilyFactory.getStartMeetECF();
194     //send a broadcast for all listeners receiving this event
195     startECF.sendEventToAll(event);
196
197
198 }

```

Figura 15: Código de envio de eventos

Fonte: Próprio Autor

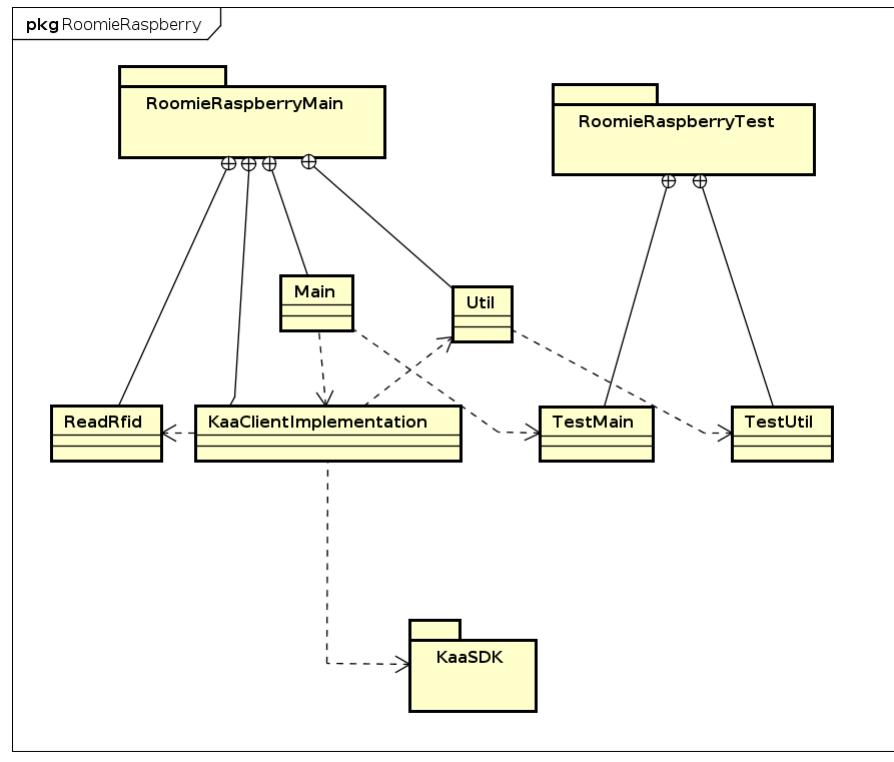


Figura 16: Diagrama de Classes RoomieRaspberry

Fonte: Próprio Autor

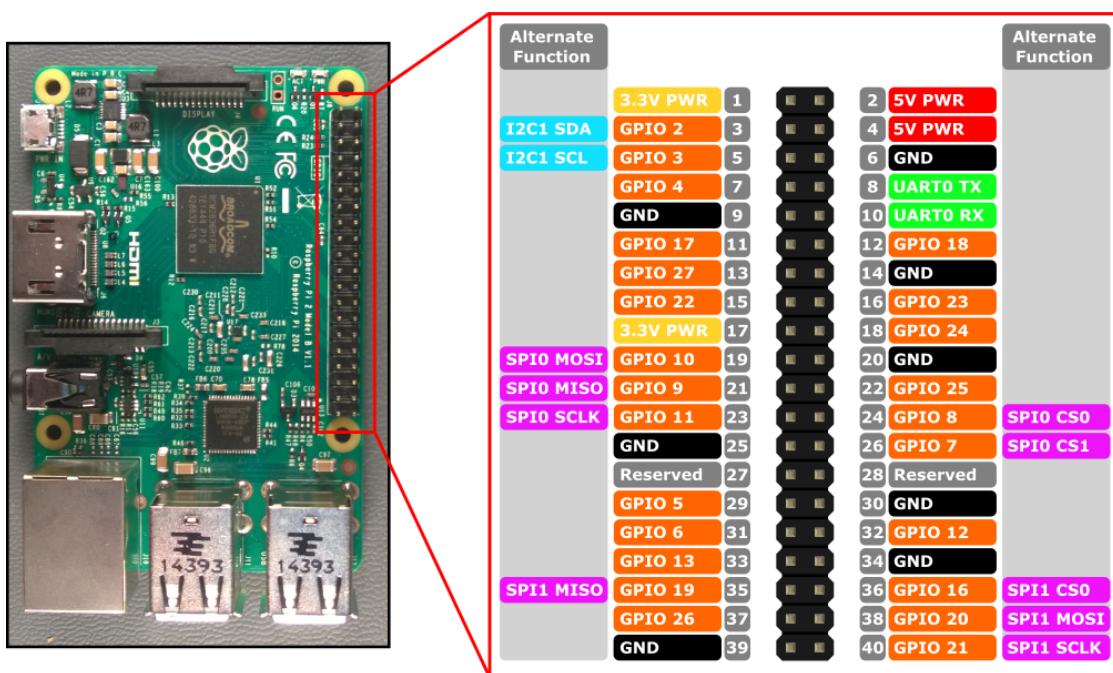


Figura 17: Mapeamento dos pinos do Raspberry Pi 2 e 3

Fonte: (Microsoft and Fritzing , 2016)

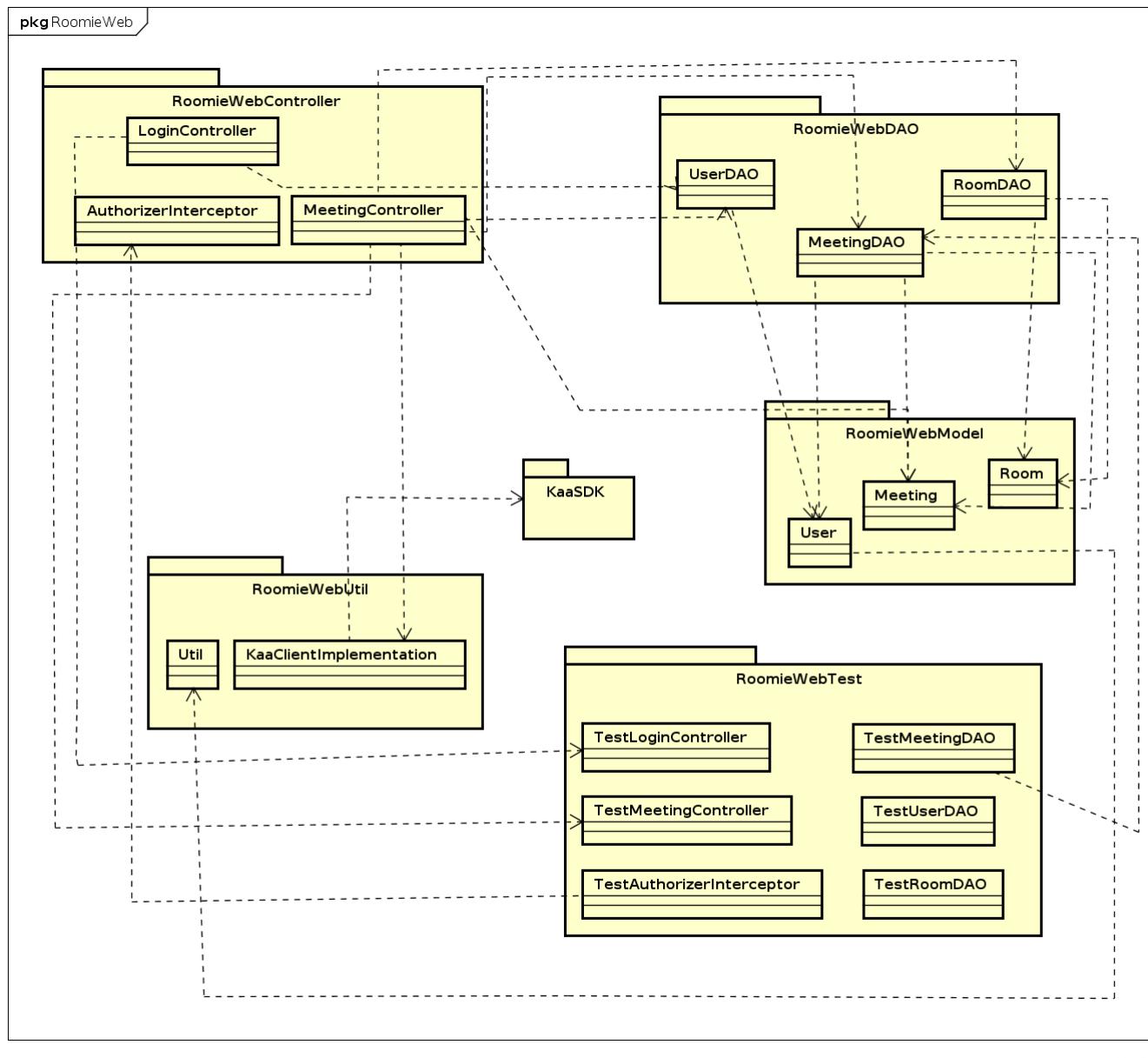


Figura 18: Diagrama de classes RoomieWeb
Fonte: Próprio Autor

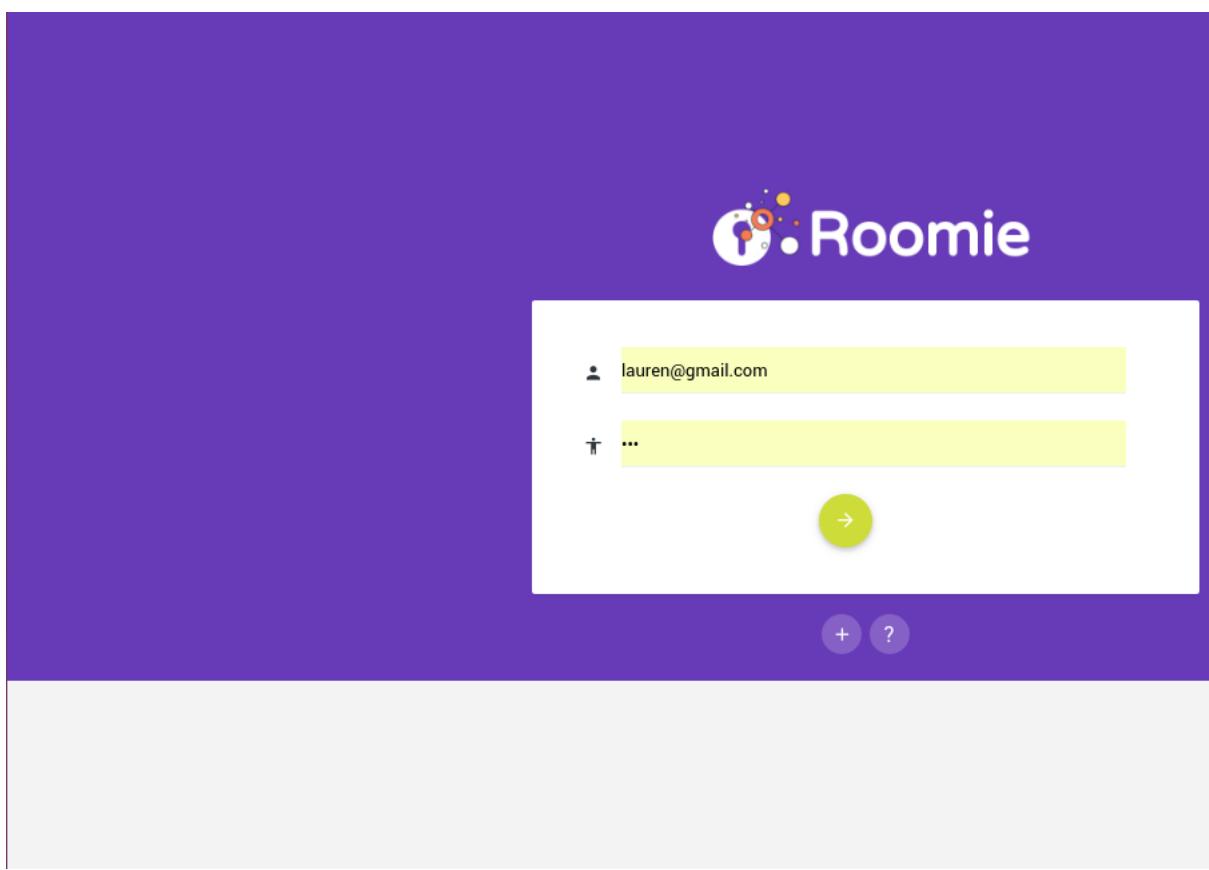


Figura 19: Tela de Login RoomieWeb
Fonte: Próprio Autor

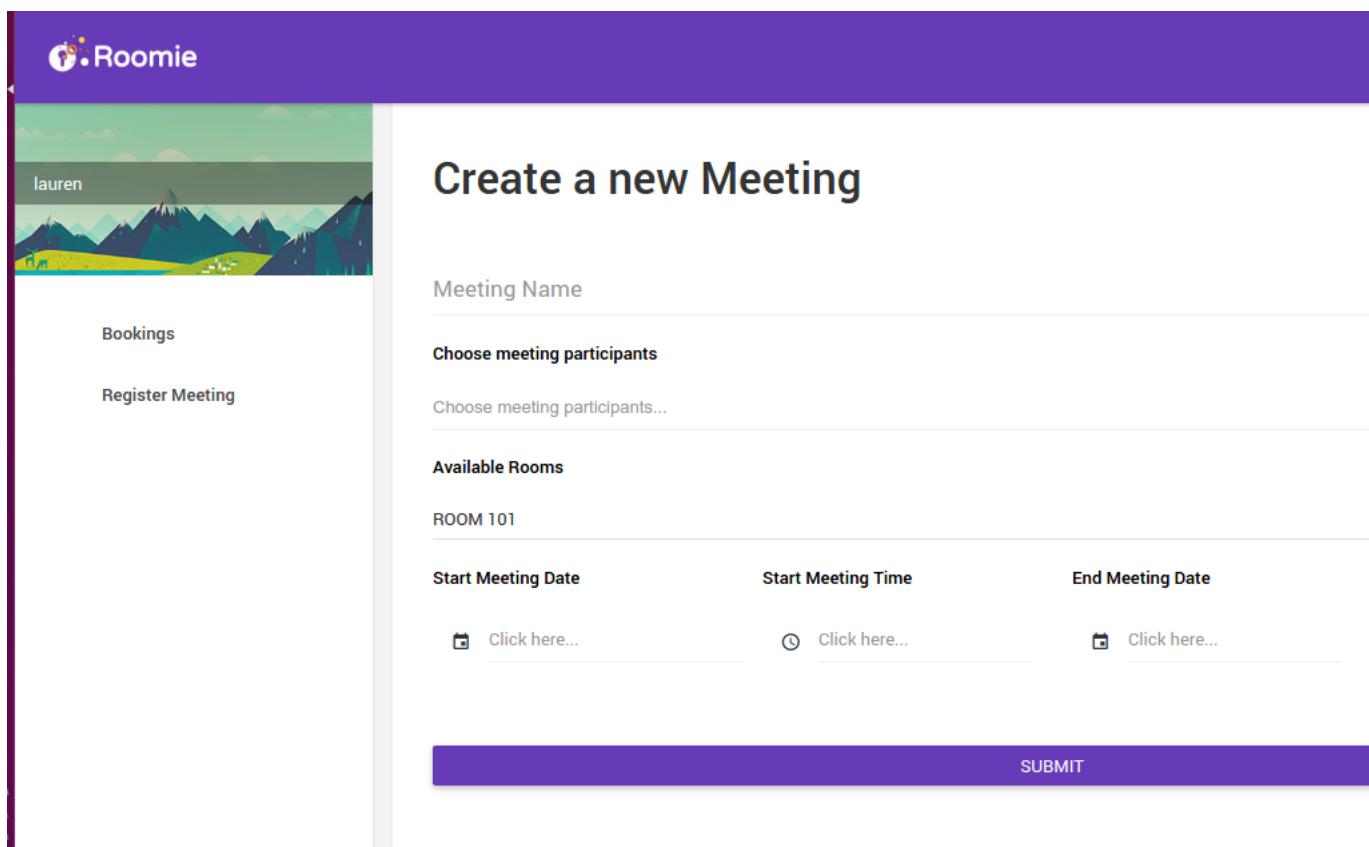


Figura 20: Tela de Login RoomieWeb

Fonte: Próprio Autor

The screenshot shows the Roomie application interface. At the top, there is a purple header bar with the Roomie logo on the left. Below the header, the user's name "lauren" is displayed. The main content area has a white background. On the left, there is a sidebar with a green header containing the user's name. Below the header, there are two buttons: "Bookings" and "Register Meeting". The "Bookings" button is highlighted with a blue border. The "Register Meeting" button is standard grey. To the right of the sidebar, the word "Reservations" is centered in a large, bold, dark font. Below "Reservations", there is a search bar with a magnifying glass icon and the word "Search". Underneath the search bar, there are two dropdown menus: one labeled "10" and another with three dots. A horizontal line separates these from a table. The table has a light gray header row with columns labeled "ID", "MEETING NAME", "START TIME", "END TIME", "ROOM", and "ROOM L". There are four data rows in the table:

ID	MEETING NAME	START TIME	END TIME	ROOM	ROOM L
1	Teste	2017-05-11 12:00:00.0	2017-05-11 12:00:00.0	room 101	1st floor
2	Lauren <3	2017-05-11 12:00:00.0	2017-05-11 12:00:00.0	room 101	1st floor
3	Michelle Obama	2017-05-11 09:00:00.0	2017-05-11 10:00:00.0	room 101	1st floor
4	Everyone	2017-05-13 10:00:00.0	2017-05-13 10:00:00.0	room 101	1st floor

Figura 21: Listagem de reuniões do usuário

Fonte: Próprio Autor

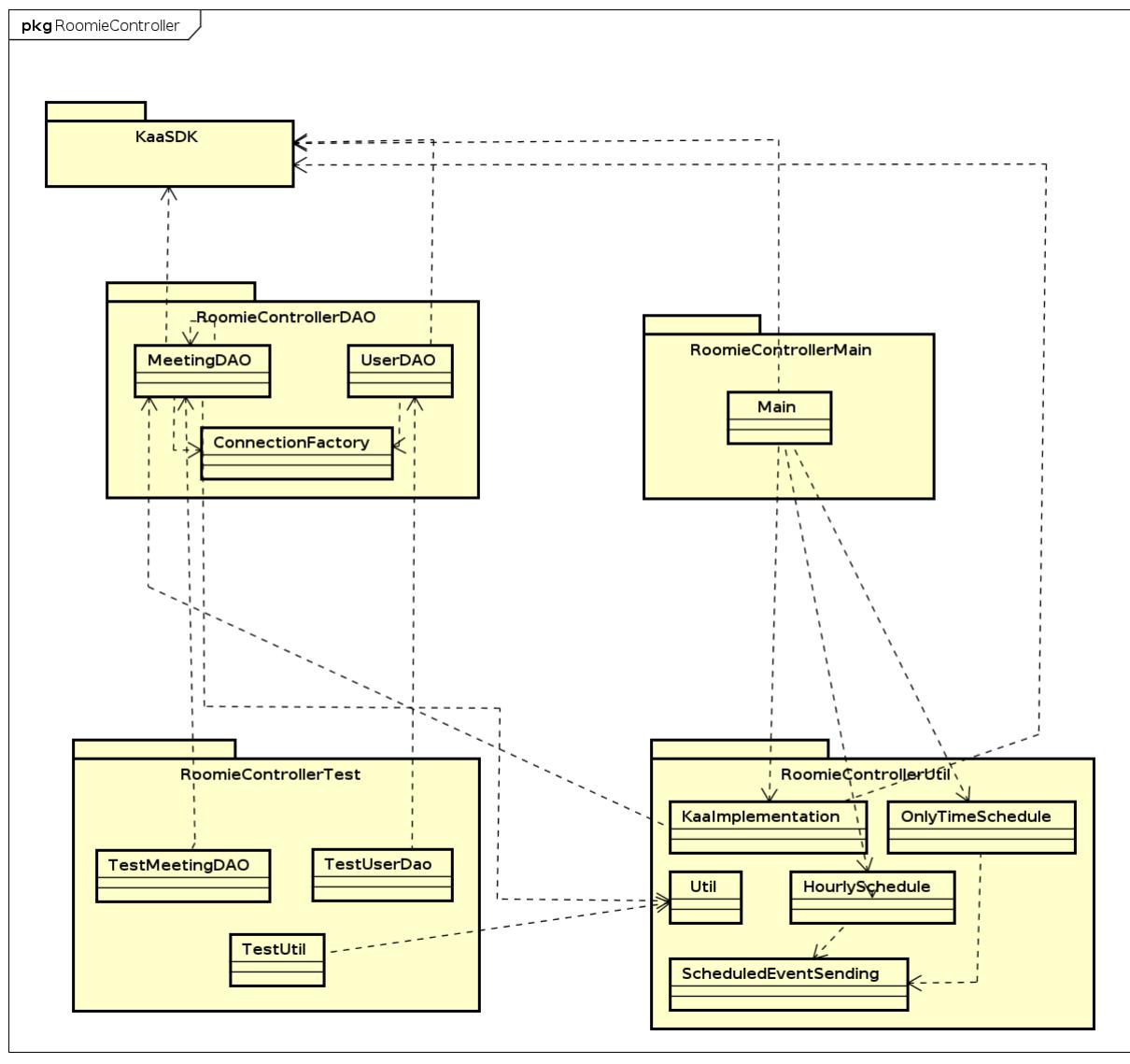


Figura 22: Diagrama de Classes RoomieController

Fonte: Próprio Autor

5 Considerações finais

5.1 Trabalhos Futuros

Referências

- ADAFRUIT. *PIR Motion Sensor:Pyroelectric ("Passive") InfraRed Sensors.* 2017. Disponível em: <<https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor?view=all>>. Acesso em 16 de Mar 2017.
- ATZORI, L. The internet of things: A survey. *Computer Networks*, v. 54, p. 2787–2805, 2010.
- BEZERRA, E. *Princípios de Análise e Projeto de Sistemas com UML*. 2th. ed. Rio de Janeiro: Elsevier, 2007.
- BOX ELECTRÓNICA. *How to solve the problem of meeting room no-shows.* 2017. Disponível em: <<https://www.boxelectronica.com/pt/rfid-nfc/77-leitor-rfid-rc522.html>>. Acesso em 02 Mar 2017.
- CYBERVISION INC. *Architeture Overview.* 2016. Disponível em: <<https://kaaproject.github.io/kaa/docs/v0.10.0/Architecture-overview/>>. Acesso em Out ,2016.
- CYBERVISION INC. *Architeture Overview.* 2016. Disponível em: <<https://kaaproject.github.io/kaa/docs/v0.10.0/Architecture-overview/attach/high-level-architecture.png>> Acesso em Out 2016.
- CYBERVISION INC. *Events.* 2016. Disponível em: <<https://kaaproject.github.io/kaa/docs/v0.10.0/Programming-guide/Key-platform-features/Events/>>. Acesso em Out ,2016.
- DUNCAN, J. *How to solve the problem of meeting room no-shows.* 2016. Out., 2016. Disponível em: <<http://blog.condecosoftware.com/how-to-solve-the-problem-of-meeting-room-no-shows>>. Acesso em Outubro ,2016.
- EASY ELECTRONYX. 2017. Disponível em: <<https://easyelectronyx.com/wp-content/uploads/2017/01/motion-sensor-productshot-01.jpg>>. Acesso em 01 Mar 2017.
- HARRIS, A. Smart buildings. *Energy And Technology Magazine*, v. 57, p. 277–295, 2012.
- JOHNSON, R. et al. *Spring Framework Reference.* 2017. Disponível em: <<http://docs.spring.io/spring/docs/current/spring-framework-reference/pdf/spring-framework-reference.pdf>>. Acesso em 17 Mar 2017.
- JUSTO, P. *Raspberry Pi or Arduino? One Simple Rule to Choose the Right Board.* 2016. Disponível em: <<http://makezine.com/2015/12/04/admittedly-simplistic-guide-raspberry-pi-vs-arduino/>>. Acesso em 11 out 2016.

KAAIOT TECHNOLOGIES. *Kaa IoT Development Platform overview*. 2016. Disponível em: <<https://www.kaaproject.org/overview/>>. Acesso em 12 Nov 2016.

Microsoft and Fritzing . 2016. Disponível em: <<https://az835927.vo.msecnd.net/sites/iot/Resources/images/PinMappings/RP2_Pinout.png> . Acesso em 10 out 2016.

NXP SEMICONDUCTORS. *MRFC522:Standard performance MIFARE and NTAG frontend*. 2017. Disponível em: <https://www.nxp.com/documents/data_sheet/MFRC522.pdf> . Acesso em 17 Mar 2017.

ORACLE. *Obtenha Informações sobre a Tecnologia Java*. 2016. Disponível em: <https://www.java.com/pt_BR/about/> . Acesso em 13 out 2016.

Oracle. *About MySQL*. 2017. Disponível em: <http://www.biblioteca.fsp.usp.br/~biblioteca/guia/a_cap_03.htm> . Acesso em 17 Mar 2017.

PADMANABH, K. et al. isense: A wireless sensor network based conference room management system. In: *BuildSys '09 Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*. [S.l.: s.n.], 2009. p. 37–32.

PI4J. *The Pi4J Project: Java I/O Library for Raspberry Pi*. 2017. Disponível em: <<http://blog.condecosoftware.com/how-to-solve-the-problem-of-meeting-room-no-shows>> . Acesso em Out 2016.

SOMMERVILLE, I. *Software Engineering: (Update) (8th Edition) (International Computer Science)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006. ISBN 0321313798.

TAN, L.; WANG, N. Future internet: The internet of things. In: *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*. [S.l.: s.n.], 2010. v. 5, p. 376–380. ISSN 2154-7491.

TRAN, L. D. et al. A smart meeting room scheduling and management system with utilization control and ad-hoc support based on real-time occupancy detection. In: *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*. [S.l.: s.n.], 2016. p. 186–191.

XBIAN COMMUNITY. *DB2XML 1.4: relational databases into documents*. 2017. Out., 2017. Disponível em: <<http://www.xbian.org/preliminary-raspberry-pi-3-support>> . Acesso em Março 1,2017.

APÊNDICE A - Primeiro apêndice

Os apêndices são textos ou documentos elaborados pelo autor, a fim de complementar sua argumentação, sem prejuízo da unidade nuclear do trabalho.