LUIZ WALBER DE SOUSA FREITAS FILHO

Orientador: Carlos Frederico Marcelo da Cunha Cavalcanti

DESENVOLVIMENTO DE SISTEMAS ESCALÁVEIS E NA NUVEM - ESTUDO DE CASO

Ouro Preto

Dezembro de 2019

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

DESENVOLVIMENTO DE SISTEMAS ESCALÁVEIS E NA NUVEM - ESTUDO DE CASO

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

LUIZ WALBER DE SOUSA FREITAS FILHO

Ouro Preto
Dezembro de 2019

F866d Freitas Filho, Luiz Walber de Sousa.
Desenvolvimento de sistemas escaláveis e na nuvem [manuscrito]: estudo de caso / Luiz Walber de Sousa Freitas Filho. - 2019.

41f.: il.: color; grafs; tabs.

Orientador: Prof. Dr. Carlos Frederico Marcelo da Cunha Cavalcanti.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Departamento de Computação.

1. Arquitetura de software. 2. Computação em nuvem . 3. Internet das coisas . 4. Código de barras. I. Cavalcanti, Carlos Frederico Marcelo da Cunha. II. Universidade Federal de Ouro Preto. III. Titulo.

CDU: 004.27

Catalogação: ficha.sisbin@ufop.edu.br

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

Desenvolvimento de sistemas escaláveis e na nuvem - estudo de caso

LUIZ WALBER DE SOUSA FREITAS FILHO

Monografia defendida e aprovada pela banca examinadora constituída por:

Dr. Carlos Frederico Marcelo da Cunha Cavalcanti – Orientador
Universidade Federal de Ouro Preto

Dr. Ricardo Augusto Rabelo Oliveira
Universidade Federal de Ouro Preto

M.Sc. Vinicius Antonio de Oliveira Martins
Universidade Federal de Ouro Preto

Ouro Preto, Dezembro de 2019

Resumo

Este trabalho apresenta uma alternativa para controle de segurança em portas, substituindo o sistema tradicional de chave e acoplando maior segurança a estas barreiras. Com o avanço da tecnologia, o celular vem se tornando cada vez mais importante na vida das pessoas, muitos sistemas estão sendo criados e centrados ao redor do celular, em alguns casos o celular consegue até mesmo substituir totalmente sua carteira, podendo até mesmo realizar pagamentos sem cartão com ele. Com todo este desenvolvimento é difícil encontrar alguém que saia de casa sem o celular, por isso seria interessante que o celular pudesse substituir o sistema de chaves da casa, trazendo maior praticidade, com apenas um dispositivo poderíamos pagar contas, entrar em casa, além de realizar as funções já realizadas. O celular geralmente possui sistema de segurança para desbloqueio, então mesmo que você o perca a pessoa que o encontrar não conseguira entrar em sua casa, diferentemente da chave. Outra falha de segurança das portas é a própria fechadura que pode ser facilmente arrombada utilizando dispositivos simples, o sistema elétrico de autenticação na nuvem dificultaria os arrombamentos. Este trabalho desenvolverá um protótipo para este sistema, tentando resolver os problemas citados anteriormente, e mantendo a autenticação para abrir a porta na nuvem como quesitos de segurança. Também será utilizado a tecnologia Qr-Code para autenticar o celular do usuário .

Palavras-chave: *Arquitetura de Software*. *RFID*. Computação na nuvem. Internet das Coisas. Desbloqueio de Portas. *QR-Code* . Micro-Serviços. Cliente Servidor. Segurança.

Abstract

This work presents an alternative way to control security in doors, replacing the traditional key system and attaching greater security to these barriers. With the advancement of technology, cell phones are becoming increasingly important in people's lives. Many systems are being created centered around the cell phone, in some cases he can even completely replace your wallet, and you can make cardless payments with it. With all this development it is difficult to find someone who leaves home without this device, so it would be interesting that the cell phone could replace the key system of the house. This would be more practical, with only this device we could pay bills, unlock our home , in addition to the usual functions already performed. As we know the cell phone usually has a security system to unlock, so even if you lose it the person who found you could not enter your home, different from the keys, another security flaw of the doors is the lock itself which can be easily broken with simple devices, making the electrical system and keeping the authentic action in the cloud would make it more difficult to break. This work will develop a prototype for this system trying to solve the problems mentioned above, keeping the authentication to open the door in the cloud for security reasons and using Qr-Code technology to authenticate the user's cell phone.

key-words: Software Architecture. RFID. Cloud computing. Internet of Things. Unlocking doors. QR code. Micro Services. Client-Server. Security.

Ao Curso de ciência da computação da Universidade Federal de Ouro Preto, e às pessoas com quem convivi neste espaço. A experiência adquirida no decorrer desses longos anos eu com certeza levarei para o resto da minha vida

Agradecimentos

Existem muitas pessoas a quem desejo agradecer, nem todas serão possíveis de citar neste texto, porém não quer dizer que elas tenham sido menos importantes para meu desenvolvimento e o desenvolvimento deste trabalho. Agradeço meu pai Luiz, minha mãe Regina, e minhas irmãs Isis, Luysa e Júlia por todo esforço que eles fizeram em minha criação, e todo apoio que eles me deram durante toda minha vida, agradeço minhas amadas sobrinhas Luana Gabriela e bruna, e agradeço aos amigos pelos momentos de confraternizações e por ter contribuído muito ao meu desenvolvimento pessoal e profissional, não poderia deixar de agradecer à república Xeque Mate por ter me acolhido no meu último ano, se tornando uma segunda casa para mim, também agradeço o professor Carlos Frederico que contribuiu muito para que esse projeto fosse realizado.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

1.1 Contextualização e Problematização

Carregamos diversos objetos tais como celular, carteira, chaves, cartões, fones de ouvido, *tablets*, carregadores e outros dispositivos. É também comum possuirmos um número considerável de chaves, como por exemplos chaves de casa, do carro, do trabalho, da moto, da garagem e de diversas outras coisas. Estes objetos ocupam muito espaço, e podem ser perdidos ou roubados. Caso isto aconteça, poderá haver o acesso não autorizado de terceiros. Este trabalho é uma proposta de uma solução para amenizar este problema, aumentando a segurança e reduzindo o número de objetos portados por uma pessoa.

Nesta era de transformações rápidas, a fechadura baseada em chave mecânica, amplamente usada em aplicações residenciais, tem sido incrivelmente resistente à mudanças como constata Click (2015), apesar do constante e rápido avanço tecnológico experimentado nas últimas décadas. Podemos verificar que devido a este grande avanço, a sociedade vem sendo transformada continuamente impondo o uso de novos dispositivos e gerando novos comportamentos. Um dos exemplos mais contundente é o uso massivo de aparelhos celulares, dispositivo esse já incorporado como um item essencial no cotidiano de grande parte da população. A Anatel (2015) estima que, no Brasil, atualmente existem 284.15 milhões de linhas ativas na telefonia móvel e uma densidade de 139.16 acessos por 100 habitantes. Com esse número elevado de celulares, vão surgindo mais aplicativos com o intuito de facilitar a vida das pessoas no trabalho, no lazer e na interação com o mundo.

O trabalho desenvolvido propõe a criação de um sistema de autorização de acessos a locais que tenham acesso restrito a um número de pessoas, a exemplo de apartamentos, prédios, locais de trabalho e outros locais. Este sistema consistirá, para o usuário, em um aplicativo de celular que substituirá as chaves mecânicas utilizadas no cotidiano. Também, o sistema será escalável, podendo funcionar em inúmeros locais ao mesmo tempo. Com esse aplicativo pretende-se reduzir o número de chaves carregadas por uma pessoa, substituindo por um mecanismo de funcionalidade similar usando um dispositivo móvel (celular). Isto adiciona uma

camada de segurança pois, por exemplo, para abrir uma porta, a pessoa autorizada deverá desbloquear seu celular autenticando-se em primeiro nível, validando-se como usuário no aplicativo, autenticando-se em segundo nível e, tendo sucesso em ambas autenticações, deverá ter autorização para perfazer a operação desejada que, neste exemplo, a operação de abrir a porta. O sistema é modular permitindo que futuramente seja adicionado outros métodos de validação.

1.2 Justificativa

Com a chegada da indústria 4.0 uma nova fase na história industrial foi iniciada. Esta revolução baseia-se na conexão de máquinas, sistemas e pessoas ao processo produtivo. Cada vez mais *Cloud Computing* está se popularizado, que nos possibilita colocar recursos computacionais na Internet Peter Mell (2011) com uma facilidade, rapidez e confiabilidade inimagináveis a uma década atrás. Isto permite que sistemas "na nuvem" possam ser acessados por inúmeros usuários ao redor de todo o mundo, bastando para isso ter acesso a internet. Nesta abordagem, a quantidade média de acessos que um servidor recebe cresce constantemente e a preocupação com a segurança dos dados na nuvem se torna uma das principais preocupações da comunidade de TI.

Considerando o grande potencial de crescimento desta nova fase industrial, muitas pessoas estão investindo em estudos e desenvolvimento de sistemas relacionados ao tema. O mercado vem se adaptando cada vez mais para receber as novas tecnologias, criando oportunidades para o surgimento de novas empresas. Diariamente são lançados diversos aplicativos e dispositivos utilizando essas novas tecnologias. Considerando este cenário em constante crescimento, este trabalho objetiva também contribuir para arquiteturas que suportam esta nova geração de aplicativos.

1.2.1 Objetivo do trabalho

Este trabalho possui dois principais objetivos: o primeiro deles é a criação de um sistema para gerenciar acessos a locais específicos e isto será realizado por meio de um aplicativo instalado e autorizado previamente no celular do usuário final. O segundo objetivo é o planejamento de uma arquitetura para que este sistema funcione na nuvem, tornando-o escalável,robusto e com menor número de vulnerabilidades contra ataques de segurança.

1.2.2 Objetivo geral

O objetivo geral deste trabalho é desenvolver um protótipo de aplicação que envolve diversas áreas com finalidade de prover uma melhor segurança física e maior praticidade no dia a dia, demonstrando também as vantagens de se criar sistemas escaláveis.

1.2.3 Objetivos específicos

- Realizar pesquisas para escolher as melhores ferramentas para o desenvolvimento da aplicação;
- Realizar pesquisas para geração de embasamento teórico e revisão bibliográfica sobre as tecnologias que serão utilizadas na confecção do protótipo;
- Desenvolver uma arquitetura em nuvem escalável, robusta e segura;
- Desenvolver uma aplicação utilizando os conhecimentos adquiridos;
- Obter capacitação pessoal através de novas técnicas.

Capítulo 2

Trabalhos Relacionados

Ao longo dos últimos anos foram desenvolvidos diversos trabalhos relacionados ao tema em questão utilizando tecnologias diferentes, mas com o mesmo intuito do trabalho desenvolvido nesta monografia. Este capítulo tem como objetivo fazer uma breve revisão sobre as últimas abordagens, citando e explicando brevemente sobre outros aplicativos que utilizam funcionalidades similares.

Lockitron (Figura 2.1) Apigy (2018) é um sistema *mobile* de controle de portas, onde se pode abrir, fechar ou verificar se uma porta encontra-se aberta mesmo que o usuário esteja a distância. O sistema permite que chaves (permissões de acesso) sejam distribuídas para usuários, que seja feito o gerenciamento dos acessos permitindo ou negando o acesso de cada pessoa individualmente e também provê o registro (*log*) de todas as entradas conectadas, permitindo complexos relatórios como: quem entrou em casa, que horas entrou e outros detalhes.

Figura 2.1: Lockitron
Apigy (2018)

O hardware do *Lockitron* é dividido em dois principais componentes: ''Bolt'' e ''Bridge'', o Bolt é responsável por travar e destravar a porta fisicamente, já o Bridge é responsável por realizar as conexões com a internet e enviar o sinal de abrir ou fechar para o Bolt. *Lockitron* também possui uma API simples e customizável onde podemos acessar diversos *endpoints* para personalizar o sistema.

August smart lock August (2018) é um sistema de controle de porta à distância. Com *August* é possível desbloquear uma porta através de um comando utilizando para isso um *smart phone* conectado à Internet de qualquer lugar do mundo. Alternativamente, pode-se também configurar o telefone para, quando o usuário se aproximar da porta, ela possa abrir automaticamente utilizando a tecnologia *bluetooth*. *August* gerencia permissões de acesso possibilitando a criação de permissões para grupo, a exemplo de grupos com permissões diferentes para familiares ou amigos, a criação de chaves permanentes ou temporárias, a administração das permissões de cada usuário ou grupo e possibilita a visualização de todos os acessos em tempo real ou em registros. Isto permite, entre outras coisas, monitorar o estado da porta, isto é, se encontra aberta ou fechada e quem liberou / trancou a porta.

Figura 2.2: August Smart Lock
August (2018)

August smart lock possui fácil instalação podendo ser realizada em minutos em quase todos os modelos de portas mais comuns. Também, pode ser integrada com a Alexa, com a Siri ou com o Google assistente. Outro recurso interessante deste sistema é a câmera disponível no modelo mais completo. Esta câmera possui sensor de movimento e saída de áudio permitindo que, se alguém aproxima de sua porta, o sensor de movimento detecta a pessoa e manda uma

notificação para celular do usuário, mostrando o video e o áudio de quem está na porta. Isto permite o usuário conversar com visitante e comandar a abertura da porta à distância. Este sistema também disponibiliza a gravação de todos os momentos que o sensor de movimento foi ativado, de forma que é possível assistir essas gravações posteriormente.

Capítulo 3

Referencial Teórico

Diversas áreas, conceitos e ferramentas estão envolvidas na confecção do protótipo objeto para este trabalho. Neste capitulo iremos introduzir os fundamentos que baseiam este trabalho.

3.1 Indústria 4.0

O mundo está cada vez mais conectado e estamos entrando em uma nova revolução industrial denominada Industria 4.0. Segundo Bahrin et al. (2016) a indústria 4.0 é uma nova estrutura de fabricação sob a forma de CPPS *(Cyber-Physical Production System)*, que visa transformar o sistema manufatureiro, tornando a troca de informações intensa e constante, digitalizando o setor. Nesta revolução, a Internet das coisas ou simplesmente "IoT", anacronismo da expressão inglesa *(Internet of Things)*, desempenha um importante papel sendo a ferramenta potencial para alimentar informações em tempo real e agregar valor à indústria de manufatura. A implementação do conceito da Industria 4.0 proporcionará uma mistura de baixo volume e alta produção de uma forma rentável.

3.2 Internet of Things (IoT)

A Internet das coisas é um conceito relativamente novo, que está em rápido crescimento e usado amplamente nos dias atuais. Porém, apesar de seu uso estar cada vez mais comum, segundo Wortmann e Flüchter (2015) ainda não possuímos uma definição formal para quais áreas abrangem o IoT. A origem do termo foi atribuída ao trabalho no MIT (*Massachussets Institute of Technology*) em infraestruturas RFID, iniciais de *Radio Frequency Identification*, e, nos dias atuais, possui uma abrangência que vai além além de um dispositivo RFID. Boa parte das definições focam nas "coisas", que neste caso são sinônimos de "dispositivos", que são conectados na Internet e outras abordagens focam nos protocolos e arquiteturas que viabilizam conectar estes dispositivos na Internet.

Ainda segundo Wortmann e Flüchter (2015) as aplicações para IoT são diversas e possuindo um destaque especial na Industria 4.0, viabilizando a construção de produtos, casas, veículos, indústrias, auxiliando o trânsito e em muitos outras "coisas" e situações que são conectadas, monitoradas e controladas pela Internet. A utilização de IoT agrega valor e amplia os ambientes virtuais possuindo assim uma coleta mais precisa e com maior quantidade de dados para tomada de decisões mais inteligentes.

3.3 Radio-Frequency Identification (RFID)

Neste novo momento, em que a Internet das Coisas está cada vez mais popular, existe uma preocupação muito grande em fazer o mundo físico se ligar com o mundo virtual. Podemos ver um crescimento de produtos já disponíveis para o consumidor final com este conceito, como geladeiras inteligentes que controlam o próprio estoque solicitando novos produtos quando estiver próximo a acabar, carros que não precisam de motoristas, pontos de ônibus que informam o tempo aproximado para o ônibus chegar, entre outros exemplos.

Existem diversas formas para fazer ligação entre o mundo físico e o virtual, sendo o RFID uma forma. De acordo com Glover e Bhatt (2007) RFID é uma tecnologia *wireless* (sem fio) que tem como objetivo a coleta e transmissão automática de dados. Esta tem como intuito ligar o mundo físico ao mundo virtual por meio de pequenos dispositivos, que podem ser acoplados às "coisas"no mundo físico. Este dispositivo é um transpônder, um dispositivo de comunicação eletrônico que envia, recebe, amplifica e retransmite um sinal, possuindo um identificador único no mundo virtual.

Segundo Mota (2006), RFID é uma tecnologia que utiliza ondas de radiofrequência para transmitir dados e é composto por três principais componentes: *tag* RFID, leitor RFID e banco de dados. Como mostrado na Figura 3.1, alguns autores dizem que basta apenas a *tag* e o leitor para o sistema RFID, porém todos citam a importância do banco de dados para um sistema completo.

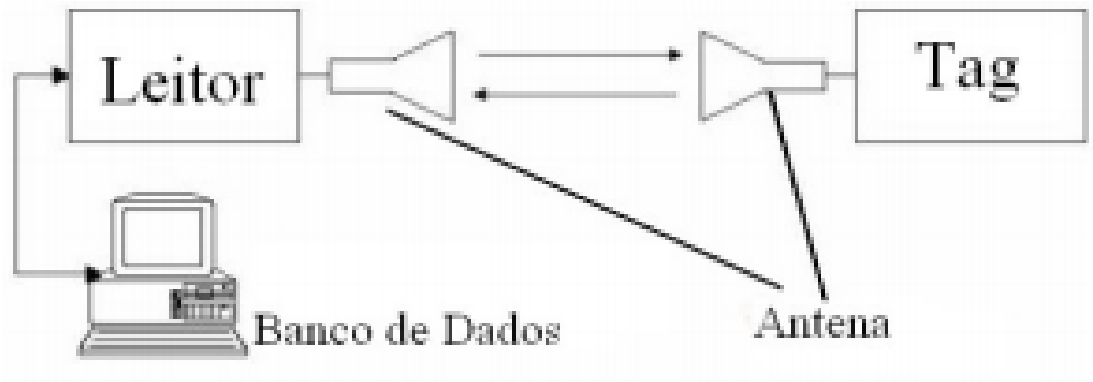


Figura 3.1: Componentes do sistema RFID

3.4 Raspberry Pi

O idealizador desta arquitetura foi a Foundation (2018) que classificou *Raspberry PI* (Figura 3.4) como um dispositivo pequeno (aproximadamente do tamanho de um cartão de crédito) e de baixo custo que mostrou-se, desde a sua criação bastante versátil. Amplamente utilizado para ensinar programação à crianças, este potente sistema é capaz de fazer tudo que podemos esperar de um computador desktop, desde navegar na internet, mostrar vídeos de alta definição, ou até mesmo rodar jogos.

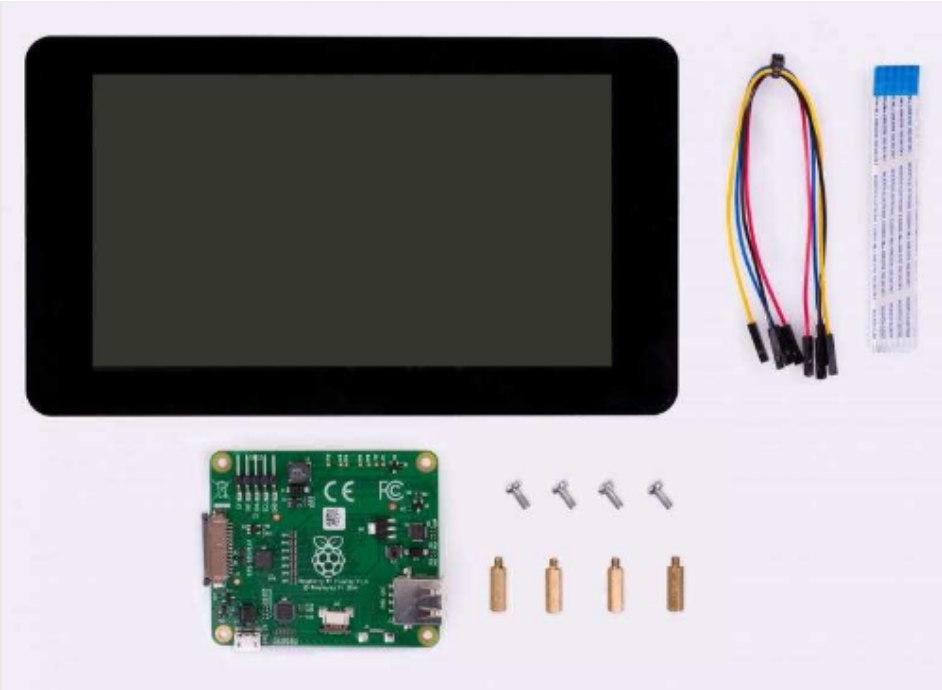


Figura 3.2: Dispositivo Raspberry Pi com Visor
Foundation (2018)

3.5 Cloud Computing

Segundo Anjomshoaa e Tjoa (2011), uma definição comumente aceita de *cloud Computing* (Computação nas Nuvens) é: um modelo para permitir acesso de rede conveniente e sob demanda a recursos compartilhados e configuráveis. Estes recursos podem ser provisionados rapidamente e lançados com esforço mínimo de gerenciamento ou interação com o provedor de serviços. *Cloud Computing* é considerada muitas vezes como a próxima etapa evolutiva lógica de tecnologias. Anjomshoaa e Tjoa (2011) também dizem que as vantagens tecnológicas e comerciais exclusivas desse paradigma estão se tornando cada vez mais evidentes e espera-se que nos próximos anos a migração da nuvem aumente radicalmente, e dentro de 10 anos a maioria dos serviços de TI sejam entregues via nuvens públicas ou privadas.

Já de acordo com o NIST (*National Institute of Standards and Technology*) Peter Mell (2011), *Cloud Computing* é um modelo que permitir acesso onipresente, conveniente e sob demanda da rede a um conjunto de recursos computacionais configuráveis que podem ser rapidamente provisionados e liberados com o mínimo de esforço de gerenciamento ou interação do provedor de serviços (redes, servidores, armazenamento, aplicativos, serviços e outros). O modelo *cloud* é composto por cinco características essenciais, tres modelos de serviço e quatro modelos de *deployment* (processo de *upload* dos recursos para a nuvem)

3.5.1 Características Essenciais

As seguintes características são consideradas essenciais:

On-demand self-service: O consumidor pode fornecer recursos unilateralmente, sem a necessidade de iteração com cada provedor de serviços;

Broad network access: Os recursos estão disponíveis na rede e podem ser acessados por diversas plataformas heterogêneas (telefones, computadores, tablets, etc);

Resource pooling: Os recursos computacionais são agrupados para atender vários consumidores usando um modelo multi locatário, com diferentes recursos físicos e virtuais dinamicamente atribuídos e reatribuídos de acordo com a demanda. O cliente geralmente não tem controle ou conhecimento sobre o local exato dos recursos fornecidos, mas pode ser capaz de especificar um local em mais alto nível (por pais, estado, *datacenter* e outros);

Rapid elasticity: Os recursos podem ser provisionados e liberados elasticamente e, em alguns casos, automaticamente e expandindo para se adequar à demanda;

Measured Service: Os sistemas em nuvem controlam e otimizam automaticamente o uso de recursos, aproveitando uma capacidade de medição em algum nível de abstração apropriado ao tipo de serviço. Os usos dos recursos podem ser monitorados, controlados e relatados, fornecendo transparência para o fornecedor e consumidor do serviço utilizado.

3.5.2 Modelos de Serviços

Software as a Service (SaaS): O consumidor tem permissão de acessar dados do provedor em execução em uma infraestrutura em nuvem e os aplicativos são acessíveis pelo cliente por meio de um programa ou diretamente pela Internet. O usuário não gerencia nem controla a infraestrutura, exceto em possíveis exceções de definições, limitadas nas configurações de aplicativos específicos do usuário.

Infrastructure as a Service (IaaS): O consumidor pode acessar recursos computacionais, podendo implantar e executar código arbitrário, porém não gerencia ou controla a nuvem subjacente à infraestrutura, mas tem controle sobre sistemas operacionais, armazenamento, aplicativos implantados e possivelmente controle limitado dos componentes de rede selecionados como por exemplo *firewalls*.

Platform as a Service (PaaS): O consumidor pode implantar na nuvem aplicativos criados ou adquiridos na infraestrutura, porém não tem controle sob a infraestrutura, incluindo rede, servidores, *firewalls*, sistemas operacionais ou de armazenamento, mas tem controle sobre os aplicativos implantados e possivelmente definições de configuração para o ambiente de hospedagem de aplicativos.

3.6 Infraestrutura na nuvem: *Amazon Web Services*

Empresas do mundo todo estão migrando para uma infraestrutura baseada em nuvem para aumentar a agilidade de TI, obter alta escalabilidade, melhorar a confiabilidade e reduzir custos. Estas empresas procuram por flexibilidade para expansões, sem precisar se preocupar em configurar uma infraestrutura nova, fazendo com que eles possam aprimorar suas experiencias com o usuário final e diminuindo a latência e a qualidade da conexão Amazon (1994).

O *Amazon Web Services (AWS)* é uma das plataformas em nuvem das mais abrangentes e adotadas do mundo, oferecendo mais de 165 serviços misturados entre os modelos de IaaS, PaaS e SaaS. Escolhemos AWS pela facilidade de uso e conveniência.

3.6.1 Benefícios

Os seguintes beneficios são encontradas em plataformas em nuvem atualmente ofertadas no mercado:

Performance: Performance e baixa latência são uma das principais preocupações que deve ser considerado ao escolher uma plataforma em nuvem. Isto dá uma sensação de rapidez e velocidade para o cliente.

Availability: As zonas de disponibilidade são *clusters* de recursos organizados de tal forma a trazer confiabilidade sendo projetadas com redundância física e oferecendo resiliência. Isto permite um provimento de serviço ininterrupto mesmo em casos de falta de energia em uma determinada localidade, inatividade na internet em alguns *clinks* ou até mesmo em casos de desastres naturais.

Security: A infraestrutura na nuvem é desenvolvida para atender os requisitos de segurança rigorosos e é monitorada 24/7 para garantir a confiabilidade, integridade e a disponibilidade dos dados aos seus clientes.

Reliability: A infraestrutura na nuvem é criada para oferecer redundância, confiabilidade e resiliência máxima contra interrupções do sistema. Provedores de infraestrutura na nuvem constroem seus *datacenters* em várias regiões geográficas e em várias zonas isoladas umas das outras, cada qual distantes quilômetros de distância uma da outra.

Scalability: A capacidade de computação dos provedores de infraestrutura na nuvem cresce em uma alta taxa, oferecendo flexibilidade e escalabilidade virtualmente infinita. As empresas clientes podem aumentar rapidamente os recursos conforme o necessário.

Low Cost: A infraestrutura na nuvem permite que os clientes se beneficiem da economia de escala e reduzam o custo total de propriedade da infraestrutura geral de TI. Relatório de pesquisa da IDC de 2018 indica que os clientes podem obter importantes benefícios financeiros que ajudam a aumentar o crescimento, impulsionar eficiências e obter importantes reduções de custo a longo prazo, incluindo: redução de 51% dos custos em 5 anos, eficiência 62% maior e 90% menos tempo para implantar um novo armazenamento.

3.7 *Quick-Response Code* (QR-Code)

De acordo com Soon (2008) QR-Code (sigla do inglês *Quick Response Code*) é um simbolo bidimensional que foi inventado em 1994 pela Denso Wave. Em das principais empresas do grupo Toyota, o QR-Code foi aprovado como padrão ISO internacional (ISO/IEC18004). Este simbolo foi inicialmente criado para controle de produção de peças automotivas, mas se espalhou para outros campos por vários motivos, sendo alguns deles:

- Densidade de dados muito maior a outros códigos 2D, podendo representar caracteres chineses e japoneses.
- Pode ser utilizado gratuitamente por qualquer pessoa, pois a Denso lançou a patente em domínio publico.
- O padrão de estrutura de dados não é pré-requisito para usos atuais.
- A maioria dos telefones celulares equipados com câmeras permitem a leitura de QR-Code.

3.7.1 Características

Além do grande volume de dados (7.089 caracteres), gravação de alta densidade (aproximadamente 100 vezes maior que símbolos lineares) e leitura em alta velocidade, o QR-Code possui outras características superiores como:

- Leitura de qualquer direção (360º) e alta velocidade;

- Resistência a símbolos distorcidos;
- Resistência a danos nos símbolos;
- Eficiência na codificação de caracteres Kanji e Kana;
- Vinculo entre símbolos;
- A confidencialidade do código.

3.7.2 Estrutura

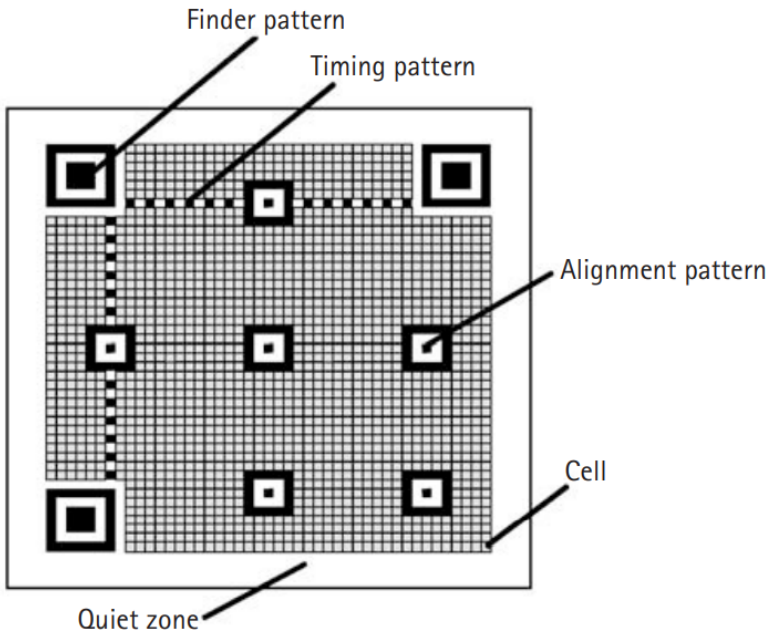


Figura 3.3: Estrutura do QR-Code
Soon (2008)

Como mostrado na Figura 3.3, existem vários marcadores na estrutura do QR-Code para auxiliar a identificação e processamento do mesmo, e eles consistem em:

***Finder pattern*:** Marcadores criados para localizar o QR-Code de qualquer direção, eles possuem proporções específicas entre preto e branco facilitando a deteção da largura e angulo e tornando a velocidade de leitura do QR-Code aproximadamente 20 vezes mais rápida;

***Timing pattern*:** Um padrão para identificar a coordenada central de cada célula, é utilizado para corrigir quando o simbolo esta distorcido ou quando existe um erro para o *pitch* da célula;

***Alignment pattern*:** Marcadores para corrigir as destorções do QR-Code;

***Cell*:** Células do QR-Code que serão transformadas para binário posteriormente, com o valor 1 representando o preto e 0 representando o branco;

***Quiet zone*:** Zona nula que com 4 ou mais células para facilitar na detecção das bordas do QR-Code.

3.8 Android

Em seu livro, Lucio Camilo Oliva Pereira (2009) explicou que o Android é uma plataforma móvel completa com sistema operacional, *middleware*, aplicativos e interface de usuário incluso. Esta plataforma foi desenvolvida em código aberto *(open source)*, possibilitando o acesso dos desenvolvedores a qualquer funcionalidade do núcleo do sistema.

Apesar de não apresentar todos os padrões Linux, esta plataforma foi construída em um kernel Linux, tendo assim uma forte capacidade de segurança através do sistema de permissionamento, só sendo possível que uma aplicação acesse as outras se possuírem a permissão explicitamente declarada. Por isto, toda aplicação possui uma chave privada que só é conhecida pelo desenvolvedor para estabelecer relações de confiança entre aplicações.

3.8.1 Recursos

Os seguintes recursos são parte da arquitetura Android:

Framework de Aplicação: Permite a incorporação, reutilização e a substituição de recursos de componentes;

Maquina Virtual Dalvik: Maquina virtual otimizada para dispositivos movéis;

Navegador Web Integrado: Navegador baseado na *engine WebKit*;

Gráficos Otimizados: Gráficos Otimizados por meio de uma biblioteca baseados na especificação Open-GL ES 1.0;

SQLite: Gerenciador de banco de dados;

Suporte Multimídia: Suporte de som, video e imagens nativamente;

Telefonia GSM: Permite fácil integração com tecnologias GSM;

Protocolo de Comunicação Wireless: Suporte a tecnologia bluetooth, Wi-Fi, 2/3/4 G, com os mais recentes, tendo suporte ao 5G;

Camera, GPS, bússola e acelerômetro: Fácil integração com hardware para diversos recursos;

Poderoso Ambiente de Desenvolvimento: Vários ambientes de desenvolvimento que emulam, depuram, analisam a memoria e performance.

3.8.2 Arquitetura

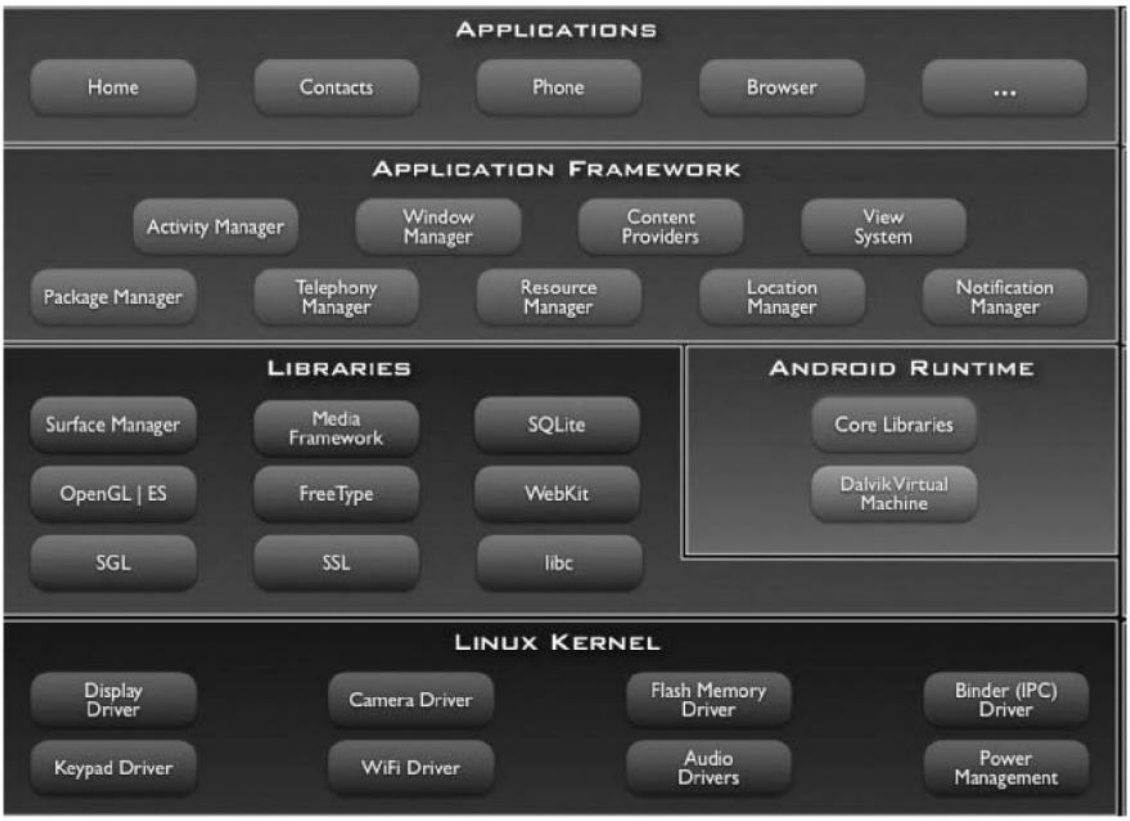


Figura 3.4: Arquitetura da plataforma Android
Lucio Camilo Oliva Pereira (2009)

Na figura 3.4 é apresentada a arquitetura Android e ela é composta por:

3.8.2.1 Aplicações (Applications)

A camada mais exterior do Android, onde é localizada todos os aplicativos do dispositivo móvel. As aplicações utilizam o Java como linguagem de desenvolvimento padrão que posteriormente serão executados em uma maquina virtual Dalvik.

3.8.2.2 Framework

Na camada de Framework se encontra todos os recursos utilizados pelas aplicações, existem inúmeros recursos que as aplicações podem acessar, sendo os principais:

Activity Manager: Gerencia o ciclo de vida das atividades;

Package Manager: É a camada que comunica com o resto do sistema e diz quais os pacotes sendo utilizados e suas capacidades;

Window Manager: Gerencia as apresentações de janelas;

Content Providers: Possibilita o compartilhamento de dados entre aparelhos e aplicativos;

View System: Disponibiliza o tratamento gráfico para a aplicação.

3.8.2.3 *Libraries*

A camada de bibliotecas consiste em um conjunto de códigos C/C++ utilizadas pelo sistema, estas bibliotecas auxiliam na multimídia, visualização 2D e 3D, aceleração de hardware, acesso ao banco de dados e entre outras. Todos estes recursos estão disponíveis no *framework* para o desenvolvimento de aplicativos.

3.8.2.4 *Android Runtime*

Esta camada consiste em uma instância da maquina virtual Dalvik, que foi criada para maior desempenho em sistemas com baixa frequência de *clock* de CPU, pouca memória RAM e consumo mínimo de bateria e memória. Dentro desta camada também temos o *Core Libraries*, que inclui um grupo de bibliotecas que fornece a maioria das funcionalidades disponíveis nas principais bibliotecas da linguagem Java.

3.8.2.5 *Linux Kernel*

Para os serviços centrais do sistema utiliza-se o kernel do Linux, ficando responsável pela segurança, gestão de memoria, processos, pilha de protocolos e modelos de *drivers*. Nesta camada também se encontra um poderoso sistema próprio de gerenciamento de energia.

3.9 Arquitetura de Software

Segundo Nenad Medvidovic (2010), no coração de todo software bem projetado se encontra uma boa arquitetura de software e um bom conjunto de decisões do projeto. A arquitetura de software estuda como os sistemas e softwares são projetados e construídos durante todo seu ciclo de vida e, dificilmente, iremos encontrar um sistema de boa qualidade com uma arquitetura mal projetada.

A arquitetura de software se manifesta em todas as etapas do software, incluindo componentes estruturais, configurações, decisões de *deploy*, melhorias, e até mesmo requisitos não funcionais (reuso, confiança, escalabilidade, segurança, eficiência entre outros).A arquitetura

deve estar em primeiro plano no desenvolvimento de um sistema, sendo mais importante que a análise ou a programação, Nenad Medvidovic (2010) diz que apenas dando a atenção adequada à arquitetura de software que podemos criar um sistema robusto, com planejamento para evoluções e efetivo.

Existem vários padrões de arquiteturais de software, cada um para solucionar um problema especifico com suas vantagens e desvantagens. Neste tópico sera apresentado brevemente os padrões utilizados para confecção do protótipo desejado.

3.9.1 Cliente-Servidor

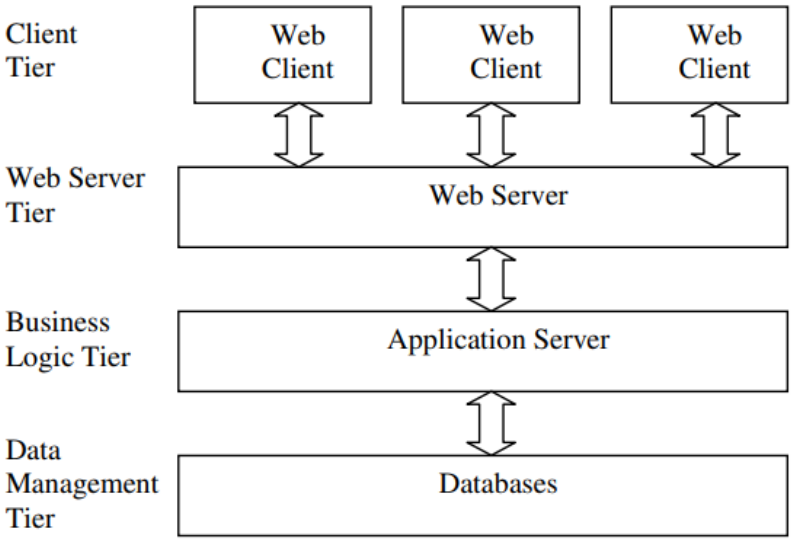


Figura 3.5: Exemplo Arquitetura Cliente-Servidor de N Níveis
Gorton (2011)

A arquitetura Cliente-Servidor é um modelo distribuído que divide as tarefas entre consumidores de um serviços (Clientes) e o(s) provedor(es) (Servidores). Em seu livro, Gorton (2011) citou vários modelos da arquitetura Cliente-Servidor e, neste trabalho, abordaremos apenas a arquitetura de N níveis (Figura 3.5) pois se adéqua melhor ao tema. Gorton (2011) recomendou esta arquitetura para aplicações com um potencial grande de clientes e requisições concorrentes, sendo que cada requisição tenha uma resposta em um curto período de tempo para processar. Esta arquitetura possui as vantagens:

3.9.2 Arquitetura de Micro serviços

De acordo com Chandramouli (2019), a arquitetura de micro serviços consiste em um sistema dividido em vários componentes, como apresentado na figura 3.6. Eles normalmente implementam uma funcionalidade específica, tendo as lógicas de negócio separadas nos serviços. Estes componentes são expostos por meio de *endpoints API* (Interface de Programação

Tabela 3.1: Características arquitetura de N niveis

Atributos	Característica
Disponibilidade	Os servidores em cada nível podem ser replicados para que algum assuma em caso de mal funcionamento, fazendo com que a aplicação continue funcionando mesmo com a queda de um serviço
Resposta a Falhas	Se um cliente esta comunicando com um servidor que falhou, sua requisição é redirecionada para um servidor ativo que possa responder sem o conhecimento do cliente.
Performance	Esta arquitetura comprovadamente tem alta performance
Escalabilidade	Os Servidores podem ser replicados e multiplicados em diversas instancias, podendo rodar em varias maquinas concorrentemente

de Aplicativos). APIs existem com vários tipos de *endpoints*, como SOAP (Protocolo Simples de Acesso a Objetos) ou REST (protocolo HTTP). Na arquitetura de micro-serviços cada entidade possui uma separação para cuidar da regra de negocio chamada serviço que serão acessados pelo cliente através da API.

3.9.2.1 Características principais:

- Cada micro-serviço deve ser controlado, escalado, atualizado e lançado na nuvem independente dos outros micro-serviços;
- Cada micro-serviço deve ter apenas uma única função, tendo responsabilidade limitada e dependência dos serviços;
- Cada micro-serviço deve ser desenvolvido para ser constantemente tolerante a falha;
- Os micro-serviços podem utilizar serviços existentes confiáveis.

3.9.2.2 Características principais para empresas

Acesso Onipresente: Os usuários acessam a aplicação de vários dispositivos diferentes (tablets, smartphones, computador etc);

Escalabilidade: Os aplicativos devem ser escalonáveis para manter a disponibilidade em face de um grande numero de usuários;

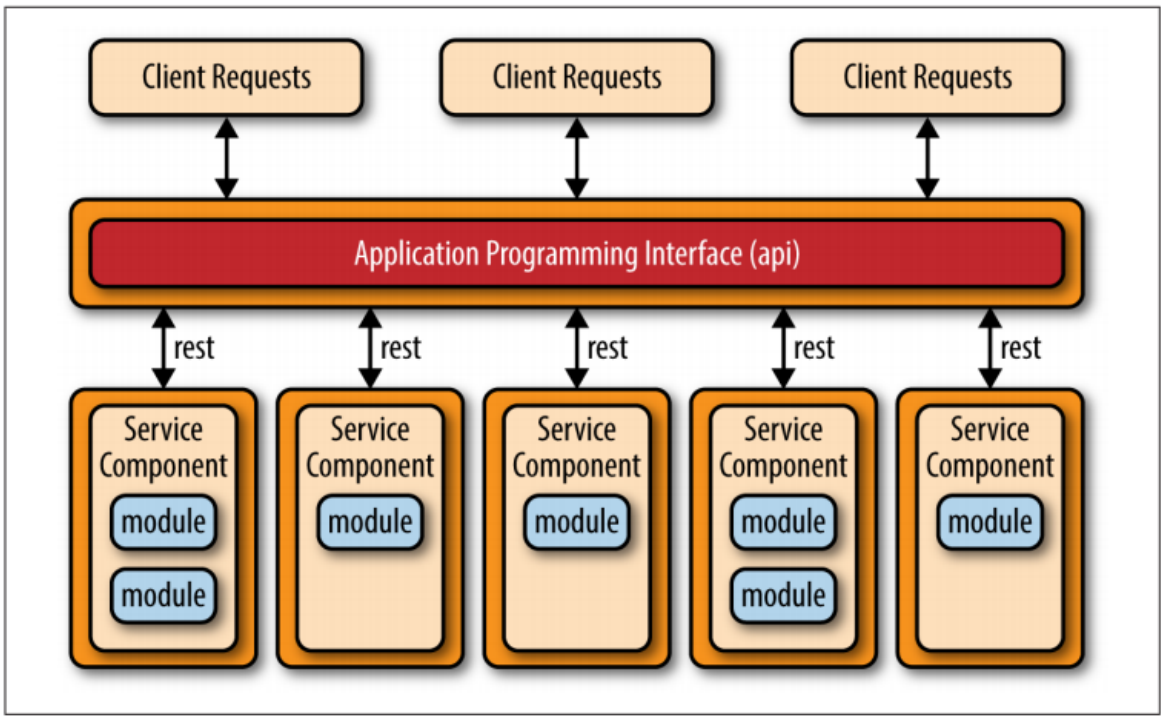


Figura 3.6: Topologia da arquitetura Micro-serviços baseada em API REST
Chandramouli (2019)

Desenvolvimento Ágil: As organizações querem atualizações constantes para responder rapidamente a mudanças.

3.9.2.3 Vantagens

- Autonomia;
- Baixo Acoplamento;
- Reuso;
- Tolerância a erros.

3.9.3 *Model-View-Controller* (MVC)

De acordo com Freeman (2013), O MVC é uma arquitetura normalmente utilizada para organizar o desenvolvimento da interface com o usuário, como mostrado na figura 3.7. O MVC divide a aplicação em 3 camadas, sendo elas:

***Model*:** A camada de modelo é responsável por armazenar os dados, algumas vezes ela cuida das alterações nestes dados, porem normalmente só armazena;

***View*:** A camada de visualização cuida da apresentação dos componentes para o usuário;

***Controller*:** A camada de controle é responsável por controlar as trocas de informações e lógicas de negócio.

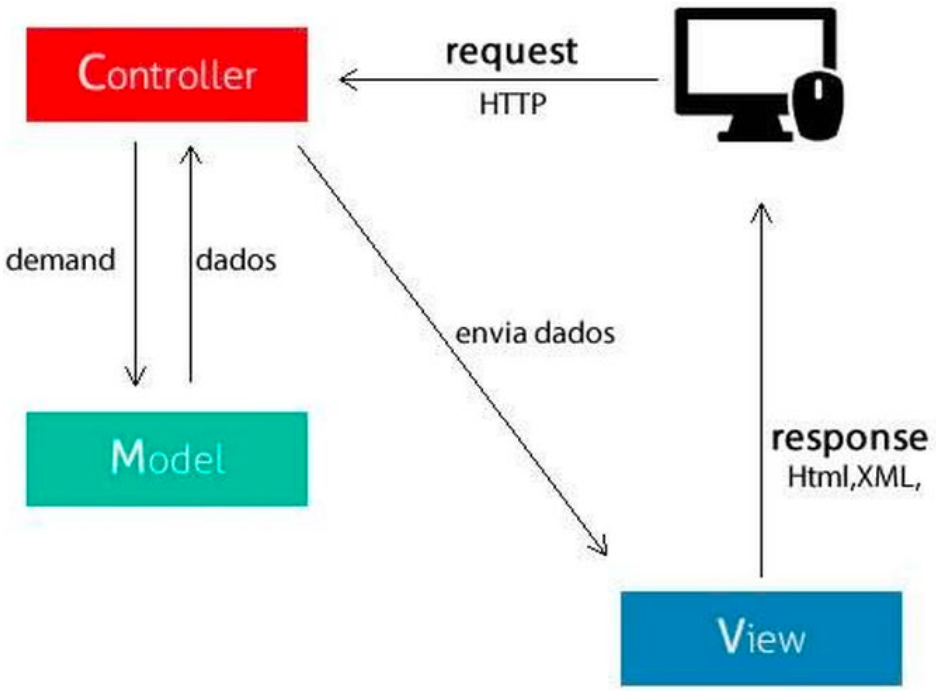


Figura 3.7: Arquitetura MVC
Ramos (2015)

Como apresentado na figura 3.7, o Controller é o responsável por receber as requisições REST. Com esta requisição, ele aplica as regras de negócio requisitando e alterando os dados da camada de Model. Logo após o Controller envia uma resposta para a camada de View que sabe como apresentar e responder esta requisição para o cliente.

3.10 Linguagens e Tecnologias

Nesta sessão será apresentada brevemente sobre as linguagens e tecnologias utilizadas para criação do protótipo desejado que consistem em:

Java: Java é uma linguagem de programação orientada a objetos, compilada e que gera uma linguagem intermediária, um *bytecode*, que é interpretada por uma máquina virtual JVM. Programas Java executando em sua máquina virtual foram idealizados para serem multiplataformas, isto é, programas Java rodam virtualmente em qualquer sistema operacional. Para isto, Java nativamente possui recursos que facilitam a cooperação en-

tre processos Java usando protocolos como HTTP e tem recursos de segurança Oracle (2019).

Spring: Spring é um *framework* de Java de código livre desenvolvido por Johnson (2002) que disponibiliza diversos módulos de software com o objetivo de facilitar o desenvolvimento de aplicações. Neste trabalho utilizamos o módulo Spring Boot, que permite iniciar rapidamente projetos Spring, gerencia as dependências e facilitando o processo de publicação, além de possuir ferramentas que facilitam a criação e configuração do servidor.

Hibernate: Hibernate é um *framework* Java de código livre que realiza o mapeamento objeto-relacional, facilitando o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto. O Hibernate tem com objetivo reduzir a complexidade dos programas Java, transformando classes em tabelas e vice-versa. Ele gera as chamadas SQL e libera o desenvolvedor do trabalho manual, mantendo a aplicação portável para quaisquer bancos de dados SQL Pivotal (2019).

Dart: Dart é uma linguagem de *script* desenvolvida pela Google (2011) e criada para ser uma linguagem otimizada para interfaces e ter alto desempenho em todas as plataformas. Outra característica marcante do Dart é que ele pode ser executada diretamente na máquina virtual ou ser compilada para código JavaScript. De acordo com o github, Dart foi a linguagem de programação com maior crescimento entre os anos de 2018 e 2019.

Flutter: Flutter é um SDK (Kit de Desenvolvimento de Software) de código aberto criado pela Google (2019) para desenvolvimento de aplicativos Andoid, iOS, Desktop ou Web com uma única base de código. Flutter foi desenvolvido visando ter um desenvolvimento rápido, ter uma interface expressiva e flexível e ter uma alta performance, compilando o código Flutter utilizando os compiladores nativos do Dart, dando ao Flutter a performance nativa do iOS e Android.

Android Studio: Android Studio é a ferramenta oficial de desenvolvimento para aplicativos Andoid. Ele foi desenvolvido pela Google (2013) e possui diversos recursos para facilitar a criação de aplicativos Android, entre eles emuladores para executar a aplicação em uma celular virtual, sugestões inteligentes de código, estatísticas em tempo real de diversos dados, entre outros.

MySql: MySql é um sistema de gerenciamento de bando de dados (SGBD) que utiliza a linguagem SQL como interface, ele possui diversos recursos, e neste protótipo foi utilizado o MySql Workbench para desenvolver, administrar e modelar o banco de dados Oracle (1994).

Capítulo 4

Desenvolvimento

Neste capítulo será apresentado como a aplicação foi desenvolvida e será explanado sobre a arquitetura do sistema, a modelagem do banco de dados, as tecnologias utilizadas e o funcionamento geral do sistema.

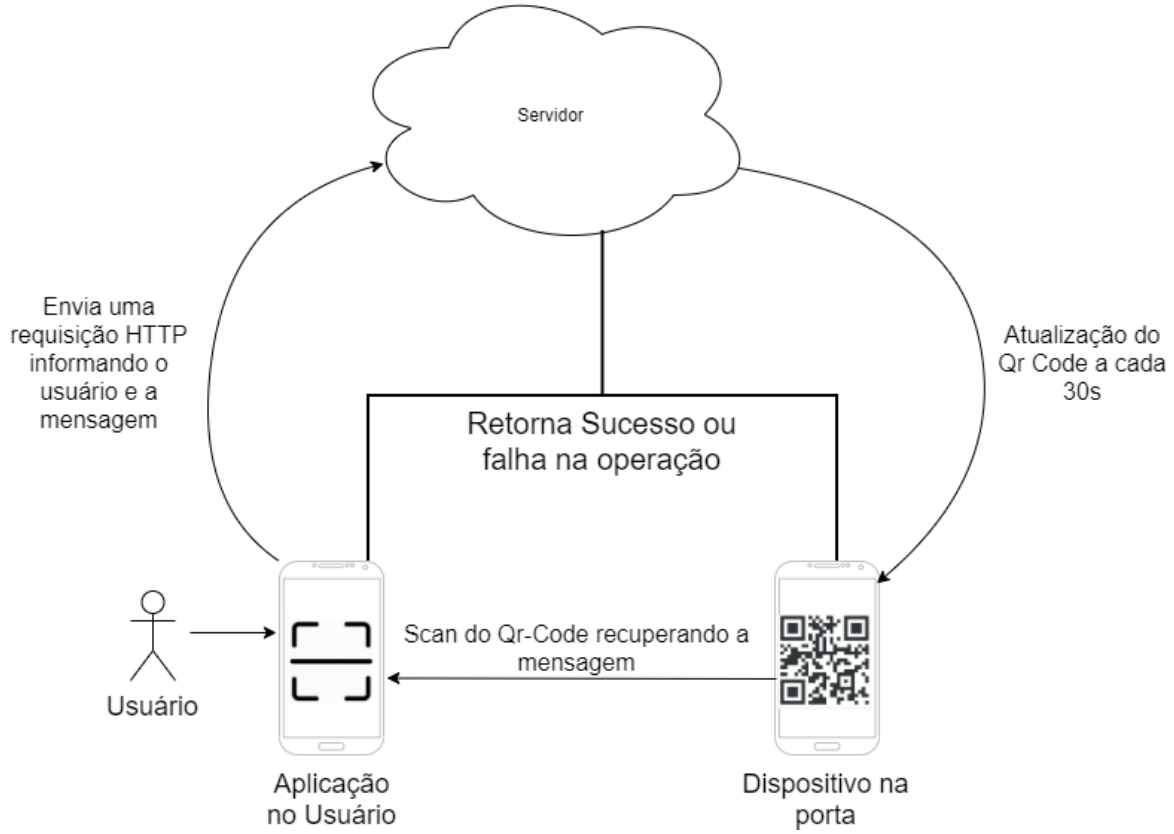


Figura 4.1: Funcionamento do sistema

O sistema que foi desenvolvido tem como um dos objetivos aumentar a segurança em locais com muitos pontos de acesso. Enumeramos alguns exemplos de locais passiveis de utilizar

este sistema: prédios públicos, universidades, apartamentos, residencias privadas, locais de trabalho, entre outros. Para isto foi criado um aplicativo celular que deverá ser instalado no aparelho do usuário final. Com este aplicativo instalado, o usuário poderá ler um QR-Code dinamicamente gerado em um *Raspberry pi* na porta. No caso desta versão inicial, o QR-Code será gerado em outro celular para demonstrar o funcionamento do sistema. Caso o usuário se autentique corretamente e tenha permissão de acesso, a porta se abrirá automaticamente. Maiores detalhes desta operação podem ser observados na figura 4.1.

O mecanismo de abrir a porta fisicamente não será abordado neste trabalho e focaremos no *design* e arquitetura do sistema. Outro ponto importante a ser citado é que o QR-Code é apenas uma maneira inicial de identificação pois a arquitetura foi preparada para futuramente utilizar outras tecnologias de autenticação sem grandes alterações, como exemplo de outros sistemas de autenticação temos o NFC (*Near Field Communication*), que funciona por aproximação do celular, ou até mesmo biometria da impressão digital, liberando o acesso diretamente com a digital na porta e contendo o aplicativo celular apenas como registro, controle de acessos e comunicação.

4.1 Arquitetura

Como citado anteriormente, a arquitetura é o coração de todo software e dificilmente encontraremos um software seguro, robusto e escalável com uma arquitetura mal projetada. As decisões arquiteturais são extremamente importantes para que o software seja bem construído.

A arquitetura desenvolvida foi inspirada nos modelos arquiteturais explicados no referencial teórico. O objetivo principal é criar uma arquitetura que produza um software que atenda os seguintes requisitos:

Performance: O sistema deve ter uma resposta rápida (baixa latência), para que os usuários não precisem esperar para ter o retorno apos ler o Qr-Code;

Disponibilidade: O sistema deve estar sempre disponível para atender requisições do usuário a qualquer momento;

Segurança: O sistema deve ser seguro, permitindo o acesso apenas de usuários autorizados para aquela entrada, negando o acesso de todos os outros;

Escalabilidade: O sistema deve estar preparado para receber milhões de requisições a todo momento, podendo aceitar mais usuários conforme a necessidade;

Tolerância a Falhas: Caso ocorra alguma falha interna o sistema deve cuidar dela para que o usuário não tenha conhecimento desta falha e receba a resposta desejada;

Fácil Manutenção: O sistema deve ser construído de maneira que seja fácil dar manutenção, reduzindo o tempo de implementação de novas funcionalidades e manutenção.

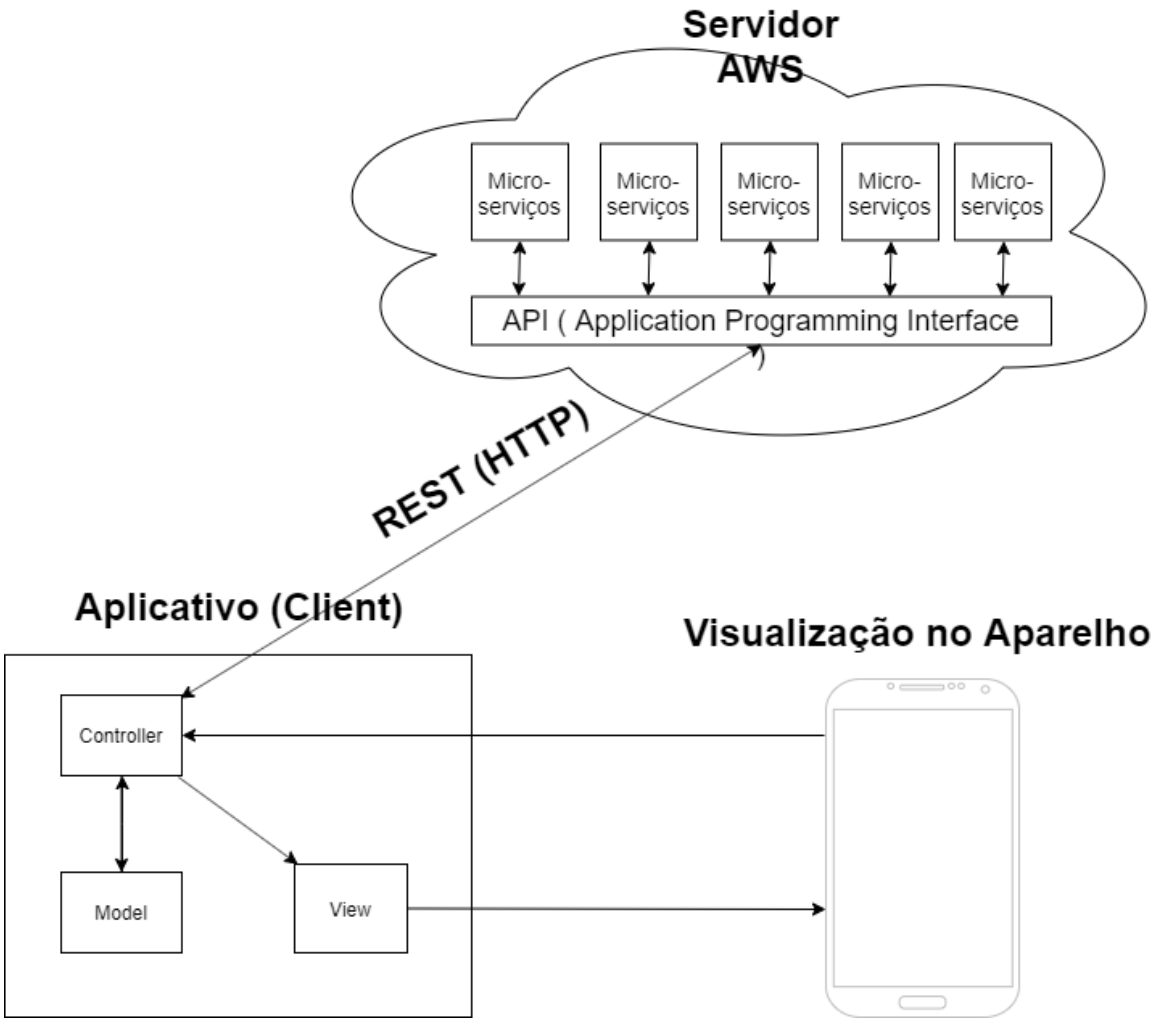


Figura 4.2: Arquitetura Geral do sistema

Para atender estes requisitos foi projetada uma arquitetura dividida entre cliente e servidor. No cliente (celular) é instalado um aplicativo desenvolvido organizado na arquitetura MVC e no servidor foi implementado na AWS expondo micro serviços como mostra na figura 4.2.

4.1.1 Cliente

Para o aplicativo cliente foi escolhida a arquitetura MVC que traz diversas vantagens, entre elas: melhor reaproveitamento de código, facilidade de manutenção e adição de recursos, maior integração da equipe de desenvolvimento e divisão de tarefas e facilidade em manter o código limpo e organizado.

A arquitetura MVC divide o código em camadas bem definidas, tornando o código mais legível e acelerando o desenvolvimento e manutenção, possibilitando um melhor controle sobre a aplicação. Como apresentado na figura 4.2, o Controller da aplicação cliente fica responsável por enviar e receber requisições do servidor e enviar os dados para a View. Esse irá apresentar os dados para o usuário, comunicar com a camada de Model. Model trabalhará em cima dos dados e atenderá as demandas de dados das requisições do usuário.

O aplicativo *mobile* foi desenvolvido em Flutter, utilizando a linguagem Dart e a IDE Android Studio. Foram criadas apenas as telas básicas para a demonstração do funcionamento do sistema, como apresentado na figura 4.3. As telas consistem na tela de *login*, a tela para escanear do Qr-Code e a tela que apresenta o Qr-Code atualizada a cada minuto.

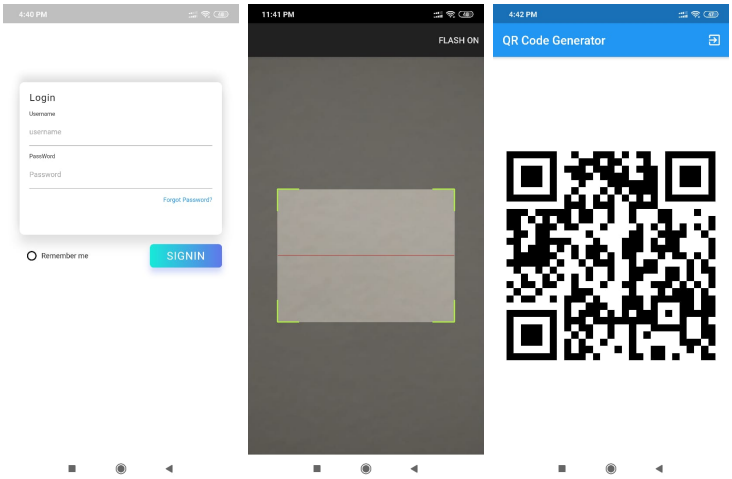


Figura 4.3: Exemplo de telas do sistema

4.1.2 Servidor

Para colocar o servidor na nuvem foi utilizado a AWS. A AWS fornece vários recursos para gerenciar a infra-estrutura do servidor. Dentre esses, foi escolhido o Elastic Beanstalk, que é um serviço de fácil implementação e voltado para performance e segurança das aplicações Amazon (1994). A arquitetura do Elastic Beanstalk consiste em:

VPN (*Virtual Private Network*): A VPN é uma nuvem pessoal privada que permite estabelecer conexões seguras entre a nuvem e os dispositivos vos de forma segura, aceitando acesso apenas de dispositivos autorizados;

***Security Group*:** Consiste em grupos de segurança que podem ser cadastrados na AWS, permitindo aceitar ou rejeitar acessos à recursos específico;

***Elastic Load Balancing*:** É um recurso responsável por distribuir o trafego de entrada para uma instância ativa. Se necessário e corretamente configurado, será criado novas

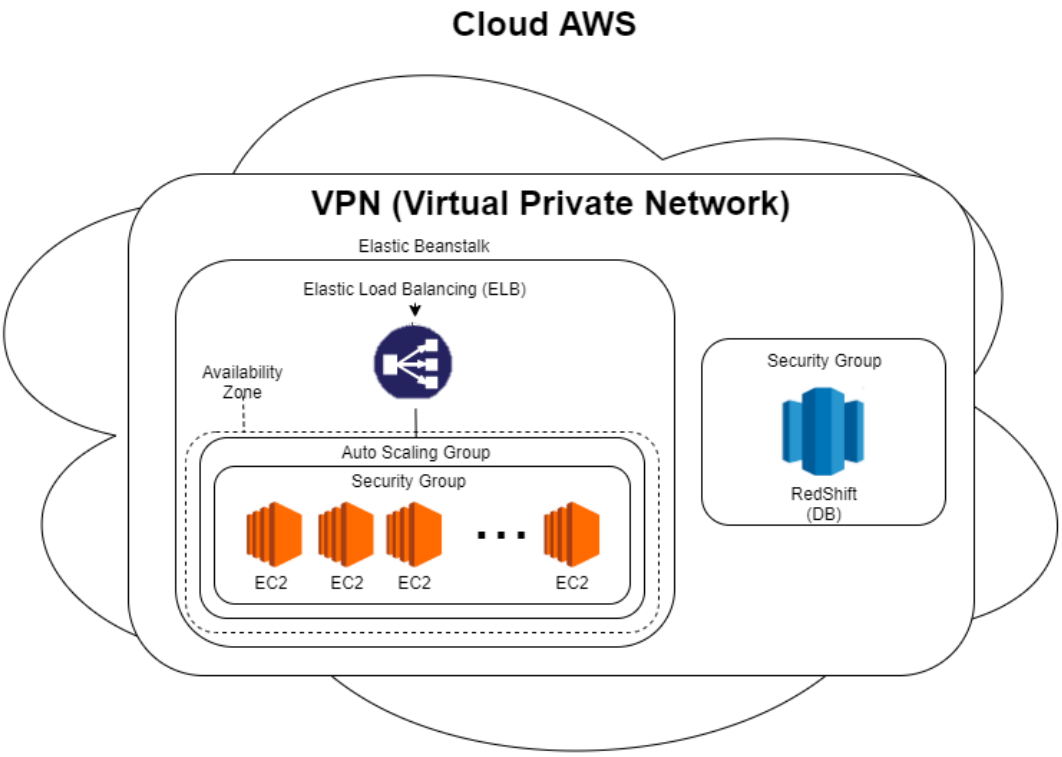


Figura 4.4: Arquitetura do Elastic Beanstalk na AWS
Amazon (1994)

instâncias automaticamente. Este serviço é recomendado para balanceamento de carga de dados, sendo capaz de lidar com milhões de solicitações por segundo, mantendo latências minímas.

***Availability Zone*:** Consiste em zonas de segurança à falhas. Estas zonas são completamente isoladas uma das outras. As instâncias são distribuídas em várias zonas de disponibilidade fazendo que, caso uma zona apresente problema (falha) outra assuma seu processamento, garantindo que o sistema fique operacional a todo momento.

***Auto Scaling Group*:** Grupo de crescimento horizontal de instâncias. Neste grupo é configurado o crescimento e redução automática de instâncias de acordo com a quantidade de requisições recebidas.

***EC2*:** Instância que estará executando o servidor lançado na AWS.

***RedShift*:** Recurso para criação de banco de dados Mysql e outros dentro da Cloud AWS.

Sendo executado nas instâncias EC2, temos uma aplicação desenvolvida em Java e utilizando os *frameworks Spring* e *Hibernate* para auxiliar no desenvolvimento e utilizando tam-

bém uma extensão da IDE eclipse denominada *Spring Tool Suite*. Este aplicativo foi desenvolvido de forma estruturada e organizada, tornando o desenvolvimento de novas funcionalidades e manutenção eficientes, com testes unitários de todos os métodos.

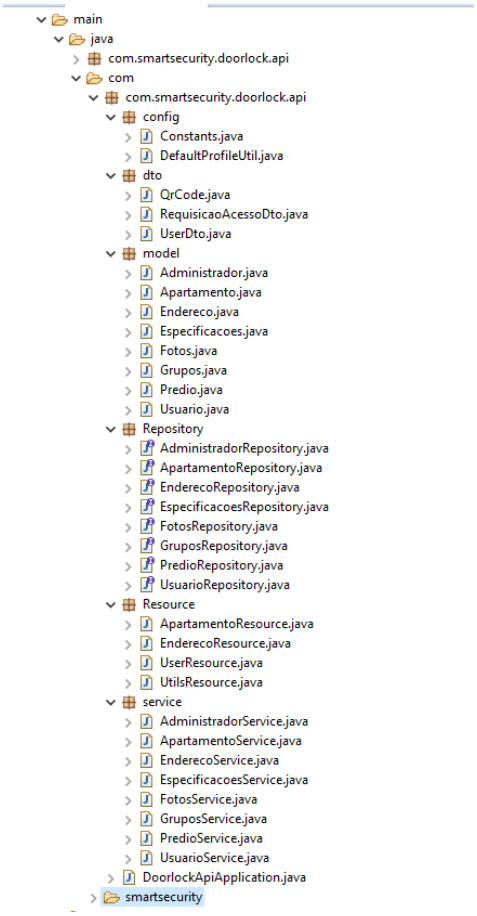


Figura 4.5: Organização do desenvolvimento do Servidor

4.2 Banco de dados

O banco de dados foi modelado no MySql Workbench como mostrado na figura 4.6, ele foi projetado de forma que atenda os re quesitos do sistema e tenha segurança e velocidade em suas pesquisas, guardando registros de todas as operações.

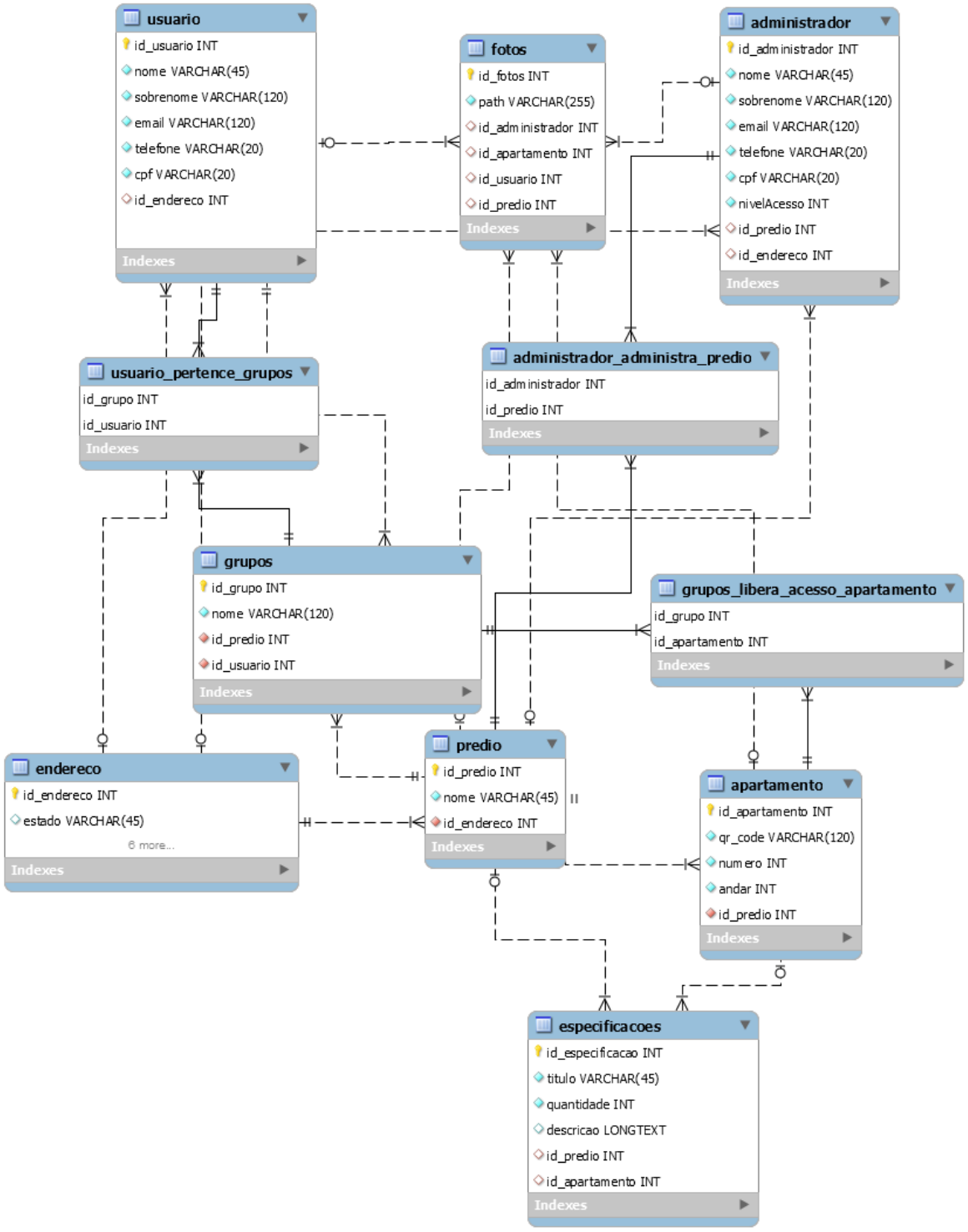


Figura 4.6: Banco de Dados

Capítulo 5

Resultados

5.1 Pesquisa de mercado

Para validar o potencial da aplicação foi realizada uma pesquisa de mercado na Internet com 43 entrevistados. A pesquisa em questão se encontra nos anexos e possui 14 perguntas, tentando descobrir principalmente o interesse e confiança da população em um sistema de controle de acessos.

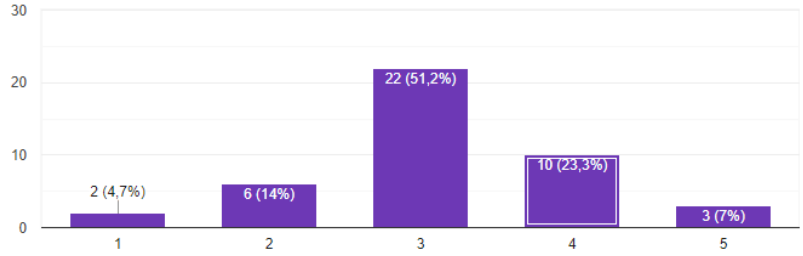


Figura 5.1: Analise de Confiança do sistema

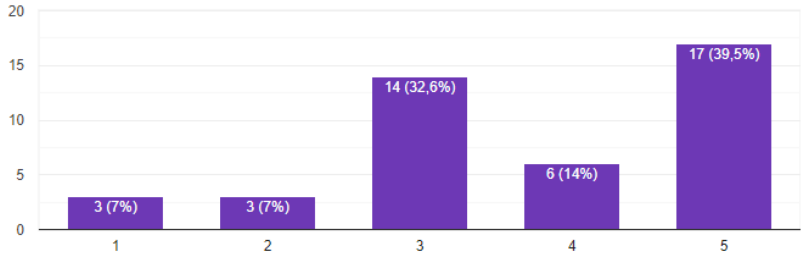


Figura 5.2: Interesse dos entrevistados pelo sistema

Analisando os resultados da pesquisa podemos perceber que existe um interesse grande por produtos deste tipo no mercado, apesar de existir outros sistemas similares eles não são muito conhecidos, abrindo oportunidade para novos sistemas.

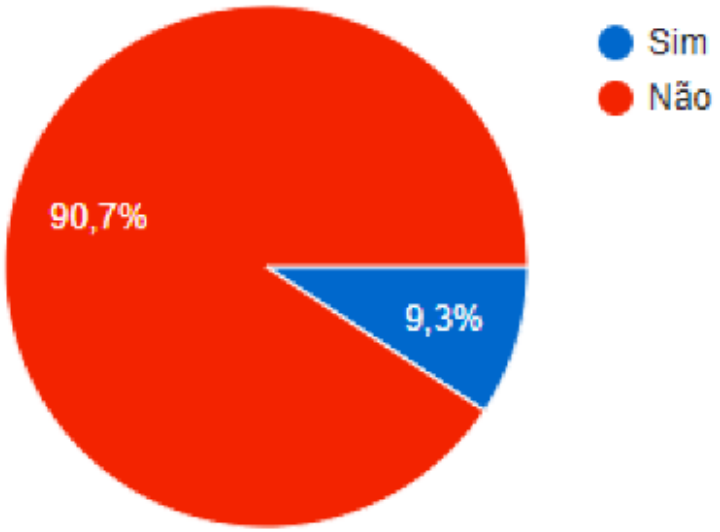


Figura 5.3: Conhecimento dos entrevistados por outros sistemas similares

5.2 Produto Mínimo Viável (MVP)

Para validar e demonstrar toda a arquitetura proposta foi desenvolvido o aplicativo "DoorLock"como um produto mínimo viável. Ele foi desenvolvido através do Android Studio e utilizando o SDK flutter, gerando um aplicativo multi-plataformas com um único código base e demonstrando os recursos mais essenciais da arquitetura planejada.

Neste aplicativo conseguimos simular vários aparelhos logando na aplicação, dependendo do tipo de usuário logado o aplicativo apresenta a tela do QR-Code que controla aquela porta especifica ou a opção de escanear Qr-Codes, se o usuário tiver acesso a uma porta da qual ele escaneou irá retornar uma mensagem de sucesso e o acesso será liberado.

Capítulo 6

Conclusão

Este trabalho propôs a criação de um sistema para autenticação de acessos, aumentando a segurança e centralizando mais funcionalidades no aparelho celular que esta cada vez mais presente no dia-a-dia. Mesmo em caso de perda do aparelho pode-se facilmente gerenciar os acessos e bloquear aquele celular especifico, tornando mais difícil o acesos não autorizado por terceiros.

O sistema apresentado difere de sistemas de acesso tradicionais pois é baseado em uma arquitetura "na nuvem", escalável, gerenciado e modular.

Ainda existem muitas melhorias para serem realizadas no sistema, porém pode-se dizer que o protótipo inicial foi desenvolvido com sucesso. O servidor já esta funcional e disponível na nuvem, utilizando uma plataforma segura e com controle rígido de acessos. A principal funcionalidade de liberar acesso a pessoas que possuem autorização foi devidamente implementada. A arquitetura foi projetada para atender os requisitos propostos.

Capítulo 7

Trabalhos Futuros

Ainda existem muitos pontos para aprimorar o presente trabalho, expandindo sua areá de pesquisa, podendo acrescentar os seguintes topicos:

Blockchain: Blockchain é uma base de dados distribuída que armazena registro de transações permanentemente e é a prova de violações. Com o uso de *Block Chain*, seria reduzido o risco de falha de segurança, tornando o sistema mais resistente a ataques maliciosos.

Requisições por HTTPs: HTTPS é uma interpretação do protocolo HTTP com uma camada adicional de segurança, fazendo com que os dados da requisição sejam transportados por meio de uma conexão criptografada, certificando a autenticidade do tanto do cliente quanto do servidor. Normalmente este protocolo é utilizado quando temos uma comunicação entre o cliente e o servidor sem acesso por terceiros, acrescentando ainda mais ao nível de segurança da aplicação.

Biometria Digital: O sistema foi projetado para aceitar vários mecanismo de autenticação como mecanismo de abrir a porta. Um destes mecanismos seria a autenticação por biometria da digital.

Aprimoramento do aplicativo: O aplicativo desenvolvido esta implementando com o mínimo possível para testar o sistema, ainda existem funcionalidades a serem desenvolvidas.

Deploy em produção: O aplicativo desenvolvido esta inacessível para terceiros, deve-se analisar a trilha de *deploy* em produção da aplicação.

Desenvolvimento do mecanismo para abrir a porta: Implementação do mecanismo para abrir a porta fisicamente.

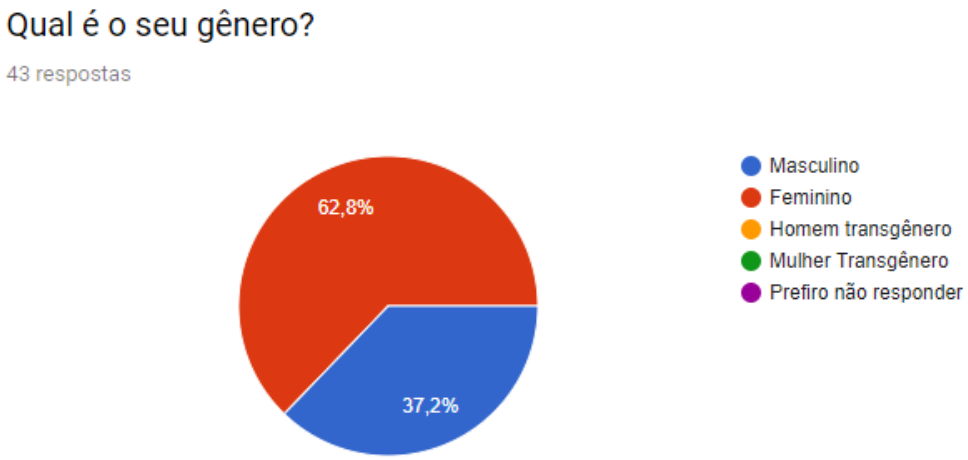
Utilização do *Raspberry Pi*: Na solução final deste trabalho pretende-se utilizar o *Raspberry Pi* como o dispositivo que apresentará o QR-Code para o usuário final na porta,

porem devido a falta de tempo e recursos não foi possível utilizar o aparelho, contudo a linguagem de programação escolhida foi criada para o desenvolvimento de aplicações multi-plataformas com o mesmo código base, sendo possível simular o funcionamento do *Raspverry Pi* em um aparelho celular.

Apêndice A

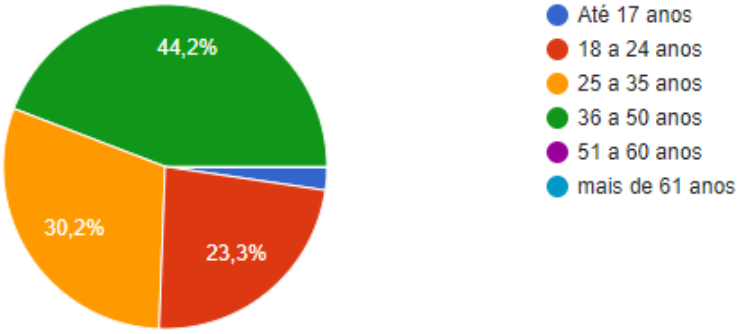
Apêndice

A.1 Resultados pesquisa de mercado



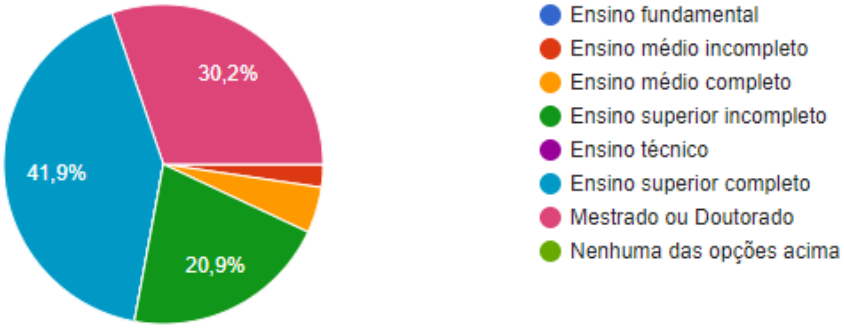
Qual sua faixa etária

43 respostas



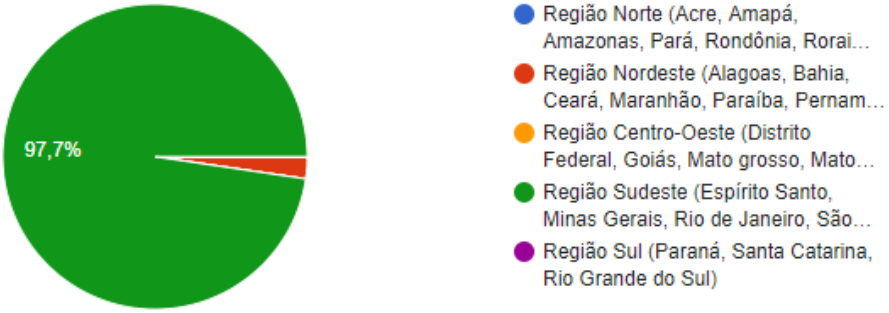
Qual seu nível de escolaridade?

43 respostas



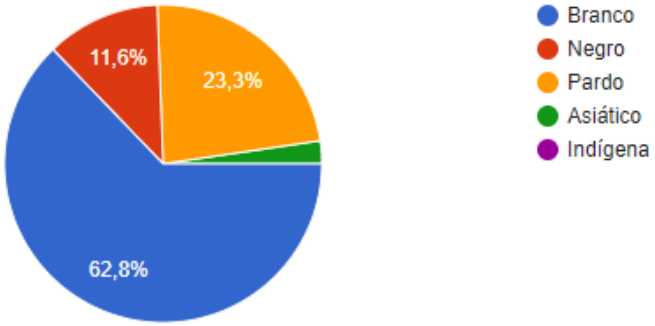
Em que região do Brasil você mora?

43 respostas



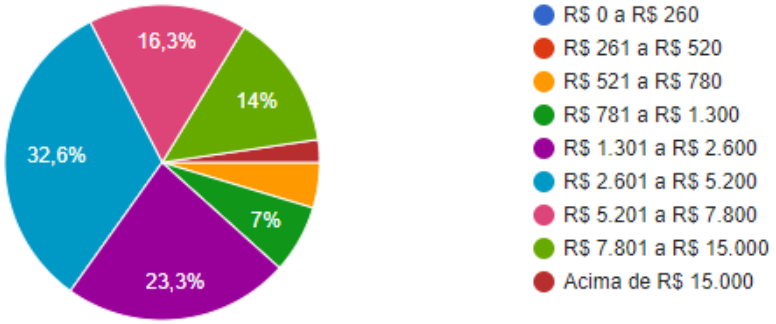
Qual raça/etnia melhor descreve a sua descendência?

43 respostas



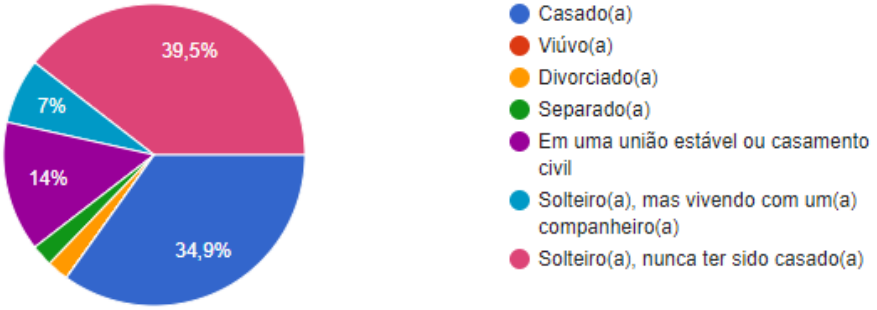
Aproximadamente, qual é a sua renda familiar mensal por dependente?

43 respostas



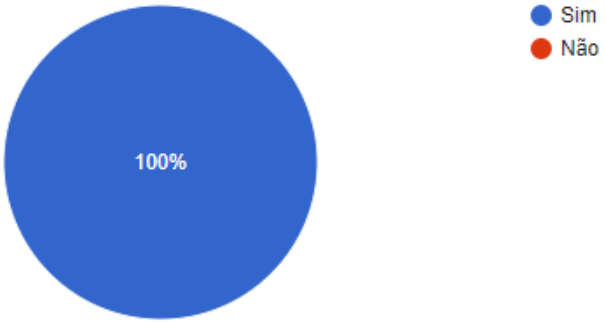
Qual das opções abaixo melhor descreve seu estado civil atual?

43 respostas



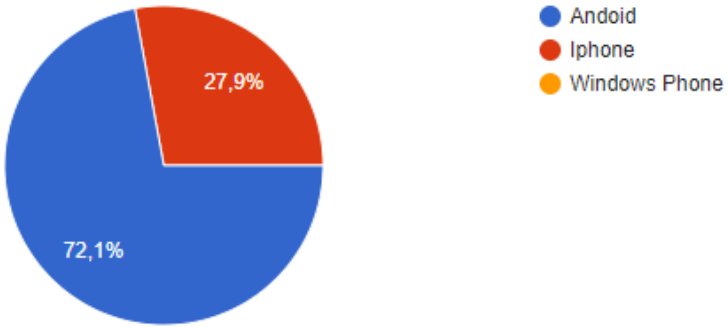
Você possui um aparelho celular?

43 respostas



Caso afirmativo, qual o sistema operacional do seu celular

43 respostas



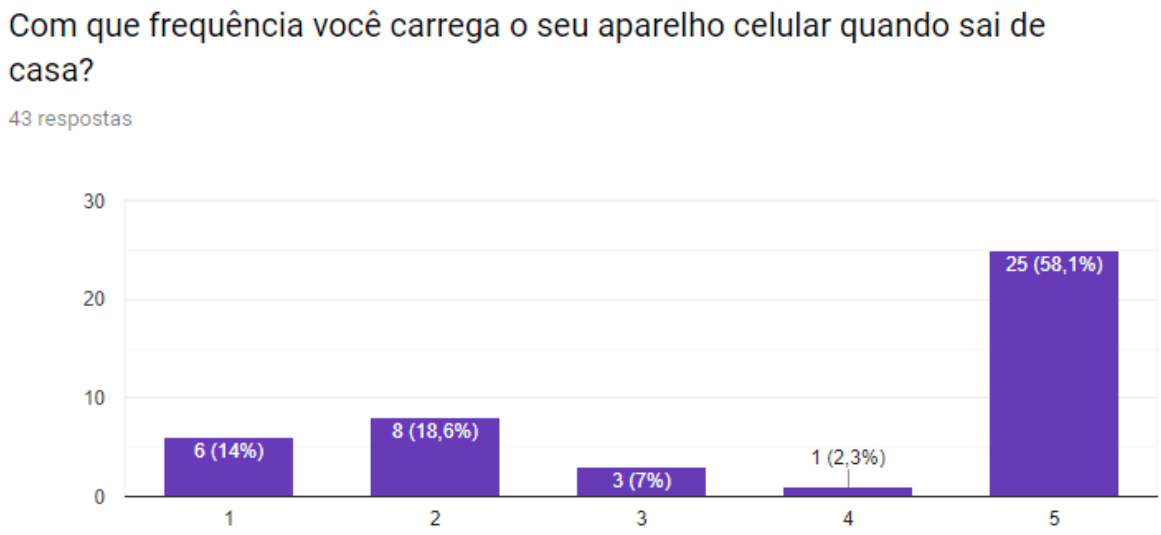


Figura A.1: Frequência uso do celular : 1. menor frequência; 5 maior frequência

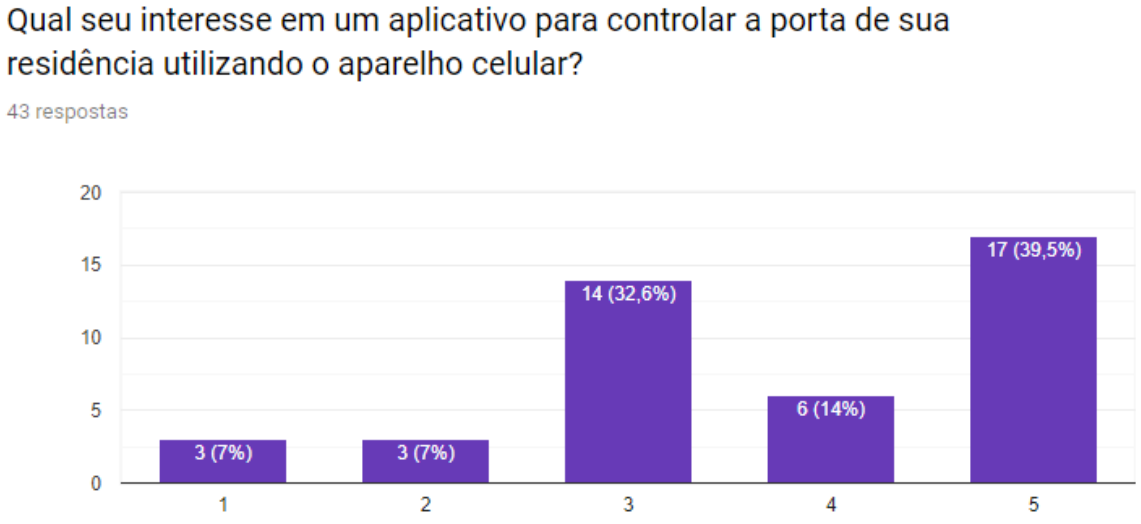


Figura A.2: Interesse pelo sistema : 1. menor interesse; 5 maior interesse

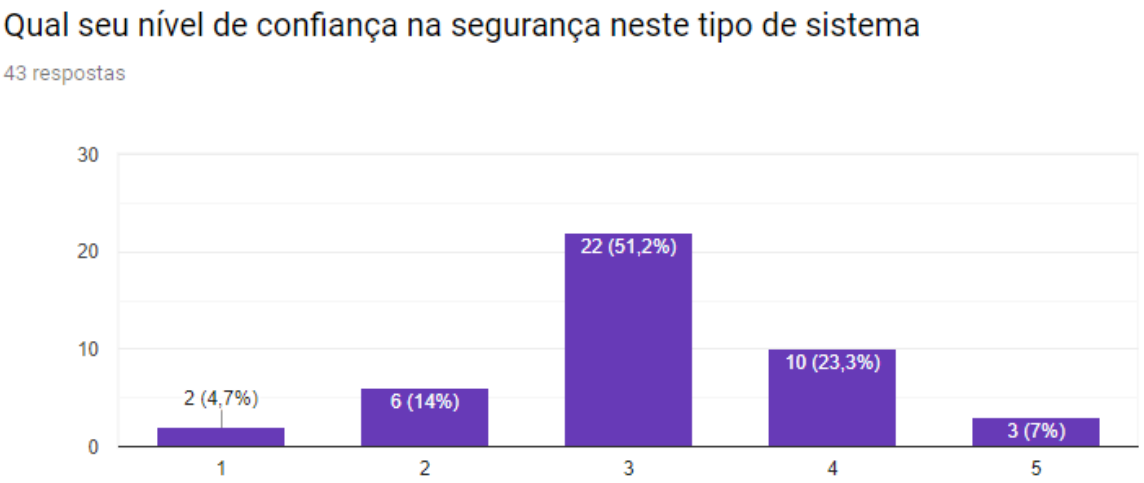
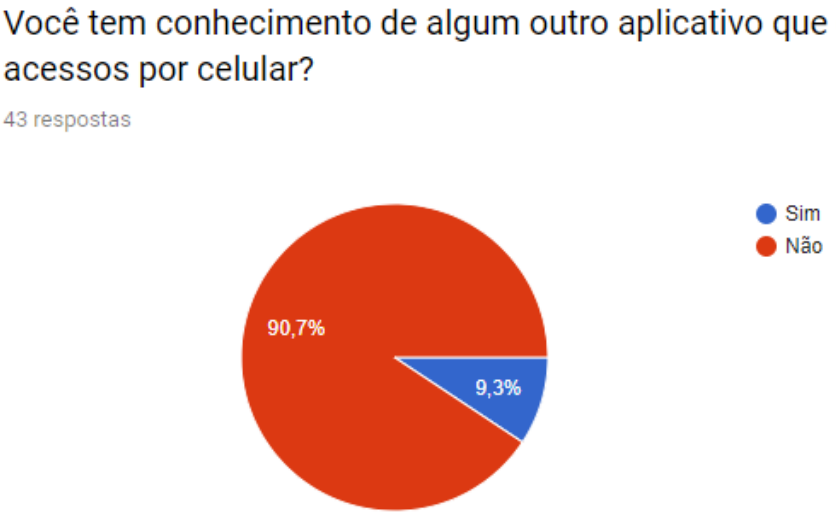


Figura A.3: Confiança pelo sistema : 1. menor confiança; 5 maior confiança



Referências Bibliográficas

aaaa.

aaaa.

aaaa.

Amazon (1994). Amazon. <https://aws.amazon.com/about-aws/>. Accessed: 2019-11-26.

Anatel (2015). Mvc. <https://www.anatel.gov.br/dados/component/content/article/84-destaque/270-destaques-smp>. Accessed: 2019-11-07.

Anjomshoaa, A. e Tjoa, A. M. (2011). How the cloud computing paradigm could shape the future of enterprise information processing. pp. 7–10.

Apigy (2018). Lockitron. <https://lockitron.com/>. Accessed: 2018-11-01.

August (2018). August smart lock - how it works. <https://august.com/pages/how-it-works>. Accessed: 2018-11-01.

Bahrin, A. K.; Mohd; Othman; Fauzi; Azli, H. N.; Nor; Talib, F. e Muhamad (2016). Industry 4.0: A review on industrial automation and robotic. *Jurnal Teknologi*, 78.

Chandramouli, R. (2019). Security strategies for microservices-based application systems. *National Institute of Standards and Technology*.

Click, R. (2015). Smartphones começam a substituir chaves de casa.

Foundation, R. P. (2018). Raspberry pi. <https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/>. Accessed: 2018-11-01.

Freeman, A. (2013). *Pro ASP.NET MVC 5*. Apress, 1 edição.

Glover, B. e Bhatt, H. (2007). *FUNDAMENTOS DE RFID*. Alta Books, 1 edição.

Google (2011). dart. <https://dart.dev/>. Accessed: 2019-11-12.

Google (2013). Androidstudio. <https://developer.android.com/studio/>. Accessed: 2019-11-12.

Google (2019). flutter. <https://flutter.dev/>. Accessed: 2019-11-12.

Gorton, I. (2011). *Essential Software Architecture*. 3, 2 edição.

Johnson, R. (2002). Spring. <https://spring.io/>. Accessed: 2019-11-12.

Lucio Camilo Oliva Pereira, M. L. d. S. (2009). *Android para desenvolvedores*. Brasport, 1 edição.

Mota, R. P. B. (2006). Extensões ao protocolo de comunicações epcgloboal para tags classe 1 utilizando autenticação com criptografia de baixo custo para segurança em identificação por radiofrequencia. *Dissertação (Mestrado em Ciências Exatas e da Terra) - Universidade Federal de São Carlos*.

Nenad Medvidovic, R. N. T. (2010). Software architecture: Foundations, theory, and practice.

Oracle (1994). Mysql. <https://www.mysql.com/>. Accessed: 2019-11-12.

Oracle (2019). Java. <https://www.java.com/pt_BR/about/>. Accessed: 2019-11-12.

Peter Mell, T. G. (2011). The nist definition of cloud computing. <http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf>. Accessed: 2019-11-07.

Pivotal (2019). Hibernate. <https://hibernate.org/>. Accessed: 2019-11-12.

Ramos, A. (2015). O que é mvc? <https://tableless.com.br/mvc-afinal-e-o-que/>. Accessed: 2019-11-07.

Soon, T. J. (2008). Qr code. *synthesis journal*, pp. 59–78.

Wortmann, F. e Flüchter, K. (2015). Internet of things. *Business & Information Systems Engineering*, 57(3):221–224.