PM

Desenvolvimento de um Sistema IoT com Diferentes Formas de Interação

PROJETO DE MESTRADO

Filipe Alexandre Nóbrega Moura
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

abril | 2021

Desenvolvimento de um Sistema IoT com Diferentes Formas de Interação

PROJETO DE MESTRADO

Filipe Alexandre Nóbrega Moura
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTAÇÃO
Karolina Baras

UNIVERSIDADE da MADEIRA

Desenvolvimento de um Sistema IoT com Diferentes Formas de Interação

Filipe Alexandre Nóbrega Moura

Constituição do júri de provas públicas:

Eduardo Miguel Dias Marques, (Professor Auxiliar da Universidade da Madeira), Presidente

Filipe Magno de Gouveia Quintal, (Professor Auxiliar da Universidade da Madeira), Vogal

Karolina Baras, (Professora Auxiliar da Universidade da Madeira), Vogal

Abril 2021

Funchal – Portugal

Resumo

A Internet das Coisas tem vindo a crescer de forma galopante, pois o uso da tecnologia da informação e comunicação (TIC) pode tornar tudo mais inteligente. A IoT inclui todos os dispositivos e objetos que podem ser conectados permanentemente à Internet, comunicar entre si e trocar dados.

Este projeto tem como objetivo principal o estudo de diferentes formas de interação de um utilizador com um sistema IoT, para averiguar qual a forma mais eficaz e prática de interação. Foi desenvolvido um sistema IoT com suporte a diferentes formas de interação, tais como, uma aplicação web, uma aplicação móvel e um assistente virtual. Este sistema possui microcontroladores que incluem sensores e atuadores, dispersos num edifício ou casa. Um sistema desta natureza gera uma enorme quantidade de dados. Por esse motivo, os dados foram armazenados numa base de dados na *cloud*.

Para a criação do sistema, foram estipulados os requisitos funcionais e não funcionais e desenhada a arquitetura do sistema. A arquitetura é constituída por um módulo microcontrolador, um servidor central, por brokers MQTT, por uma base de dados remota, por sensores e atuadores, por um website, aplicação móvel e por uma skill na Alexa. Para os microcontroladores, foram usados NodeMCUs, que são equipamentos de baixo custo, com suporte a ligações WiFi. No que diz respeito ao servidor central, que consiste numa API REST, fica hospedado num Raspberry.

Foram realizados testes de usabilidade a cada uma das aplicações desenvolvidas. Para estes testes, foi usado o questionário SUS *(System Usability Scale)*. Por fim, foram realizadas instalações em três casas para que os utilizadores pudessem interagir por mais tempo com o sistema. O objetivo destas instalações foi poder inferir qual a melhor forma do utilizador final interagir com um sistema desta natureza a fim de poder retirar conclusões sobre as diferentes formas de interagir com o sistema e assim verificar qual das aplicações desenvolvidas mais se adequa a um sistema IoT.

Palavras-chave:

IoT, Alexa, Microcontrolador, Interação, Computação na núvem

Abstract

The Internet of Things has been growing at a fast rate, as the use of Information and Communication Technology (ICT) can make everything smarter. IoT includes all the devices and objects that can be permanently connected to the Internet, communicate with each other and exchange data.

The main goal of this project is to study different ways of interaction of a user with an IoT system, to find out which is the most effective and practical way of interaction. An IoT system was developed that supports different forms of interaction, such as a web application, a mobile application and a virtual assistant. This system has microcontrollers that include sensors and actuators, dispersed in a building or house. A system of this nature generates a huge amount of data. For this reason, the data was stored in a cloud database

For the creation of the system, the functional and non-functional requirements were stipulated, and the architecture of the system was designed. The architecture consists of a microcontroller module, a central server, MQTT brokers, a remote database, sensors and actuators, a website, a mobile application, and a skill at Alexa. For the microcontrollers, NodeMCUs were used, which are a low-cost equipment, with WiFi connection support. For the central server, which consists of a REST API, is hosted in a Raspberry.

Usability tests were performed on each of the applications developed. For these tests, the SUS (System Usability Scale) questionnaire was used. Finally, installations were made in three houses so that users could interact with the system for a longer period of time. The objective of these installations was to be able to infer which is the best way for the end user to interact with a system of this nature in order to be able to draw conclusions about the different ways of interacting with the system and thus verify which of the applications developed is most suitable for an IoT system.

Keywords:

IoT, Alexa, Microcontroller, Interaction, Cloud computing

Agradecimentos

Esta secção é dedicada a todas as pessoas que contribuíram e ajudaram ao desenvolvimento deste projeto. Gostaria em primeiro lugar de agradecer à minha família por todo o apoio e acompanhamento durante toda esta jornada académica.

Gostaria também de agradecer à minha orientadora, Karolina Baras, pela disponibilidade para fornecer orientações, dicas e apoio durante todo o desenvolvimento do projeto.

Por fim, agradeço a todas as pessoas que estiveram envolvidas no projeto e que contribuíram para a realização dos testes e entrevistas.

Índice

Índice de figuras

Índice de tabelas

Lista de acrónimos

ADC - Analog-to-digital converter

AP – Access Point

API – Aplication Programming Interface

AR - Augmented reality

ARN - Amazon Resource Names

ASK – Alexa Skill Kit

AVI – Assistente Virtual Inteligente

AVS – Alexa Voice Service

AWS – Amazon Web Services

BLE - Bluetooth Low Energy

CoAP - Constrained Application Protocol

CSRF - Cross-site request forgery

DBaaS - Database-as-a-Service

DDoS - Distributed Denial of Service

GPIO - General Purpose Input/Output

GUI - Graphical User Interface

HTTP - Hypertext Transfer Protocol

HTTPS - Hyper Text Transfer Protocol Secure

IDE - Integrated Development Environment

IoT – Internet of Things

IP – Internet protocol

JSON - JavaScript Object Notation

LDR – Light Dependent Resistor

LoRaWan - Long Range Wide Area Network

M2M – Machine to Machine

MQTT - MQ Telemetry Transport

NTP- Network Time Protocol

ODM - Object Document Mapping

OTA- Over-the-Air

QoS – Quality of Service

REST - Representational State Transfer

SIoT- Social Internet of Things

SO - Smart Objects

SSL– Secure Sockets Layer

SUS - System usability scale

TCP – Transmission Control Protocol

TIC - Tecnologias da informação e comunicação

TLS - Transport Layer Security

UDP - User Datagram Protocol

UI – User Interface

URI - Uniform Resource Identifier

USB - Universal Serial Bus

UWB - Ultra Wide Band

VUI - Voice User Interface

WSGI - Web Server Gateway Interface

1. Introdução

A Internet das Coisas, compreende todos os aparelhos e objetos que se encontram habilitados a estar permanentemente ligados à Internet, tendo a capacidade de se identificar na rede, de comunicar e trocar dados entre si. Estes objetos são capazes de recolher uma grande quantidade de dados sobre o espaço que os rodeia, pelo que, o seu estado pode ser alterado pelo meio envolvente ou por intervenção humana [1].

Há uma aceleração crescente no que diz respeito à Internet das Coisas (IoT), uma vez que tudo se torna mais inteligente com o uso da tecnologia da informação e comunicação (TIC) [2]. No mundo de hoje, a Internet é um meio de comunicação popular. Muitos dispositivos têm agora *WiFi* e podem conectar-se a *Smartphones* ou a computadores domésticos.

Em certas cidades já é possível visualizar algumas aplicações práticas da internet das coisas, como por exemplo, numa melhor organização do trânsito, na agilização de tratamentos na área da saúde, numa prática mais eficiente de agricultura e na preservação do meio ambiente [1][3]. No entanto, na maior parte destes casos, a análise dos dados está bastante condicionada à capacidade humana de analisar os dados que os dispositivos geram, já que muitas das vezes é necessário o factor humano de forma a poder extrair conclusões a partir dos dados apresentados [3].

De acordo com a Gartner, uma empresa de consultoria, o número de dispositivos conectados à Internet irá exceder os 25 mil milhões em 2020, o que é um crescimento considerável relativamente às previsões de 2015, que rondavam os 4.8 bilhões de dispositivos. De acordo com a mesma fonte, a Internet das Coisas será uma tecnologia cada vez mais presente nas nossas vidas, sendo esperados resultados positivos na receção por parte dos utilizadores [3].

Esta tecnologia fornece inúmeras vantagens, quer para uso doméstico, quer para uso industrial. A IoT permite aumentar o conforto de um lar. É possível, por exemplo, controlar remotamente toda a casa a partir de qualquer lugar [4]. Como estes dispositivos interagem e comunicam uns com os outros e realizam muitas tarefas por si só, então minimizam o esforço humano e ao reduzir o esforço irão poupar tempo ao utilizador. No contexto empresarial, a IoT vai permitir identificar problemas, melhorar a utilização de recursos, aperfeiçoar infraestruturas e reduzir custos [5]. Um dos grandes benefícios da tecnologia IoT está na tomada de decisões, devido ao facto dos empresários poderem ter acesso a dados em tempo real, o que facilitará as tomadas de decisão corretas [6].

Apesar dos seus benefícios, a Internet das coisas também apresenta desafios a resolver. Um dos grandes problemas neste tipo de sistemas é a segurança. A privacidade dos dados dos utilizadores exige que sejam implementadas soluções sólidas, de forma a garantir a segurança dos dados. Também há um problema de interoperabilidade entre dispositivos, pelo facto de muitos deles, não usarem os mesmos protocolos. Outro problema, que será o foco principal e o objetivo deste projeto, é a interação, visto que, aplicações de um sistema desta natureza ainda estão relativamente pouco exploradas e não há unanimidade sobre qual a melhor forma de interação.

Um sistema IoT pode ser aplicado, por exemplo, na construção de uma casa inteligente. Numa casa inteligente é possível, a um utilizador, controlar aparelhos tais como a iluminação e termostatos, que estão conectados a uma rede através de um protocolo de comunicação, sendo o WiFi e o Bluetooth os mais comuns [7]. Muitas casas inteligentes permitem inúmeras possibilidades de integração com sistemas e aparelhos disponíveis no mercado. Embora o mercado emergente de casas inteligentes ainda tenha muito espaço para crescimento, exemplos de tecnologia de casas inteligentes, atualmente no mercado, incluem termostatos inteligentes, luzes inteligentes com sensores de presença e sistemas de segurança inteligentes.

1.1. Objetivos

Sistemas IoT vão ser cada vez mais comuns e presentes no dia-a-dia. No entanto, ainda não há unanimidade sobre qual a melhor forma de interação com um sistema deste tipo.

Neste projeto pretende-se efetuar um estudo sobre qual a melhor forma de interação com um sistema IoT, ou seja, pretende-se estudar diferentes formas de dar feedback aos utilizadores de um sistema IoT sobre os dados que estão a ser recolhidos e permitir que o utilizador possa interagir com o sistema, quer para ajustar alguns parâmetros às suas necessidades, quer para a visualização de dados, de forma a poder inferir algo.

Para o estudo de diferentes formas de interação do utilizador com um sistema IoT foi desenvolvido um sistema que permite a visualização de dados e controlo de diferentes atuadores, através de diferentes meios de interação, tais como, uma aplicação web, uma aplicação móvel e um assistente virtual.

1.2. Estrutura do relatório

Este relatório é constituído por um total de seis capítulos: Introdução, Estado da Arte, Solução, Testes e Resultados, Conclusão e Trabalho Futuro e Referências. No primeiro capítulo, é feita uma introdução ao tópico deste projeto, juntamente com uma secção que explica os objetivos do mesmo e a forma como este documento está estruturado.

O segundo capítulo, analisa os principais protocolos de comunicação usados em sistemas IoT, explora os principais assistentes virtuais da atualidade e analisa projetos de outros sistemas IoT em vários aspetos.

O terceiro capítulo, apresenta os requisitos funcionais e não funcionais do sistema, cenários de utilização, diagramas de casos de utilização e é explicada as tecnologias utilizadas no desenvolvimento do projeto, assim como as motivações para as opções escolhidas. Finalmente, é apresentada a arquitetura final do sistema desenvolvido, bem como uma descrição detalhada de cada componente da arquitetura e a racionalidade das opções adotadas.

No quarto capítulo, são analisados os testes de usabilidade a todas as aplicações desenvolvidas e os resultados das interações dos utilizadores com o sistema desenvolvido. Para finalizar, no quinto capítulo, são apresentadas as conclusões finais, possibilidades de desenvolvimento de trabalho futuro e propostas de melhoria e no sexto capítulo são apresentadas as referências bibliográficas.

2. Estado da arte

Neste capítulo são abordados diversos estudos/projetos de sistemas IoT aplicados em várias vertentes, são analisados os principais protocolos de comunicação utilizados em sistemas IoT e são explorados os principais assistentes virtuais da atualidade.

2.1. Artigos relacionados

Nesta secção são analisados projetos e artigos que exploram a temática da interação entre utilizador e máquina, em sistemas IoT. São examinados alguns projetos que apresentam diferentes formas de interação, seja por voz, gestos, aplicações móveis, etc. São uma exposição de diferentes formas de interação com sistemas IoT, sendo que alguns deles apresentam problemas, mais concretamente relativos a controlo por voz, problemas que serão explorados no âmbito deste projeto. A pesquisa destes artigos foi feita recorrendo a ferramentas web, como o Google Scholar e a biblioteca digital acm.

No projeto [8], foi desenvolvida uma aplicação ativada por voz, que permite uma melhor interação entre professores e alunos, proporcionando uma aprendizagem mais interativa e sólida. Quer o professor quer os alunos, podem invocar o dispositivo com a sua voz e fazer perguntas relacionadas com o tema do livro. O dispositivo toma a voz como input, analisando-a com inteligência artificial e machine learning e proporciona feedback aos utilizadores.

No [9], foi desenvolvido um assistente pessoal (RasPi) usando Raspberry Pi, que permite o controlo de equipamentos elétricos em casa, através de comandos de voz ou gestos.

Relativamente ao projeto [10], foi desenvolvido um espelho interativo, que permite a interação com o utilizador através de comandos de voz ou através de uma aplicação móvel. Com este espelho, seria possível ao utilizador visualizar dados como, a temperatura numa divisão da casa, a humidade, etc.

Quanto ao [11], é explorada também a influência da “awareness” do utilizador nos dados que estão a ser recolhidos num espaço. Uma falta de awareness por parte do utilizador sobre os dados a serem recolhidos num espaço, pode levar a problemas de desconfiança, frustração e a uma falta de oportunidade em obter feedback da parte dos utilizadores. A inclusão de pessoas no processo de recolha de dados analíticos pode melhorar a interpretação dos dados gerados pelo sistema através de interpretações e feedback dos utilizadores. De forma a explorar a influência do utilizador, foi desenvolvido o IoT sensor visualizer, que permite aos utilizadores visualizar os dados através de gráficos, em ecrãs posicionados em lugares públicos. Para a recolha dos dados, foi usado o “IoT egg”, dispositivo em forma de ovo, que permite a recolha de dados como, humidade, luz, gestos, etc. De modo a fornecer feedback ao utilizador, o IoT egg possui uma lâmpada LED e um motor de vibração. Todos os dados foram armazenados numa base de dados NoSQL (MongoDB).

Ainda no projeto [12], foi implementado um sistema que usa uma interação multimodal, ou seja, há uma combinação de diferentes formas de interação ao executar uma ação, sendo que, no caso deste projeto, foi usada uma combinação entre a voz e gestos. Para

tal, desenvolveram um sistema chamado Minuet, que usa UWB (Ultra Wide Band) e sensores de movimento para detetar com precisão qual o dispositivo IoT que o utilizador está a apontar, bem como para reconhecer a interação de controlo expresso por voz, gesto, ou uma combinação de ambos. Foram realizados testes com utilizadores e a grande maioria aceitou bem o sistema. Contudo, alguns problemas foram identificados, como por exemplo, o tempo que demorava a efetuar o reconhecimento de um objeto a ser apontado e problemas em interpretar o pedido do utilizador, pois alguns dos participantes não falavam inglês como língua nativa, o que levava a falhas na interpretação. O último problema apontado também foi uma oportunidade a ser explorada neste projeto, já que, falhas de pronúncia de palavras em inglês é algo a ter em conta.

Continuando, no estudo [13] efetuou-se uma análise aos desafios para desenvolver heurísticas específicas para interfaces de voz. Neste estudo, defende-se que as interações de voz devem ser baseadas em várias etapas evolutivas que constroem as heurísticas VUI, a partir dos princípios de design existentes nas interfaces GUI. A interação por fala é diferente da interação por GUI, no entanto, as diretrizes GUI fornecem uma base útil e são preferíveis ao desenvolvimento de heurísticas de raiz para as interfaces de voz. Também foram apresentados problemas ao uso de interfaces de voz, como por exemplo, os utilizadores não saberem como efetuar pedidos, já que não há pistas visuais, terem que se lembrar da lista de comandos possíveis e o facto deste tipo de interface ser inútil para pessoas com problemas de audição ou fala. Os problemas acima listados vão servir de motivação para a criação da skill da Alexa, de forma a averiguar se os mesmos acontecem e, caso aconteçam, o porquê de acontecerem.

Para finalizar, no documento [14], foi desenvolvido o Deus EM Machina, um sistema que permite ao utilizador, através do telemóvel, identificar um dispositivo IoT próximo e fornecer controlo personalizado, para seja possível interagir com esse dispositivo. Este sistema foi desenvolvido para facilitar a interação do utilizador com inúmeros dispositivos IoT, visto não ser muito prático navegar constantemente entre as inúmeras aplicações para cada tipo de dispositivo. O sistema consiste num smartphone, equipado com um sensor, que efetua leituras de emissões eletromagnéticas dos dipositivos e, através do processamento do sinal e de classificação (machine learning), é possível identificar o dispositivo. Todo o processo de identificação ocorre no telemóvel, através de um serviço executado em background. Este projeto é bastante semelhante ao Span-to-it [15], sendo que, uma das diferenças, é que o último usa a câmara como forma de recolher as imagens dos dispositivos, que posteriormente são classificadas.

2.2. Protocolos de comunicação

Um protocolo de comunicação é um sistema que permite, a duas ou mais entidades de um sistema de comunicação, comunicar e transmitir dados entre si através de qualquer tipo de variação de uma quantidade física [16]. Os protocolos são, em suma, conjuntos de regras de *hardware*/*software* que os pontos finais de comunicação devem seguir a fim de trocar algum tipo de informação. Abrangem a autenticação, a deteção e a sinalização de erros, além de incluir técnicas sofisticadas para recuperar de erros de transmissão e para codificar e descodificar dados [17]. Os protocolos podem ser implementados por *hardware*, *software*, ou por uma combinação de ambos [16].

2.2.1. Protocolos da camada de aplicação

2.2.1.1. HTTP

O *Hypertext Transfer Protocol* (HTTP) é um protocolo de comunicação para sistemas de informação distribuídos e colaborativos que permite aos utilizadores comunicar e trocar dados. Este protocolo de comunicação baseia-se no padrão cliente-servidor em que comunicam utilizando o método request-response.

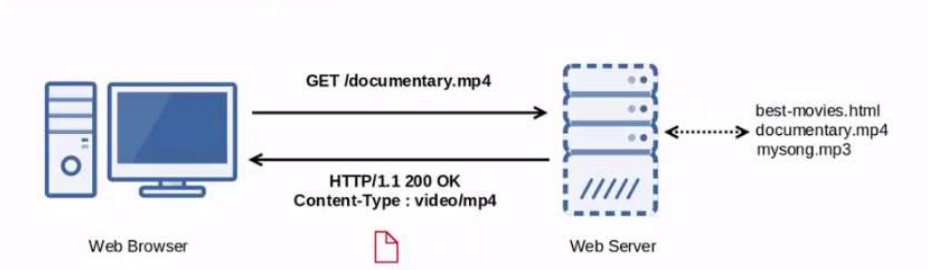


Figura 1. Exemplo de pedido HTTP do tipo GET [18].

Na figura 1 está representado um pedido GET ao servidor e a resposta é concedida pelo servidor. A porta predefinida é a porta 80, mas outras portas também podem ser usadas. O HTTP dá aos utilizadores uma forma de interagir com recursos web como ficheiros HTML, transmitindo mensagens de hipertexto entre clientes e servidores. Os clientes HTTP geralmente utilizam ligações TCP (*Transmission Control Protocol*) para comunicar com os servidores [19]. O protocolo HTTP define um conjunto de métodos de requisição responsáveis por indicar a ação a ser executada para um dado recurso.

GET → o método GET é utilizado para recuperar dados de um servidor recorrendo a um URI em particular. Um pedido do tipo GET apenas recupera dados e não efetua qualquer alteração.

POST → um pedido POST é utilizado para enviar dados para o servidor, por exemplo, informações do cliente, carregamento de ficheiros.

PUT → um pedido PUT modifica diretamente um recurso web existente ou cria um novo URI, se necessário.

HEAD → o mesmo que GET, mas apenas transfere a linha de estado e a secção de cabeçalho.

DELETE → remove todas as representações atuais do recurso alvo dadas por um URI.

OPTIONS → mostra aos utilizadores que métodos HTTP estão disponíveis para um URL específico.

CONNECT → permite converter a ligação do pedido num túnel TCP/IP transparente.

TRACE → realiza um teste de *loopback* enviando uma mensagem por todo o caminho até ao recurso alvo para o qual foi destinado [19].

Os dispositivos usados em sistemas IoT possuem poder computacional restrito, o que limita a utilização do protocolo HTTP. Como alternativa ao protocolo HTTP tem-se o CoAP e o MQTT, que são dois protocolos da camada de aplicação específicos para recuperar informações de dispositivos com baixo poder computacional.

2.2.1.2. CoAP

O CoAP (*Constrained Application Protocol*) é um protocolo de comunicação da camada de aplicação, que foi concebido como um protocolo leve máquina-a-máquina (M2M) que pode funcionar com dispositivos em que a memória e os recursos são escassos. O CoAP é muito semelhante ao HTTP, mas em vez de funcionar em cima de pacotes TCP, funciona em cima de UDP. Tal como outro protocolo baseado em UDP ou TCP, é suscetível à falsificação de endereços IP e à amplificação de pacotes, os dois principais fatores que permitem a amplificação de um ataque DDoS [20]. Devido ao facto do protocolo TCP não ser utilizado, o CoAP é responsável pelo ACKs das mensagens, pelo controlo de congestionamento e deteção de mensagens duplicadas.

O CoAP, tal como o HTTP, é um protocolo cliente-servidor, em que o cliente faz um pedido e o servidor envia de volta uma resposta. Os métodos utilizados pelo CoAP são os mesmos utilizados pelo HTTP. O CoAP suporta 4 tipos de mensagens: *Confirmable*, *non confirmable*, *acknowledgement* e *reset* [21][22]. Uma mensagem *confirmable* é uma mensagem fiável, onde o cliente tem a certeza de que a mensagem será entregue ao servidor. O servidor, por sua vez, ao responder ao pedido do cliente, irá enviar uma mensagem *acknowledge* que irá conter o mesmo ID da mensagem *confirmable* enviada pelo cliente.

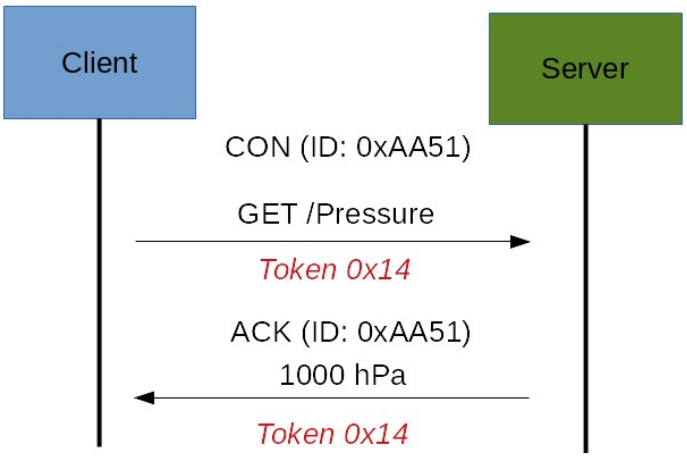


Figura 2. Exemplo de mensagem confirmable [21].

Na eventualidade do servidor ter problemas em responder ao pedido, poderá enviar uma mensagem *reset* ao cliente. No que diz respeito às mensagens *non confirmable*, como o próprio nome indica são mensagens que não requerem um *acknowledge* por parte do servidor, ou seja, são mensagens pouco fiáveis, geralmente usadas no transporte de dados das leituras dos sensores.

2.2.1.3. MQTT

O MQTT (*MQ Telemetry Transport*) é outro protocolo de comunicação da camada da aplicação que permite, em topo do TCP/IP, o transporte de mensagens de *publish*/*subscribe* extremamente leves. O protocolo MQTT é uma boa alternativa para as redes sem fios que sofrem taxas de latência variáveis, devido a limitações ocasionais de largura de banda ou ligações instáveis [23].

Este protocolo é caraterizado por dois tipos de elementos: um *broker* e vários clientes (*publishers* e *subscribers*). O *broker* é responsável por receber todas as mensagens dos clientes e, depois disso, direcioná-las/roteá-las para os clientes alvo desejados. Um cliente é qualquer elemento que interage com o *broker* com o objetivo de receber e enviar mensagens. Um cliente pode ser, para ilustração, um sensor ou uma aplicação [24]. O facto de as mensagens estarem organizadas por tópicos permite que certos clientes subscritos possam interagir apenas com certas mensagens. Na figura 3 é possível visualizar um exemplo do funcionamento do protocolo MQTT.

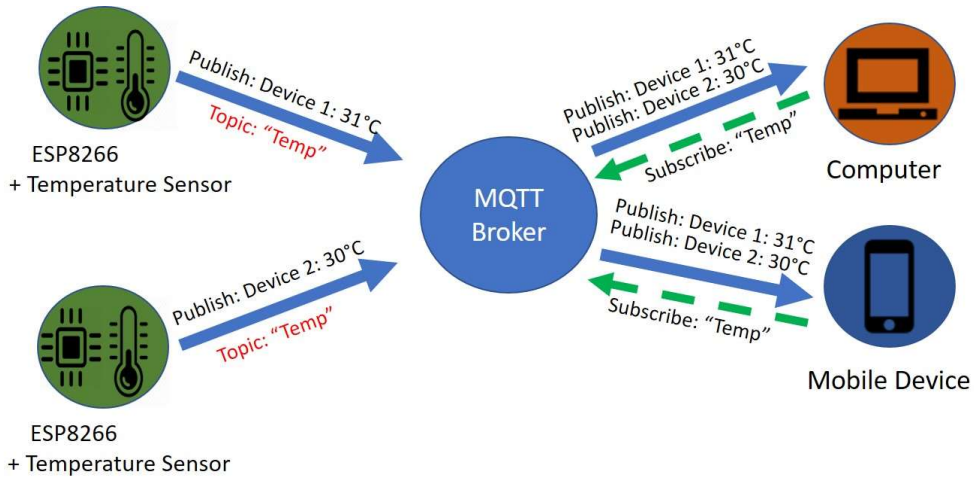


Figura 3. Exemplo de trocas de mensagens através do protocolo MQTT [25].

O MQTT suporta 3 níveis de QoS (*Quality of Service*). No nível 0 a mensagem é enviada apenas uma vez e o cliente e o *broker* não tomam medidas adicionais para confirmar a entrega. Este nível não é confiável visto que, se o cliente estiver indisponível no momento, perderá a mensagem. No nível 1 existe a confirmação de entrega de uma mensagem. Este nível de QoS utiliza uma sequência de pacotes *PUBLISH/PUBACK* entre o *broker* e o seu *publisher*, assim como entre o *broker* e os subscritores. Um pacote de confirmação verifica se

o conteúdo foi recebido e caso a confirmação não chegue dentro de um intervalo de tempo razoável a mensagem será reenviada com o conteúdo original.

Para finalizar, o nível 2 garante que apenas uma cópia da mensagem é enviada e recebida.
Este nível é o mais alto nível de serviço no MQTT. Garante que cada mensagem é recebida apenas uma vez pelos destinatários, mensagens estas armazenadas pelo *broker*. É o nível de qualidade de serviço mais seguro e consequentemente mais lento. Este nível de QoS entrega a mensagem com dois pares de pacotes. O primeiro par é chamado *PUBLISH/PUBREC*, e o segundo par é chamado *PUBREL/PUBCOMP*. Os dois pares garantem que, independentemente do número de novas tentativas, a mensagem só será entregue uma vez [23][24][26]. Na figura 4 está exemplificada como funciona a comunicação com um QoS de nível 2.

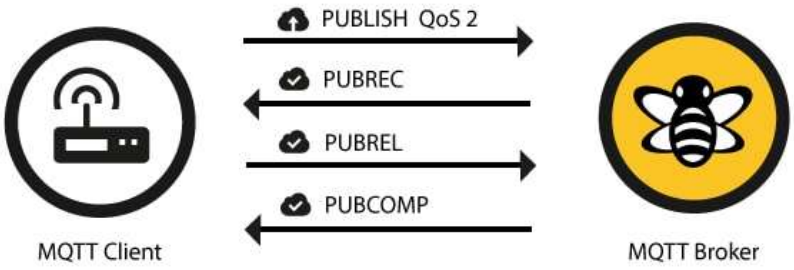


Figura 4. Troca de mensagem entre um cliente e o broker com QoS nível 2 [27].

2.2.2. Protocolos da camada ligação de dados e física

2.2.2.1. WiFi

O WiFi é talvez a tecnologia de comunicação mais usada para a comunicação entre dispositivos devido à sua grande divulgação e infraestrutura, visto que utiliza uma vasta gama de infraestruturas, é capaz de realizar transferências de dados até centenas de megabits por segundo e de lidar com enormes quantidades de transferências de dados. Opera nas frequências de 2.4 GHz e 5GHz. O WiFi é baseado na família de normas IEEE 802.11 e a sua primeira versão foi lançada em 1997. Esta versão era capaz de fornecer até 2Mbit/s de velocidade de ligação. Atualmente, a norma mais comum de WiFi é a 802.11n, que se baseia na IEEE 802.11 [28]. A principal desvantagem do Wi-Fi é o maior consumo de energia, quando comparado com outras tecnologias de comunicação. Contudo, algumas das suas vantagens são o seu alcance, que ronda os 50 metros e a quantidade de dados transmitidos, o que o torna adequado para navegação na Internet em vários dispositivos. De todos os meios de comunicação, este é o que permite uma maior taxa de transferência de dados.

2.2.2.2. ZIGBEE

O ZigBee é um protocolo IoT que permite a objetos inteligentes trabalharem em conjunto através de ligações de rede em malha - ligações a outros dispositivos conectados - para ligar os seus dispositivos uns aos outros e à Internet [29]. Geralmente mais usado em ambientes industriais, o ZigBee é utilizado com aplicações que têm suporte a uma baixa taxa de transferência de dados em curtas distâncias. Os principais atributos que tornam o ZigBee

adequado para uma comunicação eficaz entre dispositivos IoT são, o baixo consumo de energia, a elevada escalabilidade, a segurança e a durabilidade, além de poder possuir uma grande quantidade de nós na rede que pode atingir um total de 1024 nós. É baseado no standard IEEE 802.15.4, opera em frequências que rondam os 2.4 GHz, possui um alcance que vai dos 10 metros até aos 100 metros e pode alcançar uma taxa de transferência de dados na ordem dos 250 kbps [28].

2.2.2.3. BLE

O Bluetooth é um dos protocolos mais utilizados para comunicações de curto alcance, geralmente integrado na maioria dos *smartphones* e dispositivos móveis, o que constitui uma grande vantagem para os produtos pessoais. Trata-se de um protocolo normalizado para a transmissão de dados sem fios. Este protocolo de comunicação é seguro e perfeito para a transmissão de curto alcance, de baixa potência, de baixo custo e sem fios entre dispositivos. O BLE (*Bluetooth Low Energy*) é uma versão de baixo consumo de energia do protocolo Bluetooth, que reduz o consumo de energia na troca de dados entre entidades de um sistema IoT. Este meio de comunicação funciona na frequência de 2.4 GHz, tem um alcance entre os 50 e 100 metros e taxas de transferência de dados na ordem dos 1Mbps [30].

2.2.2.4. LoRaWan (Long Range Wide Area Network)

O LoRaWan é um protocolo de baixa potência e de longo alcance que permite a deteção de sinais abaixo do nível de ruído. Este protocolo permite a ligação entre dispositivos à Internet, tanto em redes privadas como em redes globais. É mais aplicável num contexto de cidades inteligentes, onde existem milhões de dispositivos que funcionam com menos energia e memória. Um exemplo em que este tipo de protocolo pode ser aplicado numa cidade inteligente é no caso da gestão da iluminação pública. A taxa de comunicação alcança valores entre 0.3 kbps a 50 kbps [28]. O consumo de energia é pequeno, o que permitirá aos dispositivos se manterem ativos por períodos mais longos de tempo. Este protocolo tem uma larga escala de frequência em que pode trabalhar e um alcance que varia entre os 2 a 5 km em ambientes urbanos e até 15 km em ambientes rurais [30].

2.2.2.5. Z-Wave

O Z-Wave é um protocolo de comunicação de radiofrequência de baixa potência, utilizado principalmente para sistemas de automação doméstica e outros dispositivos eletrónicos, tais como lâmpadas inteligentes e sensores. Com a utilização deste protocolo, existirão menos interferências, visto que, outros protocolos de comunicação sem fios, tais como WiFi, Bluetooth e ZigBee, operam na gama dos 2,4 GHz e este protocolo funciona numa frequência que ronda dos 900 MHz. O seu débito de dados varia entre 40kbps e 100kbps e o seu alcance varia entre os 30 e 100 metros [28][31]. Ao contrário do Wi-Fi, em que os dispositivos se ligam a um router/AP central, os dispositivos Z-Wave estão todos ligados entre si para formar uma rede em malha, ou seja, não existe a necessidade de um nó coordenador, o que lhe permite ser muito escalável, com um controlo de até 232 dispositivos [31].

2.3. Assistentes virtuais

Assistente Virtual Inteligente (AVI) ou Assistente Pessoal Inteligente (API) é um agente de software que pode executar tarefas ou serviços para os utilizadores, com base em comandos ou pedidos [32]. Pode ajudar com atividades diárias, tais como marcação de compromissos, realização de chamadas (áudio/vídeo) e muito mais. Há dois tipos principais de input nos AVI: a interface de texto e interface de voz. O objetivo é que os utilizadores sejam capazes de interagir com o AVI usando a sua linguagem natural. Estes assistentes usam técnicas de processamento de linguagem natural, de forma a mapear a voz do utilizador para comandos executáveis.

Os AVI permitem às empresas prestar serviços personalizados a todos os contactos e clientes. A capacidade de comunicar de forma humana é uma enorme melhoria em relação às interações empresariais existentes, já que estas, até há pouco tempo, eram limitadas pelas capacidades dos funcionários ou, em grande parte, por programas de email [33]. Outro ponto bastante importante de um AVI é a componente de *machine learning*, que se refere à capacidade de aprender sem ser explicitamente programado e tornar-se "mais inteligente" com informação que recebe.

Geralmente, os AVIs utilizam o microfone do dispositivo para receber pedidos de voz, e a saída de voz é executada no altifalante. De forma a interpretar o pedido do utilizador e devolver uma resposta, é usada uma combinação de várias tecnologias diferentes: reconhecimento da fala, análise da fala e processamento da linguagem. Quando o utilizador pede ao assistente para executar uma tarefa, o sinal áudio da linguagem natural é convertido em dados digitais, que podem ser analisados pelo software. Depois, são usados algoritmos para comparar estes dados com a base de dados, para encontrar a resposta certa. A base de dados está localizada num servidor distribuído na *cloud*. Por este motivo, sem uma ligação fiável à Internet, a maioria dos assistentes pessoais não funciona. À medida que o número de consultas aumenta, a base de dados do software é expandida e otimizada, o que melhora o reconhecimento da fala e aumenta o tempo de resposta do sistema [34].

O ecossistema do AVI é composto por três partes principais. Na *cloud* o centro de processamento de pedidos analisa e responde aos comandos de texto/voz, e realiza a operação solicitada. Do lado do utilizador, existem dispositivos terminais como o Amazon Echo, ou no caso do Cortana o PC. Estes dispositivos têm aplicações incorporadas que comunicam com o assistente virtual.

No entanto, estes tipos de sistemas apresentam potenciais problemas de segurança, que, no entanto, não são objeto deste projeto. O diálogo do AVI pode ser acedido através de dispositivos como o Echo, o que representa um risco de privacidade, pois as gravações de voz podem fornecer informações de identificação pessoal, ou entidades não autorizadas podem utilizar os dados para identificar os utilizadores. Alguns dos problemas identificados foram *packet sniffing*, ou seja, a possibilidade de haver uma captura de dados durante a comunicação entre o assistente e o meio que o utilizador usou para efetuar o pedido. Isto acontece porque nem todo o tráfego da rede é transmitido através de uma ligação segura. Outro problema é que os próprios dispositivos IoT podem ficar comprometidos. Câmaras e outros dispositivos podem ficar infetados com *malware* ou sofrer ataques DDoS. Um

problema adicional podem ser comandos maliciosos, em que um atacante pode passar-se pelo utilizador legítimo. Para finalizar, pode acontecer que o assistente grave as conversas sem ser essa a intenção do utilizador, pelo facto deste falar dentro do alcance do dispositivo. Apesar desta situação acontecer acidentalmente, os dados são transmitidos de igual forma para a *cloud* [35].

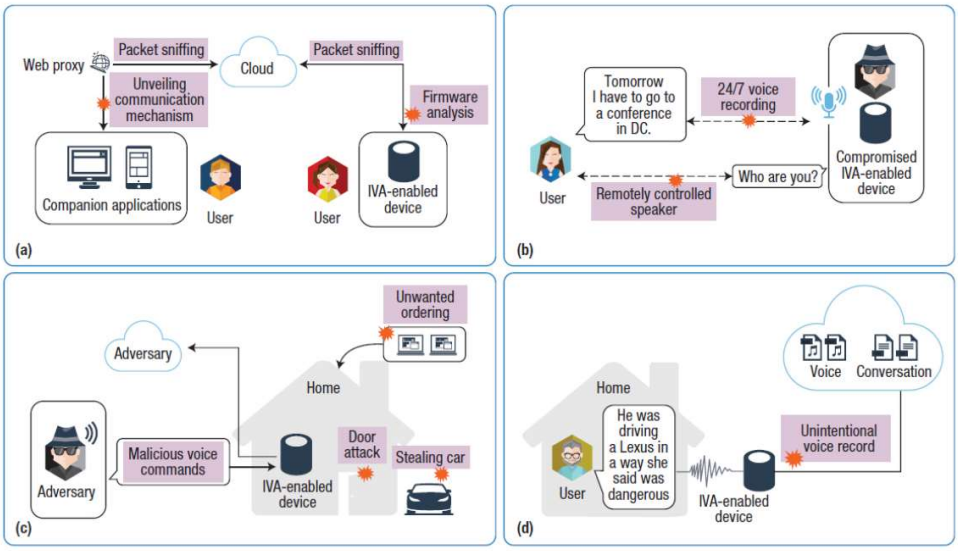


Figura 5. Problemas de segurança e de privacidade dos assistentes virtuais; a) packet sniffing; b) dispositivos IoT comprometidos; c) comandos de voz maliciosos; d) gravação de voz não intencional [35].

2.3.1. Exemplos de assistentes virtuais

Nesta subsecção serão apresentados os principais assistentes virtuais da atualidade disponíveis no mercado, sendo que os mais amplamente usados são a Alexa, a Siri, o Bixby, a Cortana e o Google Assistant.

2.3.1.1. Alexa

A Alexa é o AVI lançado pela Amazon [36]. Para o uso da Alexa podemos recorrer à aplicação fornecida pela Amazon ou aos dispositivos da família Echo, que foram lançados originalmente em 2014.

Figura 6. Altifalante Echo dot da Amazon [37].

A Alexa utiliza o sistema de processamento de linguagem natural da Wolfram para executar funções de inteligência artificial, tais como, traduzir texto para uma voz parecida com a voz humana, recolher dados, processar e gerar respostas precisas para os utilizadores. Uma vez ouvido um comando de voz ou um pedido, este é enviado para a *cloud* da Amazon, onde o processamento acontece e a resposta é devolvida. A Alexa é compatível com muitos altifalantes e dispositivos inteligentes de terceiros como, auscultadores, lâmpadas, fechaduras, câmaras de segurança, robôs, telefones, e muito mais. A Alexa também permite aos utilizadores criarem *skills,* isto é, funcionalidades extra que permitem estender as funcionalidades base da Alexa. A Alexa também apresenta algumas limitações, como por exemplo não é possível enviar email usando os comandos de voz e alguns resultados de pesquisa na Web podem não ser resultados de qualidade [38].

2.3.1.2. Siri

A Siri é um assistente pessoal para iOS, macOS e watchOS, para que possa ser usada no iPhone, iPad, iPod Touch, Apple Watch e Mac [39]. É propriedade da Apple e usa processamento de linguagem natural para responder a perguntas, fazer sugestões e realizar ações. Foi o primeiro assistente a ser lançado, em 2011. A principal diferença entre a Siri, o Google Assistant, a Alexa e outros assistentes é que a primeira está disponível apenas em dispositivos da Apple. Para executar uma tarefa na Siri diz-se "Hey Siri" num dispositivo Apple com suporte ao Siri.

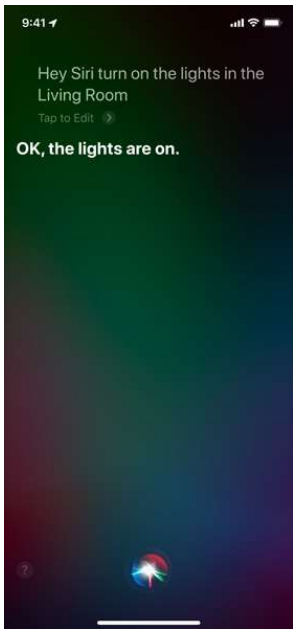


Figura 7. Exemplo de interação com a Siri [39].

A Siri pode realizar várias tarefas, como por exemplo: direções de navegação, lembretes, fazer procuras na web, informações gerais ou mudar definições/executar ações no dispositivo móvel. Algumas das limitações da Siri são o facto de apenas funcionar em dispositivos iOS e caso o utilizador efetue o pedido falando demasiado depressa pode levar a falhas na interpretação [38].

2.3.1.3. Bixby

O Bixby é o assistente virtual da Samsung [40]. Foi lançado originalmente com o Samsung Galaxy S8 em 2017. Este assistente fornece uma gama de funcionalidades como o Bixby Vision, que consiste numa câmara de realidade aumentada, que reconhece objetos em tempo real e fornece aos utilizadores as funções de compra online, tradução de texto, leitura de código QR e reconhecimento de pontos de referência, o Bixby Home, que geralmente fica no ecrã inicial do dispositivo, que consiste numa lista vertical de informação, como por exemplo meteorologia e atividade física, com a qual o Bixby pode interagir. Para finalizar, tal como com os outros assistentes, o Bixby tem a capacidade de criar lembretes (Bixby Routines) [41]. Por exemplo, pode-se pedir para lembrar de chamadas e mensagens de texto que o utilizador deseje fazer mais tarde e de conteúdo web que pretenda visitar posteriormente.

Uma funcionalidade que este assistente apresenta que os outros não possuem é o facto de possuir um botão físico em dispositivos da Samsung para invocar o assistente. Tal como os outros assistentes, o Bixby pode apresentar limitações em fornecer respostas diretas e falhar em apresentar pesquisas na web [38].

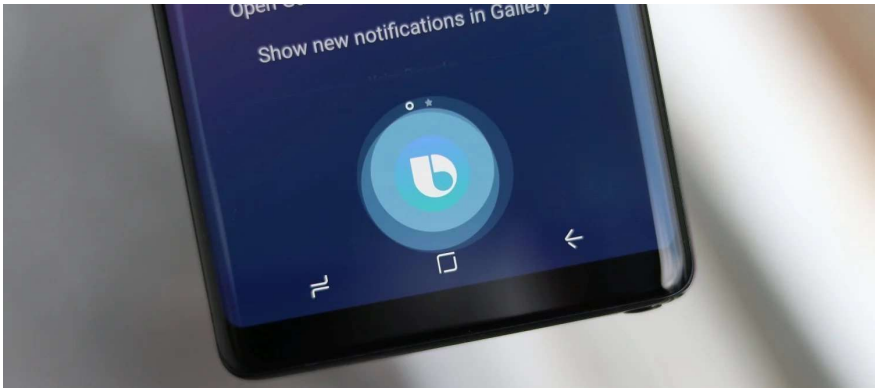


Figura 8. Exemplo de interação com o Bixby [42].

2.3.1.4. Cortana

A Cortana é o assistente virtual da Microsoft, lançado em abril de 2014 [43]. Por ser integrado no Windows 10 é bastante usado em PCs desktop e laptop. A Microsoft entrevistou assistentes pessoais humanos, num esforço de tentar desenvolver uma versão virtual mais natural. O Cortana processa capacidades de linguagem natural utilizando o serviço de resposta de voz interativa da Tellme Networks [44]. Utiliza o motor de busca Bing e os dados do utilizador para realizar tarefas e responder a perguntas dos utilizadores. A principal limitação deste assistente é o facto de o motor de busca padrão ser o Bing [38].

Figura 9. Exemplo de interação com a Cortana [45].

2.3.1.5. Google Assistant

O Google Assistant [46] foi originalmente lançado em maio de 2016, como parte do altifalante inteligente do Google Home. Inicialmente, era apenas uma aplicação de envio de mensagens, mas mais tarde evoluiu para o Google Assistant. O assistente do Google utiliza o algoritmo de processamento de linguagem natural do Google para conduzir uma conversa bidirecional [47]. Os resultados de pesquisa web que são apresentados são clicáveis. O Assistente do Google funciona muito bem ao dar respostas ao utilizador, enquanto dá dicas sobre como utilizar o assistente. De acordo com várias revisões na Internet, as respostas do assistente do Google são muito rápidas e precisas, especialmente porque funciona bem com reconhecimento de sotaque diferente [48]. No que diz respeito as suas limitações, algumas das suas características não estão disponíveis em todos os países [38].

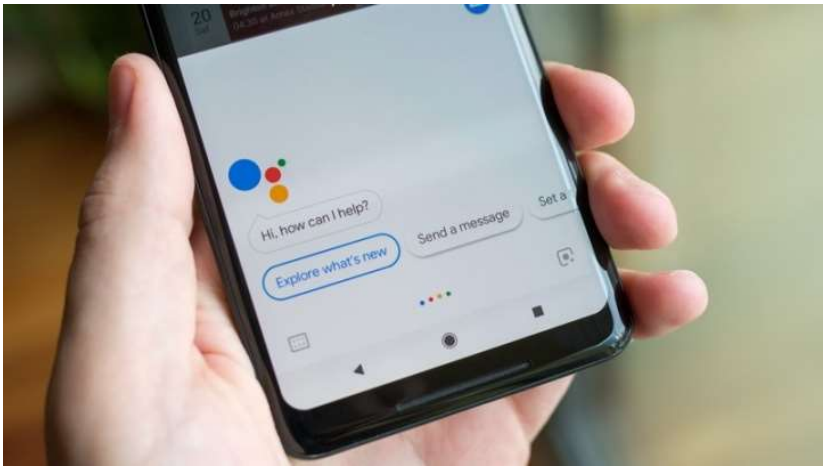


Figura 10. Exemplo de interação com o Google Assistant [49].

2.3.2. Comparação assistentes virtuais

A tabela 1 apresenta um resumo/comparação dos AVIs apresentados anteriormente.

Tabela 1. Comparação entre os principais assistentes virtuais [38].

	Siri	Cortana	Alexa	Google Assistant	Bixby
Data de lançamento	2011	2014	2014	2016	2017
Funciona com	Dispositivos Apple iOS - iPad, iPhone e iPod Touch	Dispositivos da Microsoft	Dispositivos Echo	Google apps, Chromecast, Google Home, & outros dispositivos Google	Dispositivos Samsung
Não funciona bem com	Gmail	Youtube	Youtube	Funciona bem com todas as aplicações de terceiros	Começou a trabalhar com aplicações de terceiros em 2018.
Conexão à Internet	WiFi ou dados móveis	WiFi ou dados móveis	WiFi	WiFi ou dados móveis	WiFi ou dados móveis
Formas de interação	Escrita e comandos de voz	Escrita e comandos de voz	Escrita e comandos de voz	Escrita e comandos de voz	Escrita e comandos de voz

3. Solução

Neste capítulo são apresentados os requisitos funcionais e não funcionais do sistema, alguns cenários de qualidade, o diagrama de casos de utilização, as tecnologias utilizadas no desenvolvimento do projeto e os respetivos esclarecimentos. Para finalizar, é apresentada a arquitetura final do sistema desenvolvido, juntamente com uma explicação detalhada de cada componente arquitetural e a justificação das opções tomadas.

3.1. Requisitos

O primeiro passo para o desenvolvimento do projeto foi a definição dos requisitos, sendo que, no total, o sistema tem 32 requisitos, 19 não funcionais e 13 funcionais. Os requisitos de software são a descrição das características e funções do sistema alvo, ou seja, funcionalidades que terão quer ser implementadas pelos desenvolvedores. Os requisitos transmitem as expectativas do utilizador em relação ao produto de software. Da perspetiva do cliente, estes requisitos podem ser óbvios ou ocultos, conhecidos ou desconhecidos, esperados ou inesperados. Os requisitos funcionais representam todos os requisitos que especificam funcionalidades que o sistema deve apresentar, ou seja, todos os aspetos funcionais do software, enquadram-se nesta categoria. Os requisitos não funcionais representam todos os requisitos que não estão relacionados com a funcionalidade do software. São características implícitas ou esperadas do software assumidas pelo utilizador. A elicitação dos requisitos foi efetuada durante as várias reuniões com o orientador, e assim, foi possível chegar a um consenso quanto ao que o sistema deveria ser capaz de efetuar. A lista de requisitos não funcionais e a lista dos funcionais encontra-se representada nas tabelas 2 e 3 respetivamente.

Tabela 2. Requisitos não funcionais do sistema.

ID	Tipo	Nome	Definição
01	Não Funcional - Segurança	Autenticação do website	O website deve permitir ao utilizador autenticar-se através das suas credenciais.
02	Não Funcional - Segurança	Autenticação da aplicação móvel	A aplicação móvel deve permitir ao utilizador autenticar-se através das suas credenciais.
03	Não Funcional - Segurança	Autenticação a partir aplicação da Alexa	O utilizador deve autenticar-se no sistema a partir da aplicação da Alexa.
04	Não Funcional - Segurança	Autenticação no pedido HTTP	Todos os pedidos para a API dos NodeMCUs deverão ter um mecanismo de autenticação por HTTP.
05	Não Funcional - Segurança	Autenticação por *tokens* na API principal	Todos os pedidos para a API principal deverão ter um *token* que identifique unicamente um utilizador.

06	Não Funcional - Segurança	Autenticação do broker MQTT na *cloud*	Caso o servidor central e o módulo microcontrolador estejam ligados ao *broker* na *cloud*, a troca de mensagens MQTT só é possível se estiverem autenticados.
07	Não Funcional - Segurança	Autenticação do *broker* MQTT local	Caso o servidor central e o módulo microcontrolador estejam ligados ao broker local do Raspberry, a troca de mensagens MQTT só é possível se estiverem autenticados.
08	Não Funcional - Segurança	Renovação de *tokens*	Os *tokens* de autenticação devem ser renovados a cada 5 minutos.
09	Não Funcional - Segurança	Proteção ataques CSRF	O sistema deverá fornecer proteção a ataques do tipo CSRF.
10	Não Funcional - Fiabilidade	Armazenar informação de rede localmente	Informações sobre a rede local deverão ser armazenadas na memória do NodeMCU.
11	Não Funcional - Fiabilidade	Armazenar dados dos atuadores localmente	Dados relativos aos atuadores devem ser armazenados na memória do NodeMCU.
12	Não Funcional - Fiabilidade	Armazenamento dos dados recolhidos pelos sensores numa base de dados	Todos os dados recolhidos pelos sensores e ações sobre os atuadores, deverão ser armazenados numa base da dados remota.
13	Não Funcional - Fiabilidade	Armazenamento dos dados no *broker*	Em caso de falha de comunicação com o servidor, deverá ser garantida entrega das mensagens ao usar QoS de nível 2.
14	Não Funcional - Fiabilidade	*Broker* secundário	Em caso de falha de comunicação com o *broker* na *cloud,* o sistema deverá utilizar o *broker* local do Raspberry.
15	Não Funcional - Fiabilidade	Armazenamento do fluxo de dados num ficheiro local	O sistema deverá garantir que todo o fluxo de dados no sistema, quer sejam ações que o utilizador realiza ou novos dados que chegam ao sistema, são armazenados num ficheiro local.
16	Não Funcional - Usabilidade	Interagir por voz	O utilizador pode interagir por meio da voz com o sistema.
17	Não Funcional - Usabilidade	Ajuda por parte do assistente virtual para efetuar *account linking*	O assistente virtual deverá instruir o utilizador em como efetuar o *account linking.*

18	Não Funcional - Usabilidade	Ajuda por parte do assistente virtual ao fornecer exemplos.	O assistente virtual deverá instruir o utilizador em como efetuar pedidos à *skill*.
19	Não Funcional – Desempenho	Resposta por parte da Alexa em menos de 10 segundos.	O assistente virtual deverá responder aos pedidos do utilizador em menos de 10 segundos.

Tabela 3. Requisitos funcionais do sistema

ID	Tipo	Nome	Definição
1	Funcional	Autenticação na rede WiFi	O utilizador pode ligar o sistema à sua rede local através das suas credenciais.
2	Funcional	Criação de novo utilizador	O utilizador pode criar uma conta de utilizador no sistema.
3	Funcional	Criação de novo nodo	O utilizador pode criar um nodo de atuação e recolha de dados no sistema.
4	Funcional	Consulta de dados em tempo real	O utilizador pode consultar os dados em tempo real de múltiplos nodos simultaneamente.
5	Funcional	Consulta de dados por nodo	O utilizador pode consultar os dados de um nodo em particular.
6	Funcional	Consulta de dados históricos por nodo	O utilizador pode consultar dados de uma data específica para um dado nodo.
7	Funcional	Ligar/desligar led	O utilizador pode ligar e desligar a led de um dado nodo.
8	Funcional	Mudar cor de led	O utilizador pode mudar a cor de uma led de um dado nodo.
9	Funcional	Mudar intensidade de led	O utilizador pode mudar a intensidade de uma led de um dado nodo.
10	Funcional	Mudar nome de um nodo	O utilizador pode mudar o nome de um dado nodo.
11	Funcional	Apagar nodo	Um utilizador pode apagar um nodo e todos os seus dados associados.
12	Funcional	Consulta de dados históricos de nodos inativos	O utilizador pode consultar dados de uma data específica para um dado nodo que esteja desativado.

13	Funcional	Mudar frequência de envio de dados	O utilizador pode alterar a frequência na qual os dados são enviados para o sistema.

3.2. Cenários de qualidade

Nesta secção são definidos quatro cenários de qualidade partir de alguns dos requisitos não funcionais mencionados na tabela 2. A partir destes cenários vai ser possível fazer uma descrição mais detalhada destes requisitos, através de uma descrição estímulo-resposta. A tabela 4 descreve o cenário de qualidade de segurança referente ao requisito 5, na qual descreve a necessidade de garantir que não aja acessos não autorizados aos recursos disponibilizados pela API do servidor central.

Tabela 4. Cenário de qualidade de segurança do requisito 5

Atributo de qualidade	**Segurança.**	
Preocupação do atributo	Garantir que não aja acessos não autorizados à API do servidor central.	
Cenário	É efetuado um pedido não autorizado à API do servidor central.	
	Fonte do estímulo	Utilizador/Sistema.
	Estímulo	Pedido não autorizado à API do servidor central.
	Artefacto	Servidor central.
	Ambiente	Normal.
	Resposta	O servidor central nega acesso aos recursos.
	Medida de resposta	Garantia da proteção de dados protegidos.

O cenário descrito na tabela 5 retrata a ajuda que o assistente virtual fornece ao utilizador de forma a que o mesmo possa utilizar a *skill* e efetuar *account linking*.

Tabela 5. Cenário de qualidade de usabilidade do requisito 17

Atributo de qualidade	**Usabilidade.**	
Preocupação do atributo	Assistir o utilizador no uso da *skill* da Alexa.	
Cenário	O utilizador não sabe como efetuar *account linking*.	
	Fonte do estímulo	Utilizador final.
	Estímulo	Interação com a Alexa.
	Artefacto	Alexa.
	Ambiente	Tempo de execução.

	Resposta	A Alexa deve fornecer informações sobre como o utilizador deve efetuar *account linking*.
	Medida de resposta	Capacidade de efetuar *account linking*, imediatamente após a ajuda da Alexa.

O cenário da tabela 6 especifica o tempo máximo que o assistente virtual tem para responder ao pedido do utilizador, de forma a não afetar a usabilidade.

Tabela 6. Cenário de qualidade de desempenho do requisito 19

Atributo de qualidade	**Desempenho.**	
Preocupação do atributo	Fornecimento de respostas por parte da Alexa ao utilizador em tempo útil.	
Cenário	O utilizador efetua um pedido à Alexa e recebe a resposta de volta em menos de 10 segundos.	
	Fonte do estímulo	Utilizador final.
	Estímulo	Efetuação de um pedido.
	Artefacto	Servidor central e AWS Lamdba.
	Ambiente	Normal.
	Resposta	Processar o pedido e enviar reposta no formato JSON.
	Medida de resposta	A resposta deve ser concedida em menos de 10 segundos.

Tabela 7. Cenário de qualidade de fiabilidade do requisito 13

Atributo de qualidade	**Fiabilidade.**	
Preocupação do atributo	Garantir que não aja perda de dados.	
Cenário	Falha na comunicação com o servidor central por parte do microcontrolador.	
	Fonte do estímulo	Microcontrolador.
	Estímulo	Envio de informação através do protocolo MQTT.
	Artefacto	Servidor central e microcontrolador.
	Ambiente	Normal.

	Resposta	Armazenamento das mensagens no broker.
	Medida de resposta	Garantia do envio das mensagens em falta.

Para finalizar, na tabela 7 está representado o cenário de qualidade de fiabilidade referente ao requisito 13. Neste cenário é descrita a situação de estímulo-resposta na eventualidade do servidor central estar incomunicável.

3.3. Diagrama casos de utilização

Um diagrama de caso de utilização é uma representação gráfica e semântica da interação de vários tipos de utilizadores com o sistema. Na figura 11 está representada o diagrama com base nos requisitos funcionais da tabela 3. Neste diagrama está apresentado as interações dos três tipos de utilizadores, ou seja, um utilizador da aplicação web, móvel e do assistente virtual.

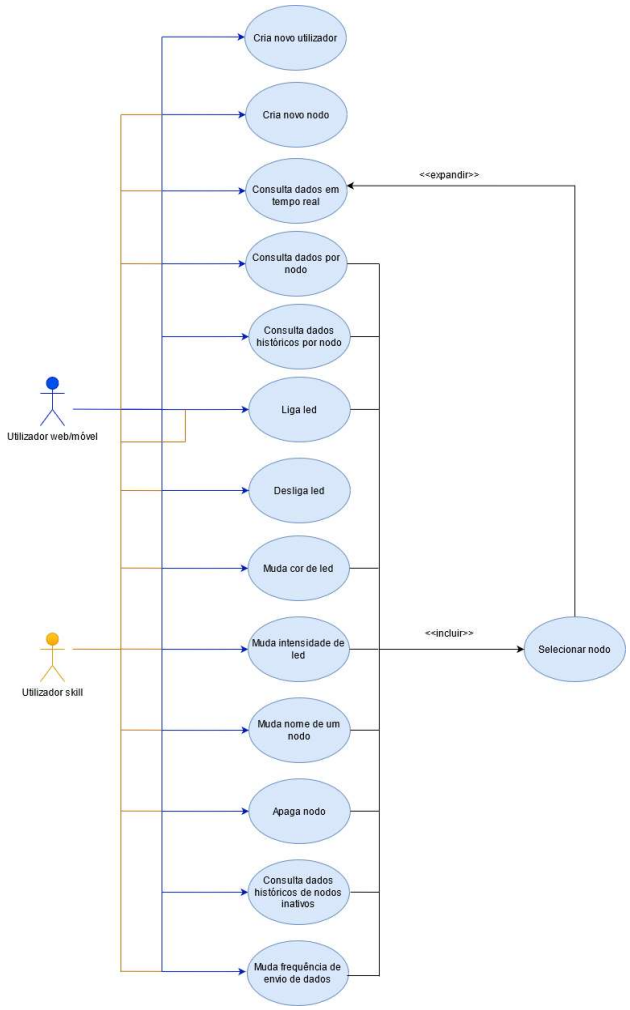


Figura 11. Diagrama casos de utilização.

3.4. Tecnologias

Nesta seção são apresentadas as diversas tecnologias utilizadas para o desenvolvimento do sistema, quer de *hardware* quer de *software*.

3.4.1. Hardware

3.4.1.1. NodeMCU

O NodeMCU é uma plataforma de desenvolvimento *opensource* orientada para o desenvolvimento de projetos IoT. Este microcontrolador é constituído por um chip controlador (ESP8266 ou ESP-12E), uma porta micro USB para alimentação e programação e um conversor USB serial integrado. Os casos mais comuns de aplicação desta placa são para redes de sensores, redes malha, projetos de automação, monitorização de ambientes industriais, domótica, etc [50]. As características técnicas podem ser encontradas na tabela 8.

Tabela 8. Especificações técnicas do NodeMCU [50].

Wi-fi 2,4 Ghz com suporte para as normas 802.11 b/g/n
Alimentação: 5V via porta micro USB
Processador Tensilica LX106 – até 160MHz
Pilha protocolar TCP/IP (apenas IPv4)
Botão de Reset e Flash
Memória RAM: 96kBytes
Memória ROM boot: 64 KBytes
Memória flash: 4 MB
Potência de saída: 0,15 (W); +19.5dBm em modo 802.11b
Consumo em modo de baixa energia: > 10 uA
GPIO com funções de PWM, I2C, SPI, etc
Conversor analógico digital (ADC)
Dimensões: 49 x 24.5 x 13mm
Modos de operação: STA/AP/STA+AP

Ao contrário do Arduino Uno, este microcontrolador possui um módulo WiFi, o que permite a construção de páginas web e servidores. Uma das grandes vantagens em trabalhar com o NodeMCU, é a possibilidade de se programar utilizando a IDE do Arduino. O NodeMCU é um *firmware* baseado em Lua da ESP8266. Uma alternativa de programação para o microcontrolador seria escrever scripts Lua para o NodeMCU. No entanto, também é possível, como mencionado anteriormente, utilizar o ambiente de desenvolvimento Arduino, o que irá facilitar a vida dos programadores que já estão familiarizados com essa plataforma. Este microcontrolador contém no total 11 portas GPIO e uma porta analógica, e pode ser encontrado na figura 12.

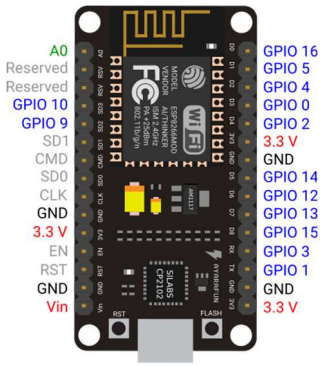


Figura 12. Pinos do NodeMCU [51].

3.4.1.2. DHT22

O DHT22 é um sensor que permite medir tanto valores de temperatura como de humidade. Permite fazer leituras de temperaturas entre -40 a +80 graus Celsius e de humidade entre 0 a 100%. Este sensor é formado por um sensor de humidade capacitivo e um termístor, ambos conectados a um controlador de 8 bits, para medir o ar ao redor, enviando para o pino de dados um sinal digital [52].

Figura 13. Pinos do DHT22 [53].

É bastante simples de usar, mas requer um *timing* cuidadoso para recolher corretamente os dados. Este sensor está capacitado para obter novos dados a cada 2 segundos [54]. Para efetuar a ligação ao NodeMCU, deve-se ligar o primeiro pino à fonte de 3-5V, o segundo pino ao seu pino de entrada de dados e o mais à direita ao *ground*. O sensor e os seus pinos estão representados na figura 13, a ligação ao NodeMCU na figura 54 e as suas características técnicas na tabela 9.

Tabela 9. Especificações técnicas do DHT22 [52].

Modelo AM2302
Tensão de operação: 3-5VDC (5,5VDC máximo)
Faixa de medição de umidade: 0 a 100% UR
Faixa de medição de temperatura: -40º a +80ºC
Corrente: 2,5mA max durante uso, em stand by de 100uA a 150 uA
Precisão de umidade de medição: ± 2,0% UR
Precisão de medição de temperatura: ± 0,5 ºC

Resolução: 0,1
Tempo de resposta: 2s
Dimensões: 25 x 15 7mm (sem terminais)

3.4.1.3. LDR

De forma a ser possível medir a intensidade da luz num certo espaço, foi usado um LDR (*light dependent resistor*), que consiste numa resistência que varia a sua resistência conforme a intensidade da luz que incide sobre ela, ou seja, quanto mais luz incidir sobre o componente, menor a resistência. Um exemplo de um LDR semelhante ao que foi usado pode ser encontrado na figura 14.

Figura 14. Exemplo de LDR [55].

Quando a luz começa a incidir sobre o material semicondutor do LDR, os fotões libertam os eletrões para a banda condutora, diminuindo a resistência do material por meio do aumento da sua condutividade [56]. A faixa em que o LDR obtém melhores leituras é na faixa da luz visível, principalmente na faixa verde-amarela. Por este motivo, o LDR é muito usado para detetar a luz do dia, sendo amplamente utilizado em fotocélulas [57]. Tal como os outros sensores, o LDR também apresenta um atraso na recolha dos dados, ou seja, não é instantâneo. Tendo em conta as diferentes variedades de LDR, pode ser sensível a diferentes faixas de luz, como o infravermelho (IR), a luz visível ou ultravioleta (UV). O LDR é frequentemente utilizado para controlar a iluminação dos postes públicos e de residências privadas.

3.4.1.4. RaspberryPi

O Raspberry Pi [58] é um *system on a chip* que executa Linux, e que fornece um conjunto de pinos GPIO (*general purpose input/output*), que permitem controlar componentes eletrónicos para explorar a Internet das Coisas. O Raspberry Pi é bastante popular entre desenvolvedores com os seus próprios projetos de domótica, visto que permite ao desenvolvedor ter o controlo em vez de utilizar um sistema proprietário fechado. Utiliza o chamado *system on a chip*, que integra o CPU e GPU num único circuito integrado, com a RAM, portas USB e outros componentes soldados na placa para um pacote tudo-em-um. Não tem armazenamento *onboard*, mas tem um *slot* para cartão SD que pode ser usado para alojar o sistema operativo e ficheiros. O Raspberry Pi é pequeno, não utiliza muita energia e é relativamente barato [59]. Neste projeto foi utilizada a versão Pi 3, pelo facto de já ter em posse uma unidade desta versão, cujas especificações podem ser encontradas na tabela 10

Tabela 10. Especificações técnicas do Raspberry Pi 3 [60].

Raspberry Pi 3	
Lançamento	29/02/2016
Tipo Chip	Broadcom BCM2837
Tipo Core	Cortex-A53 64-bit
Nº Core	4
Clock CPU	1.2 GHz
GPU	VideoCore IV
RAM	1 GB
Wireless	802.11n
Bluetooth	4.1 (BLE)
Consumo	350 mA

3.4.2. Software

3.4.2.1. Ngrok

O Ngrok [61] é uma ferramenta que permite criar túneis seguros para um servidor local, bastante útil em desenvolvimento de projetos *server side*, onde é necessária a partilha do trabalho desenvolvido com a equipa de desenvolvimento ou com um cliente. Quando o ngrok está a funcionar, fica à escuta na mesma porta em que o servidor local está a funcionar e procura pedidos externos para a máquina local. Um ponto positivo, é que permite lidar com configurações TLS por defeito, o que é bastante útil, já que permite que não seja necessário configurar manualmente certificados SSL em instâncias do servidor [62]. Esta ferramenta, além de fornecer um endereço URL HTTP, também fornece uma versão HTTPS que aponta para o *localhost*. O Ngrok também possui um painel de controlo online, que permite visualizar todos os detalhes sobre o tráfego HTTP que atravessa o túnel. Isto pode ser extremamente útil durante o desenvolvimento de aplicações. No caso deste projeto, o uso desta ferramenta foi para expor a API do servidor central à Internet, visto que, o URI de autorização do servidor de autenticação tem que ser acessível por HTTPS. Na figura 15 é possível visualizar um exemplo da saída do terminal ao executar o Ngrok.

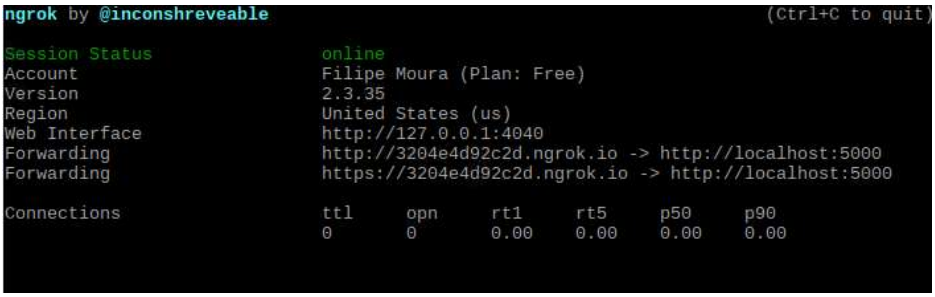


Figura 15. Saída do terminal ao executar o ngrok

3.4.2.2. Flask

O Flask [63] é uma *framework* web escrita em *Python* e baseada na biblioteca WSGI Werkzeug e na biblioteca de Jinja2, sendo um dos mais populares *frameworks* para desenvolvimento web em *Python*. O Flask tem a flexibilidade da linguagem de programação *Python* e fornece um modelo simples para desenvolvimento web. Uma vez importado para o *Python*, o Flask pode ser usado para economizar tempo construindo aplicações web. Possui um núcleo simples, visto que não tem uma camada de abstração de base de dados ou validação de formulários, mas é extensível, já que é possível usar extensões que fornecem essas funcionalidades.

Um dos principais motivos para a escolha do Flask foi a sua simplicidade. Apesar de possuir um núcleo mais simples que outras *frameworks* como por exemplo o Django, o Flask fornece todos os recursos necessários para o desenvolvimento de uma aplicação web, como o sistema de rotas, *templates* e um servidor web embutido. Caso seja necessário incrementar funcionalidades fora do *scope* do Flask, o utilizador poderá sempre recorrer a extensões externas que são desenvolvidas pela extensa comunidade. Convém também salientar que o Flask possui uma velocidade superior a outras *frameworks* como por exemplo o Django [64].

3.4.2.3. Android Studio

O Android Studio [65] é o IDE oficial para o desenvolvimento de aplicações Android, baseado no IntelliJ IDEA e incorpora as suas ferramentas de edição de código e de desenvolvimento. O software foi anunciado pela primeira vez em maio de 2013, e a primeira versão estável foi lançada em dezembro de 2014. Como forma de suporte durante o desenvolvimento das aplicações, o Android Studio utiliza um sistema de *build* baseado no *Gradle*, um emulador, *templates* de código e integração com o Github [66]. Para o desenvolvimento das aplicações, os desenvolvedores têm a opção de desenvolver em Java ou Kotlin.

3.4.2.4. Mosquitto/CloudMQTT

O Mosquitto é um broker MQTT que implementa as versões 5.0, 3.1.1 e 3.1 do protocolo MQTT. É leve e adequado para a utilização em todos os dispositivos, desde microcontroladores de baixa potência computacional até servidores mais capacitados [67]. O CloudMQTT [68] é um serviço na *cloud* onde são geridos servidores Mosquitto. O CloudMQTT proporciona métodos leves para enviar mensagens utilizando o modelo de *publish*/*subcribe*.

3.4.2.5. MongoDB Atlas

O MongoDB Atlas [69] é uma base de dados na *cloud* (*database-as-a-service*) que utiliza documentos JSON com esquemas dinâmicos e é responsável por gerir muito do trabalho manual que seria da responsabilidade do desenvolvedor, já que o modelo DBaaS permite a transferência de tarefas, tais como administração da base de dados, configuração da base de dados, *hosting* e segurança dos *clusters* dos utilizadores da base de dados para a empresa que

gere o serviço [70]. Fornece indexação e consultas avançadas e um modelo de dados flexível. O Atlas também oferece o duplo benefício da flexibilidade e escalabilidade.

3.4.2.6. AWS Lambda

O AWS Lambda é um serviço de computação *serverless/function as a service* que permite executar código sem provisionar ou gerir servidores. Executa código somente quando necessário e é dimensionado automaticamente [71]. O utilizador é apenas responsável pelo código e a função Lambda aloca de maneira automática o poder de execução de computação e executa o código com base na solicitação ou evento de entrada, para qualquer dimensão de tráfego [72].

3.5. Arquitetura

Na figura 16 está exemplificada a arquitetura final do sistema desenvolvido. É constituída por um ou mais módulos com microcontrolador, um servidor central, por *brokers* MQTT, por uma base de dados remota, por sensores e atuadores, por um website, uma aplicação móvel e por uma *skill* na Alexa. A arquitetura foi definida, de forma a que fosse fácil adicionar novos componentes ao sistema, para permitir uma maior flexibilidade e escalabilidade.

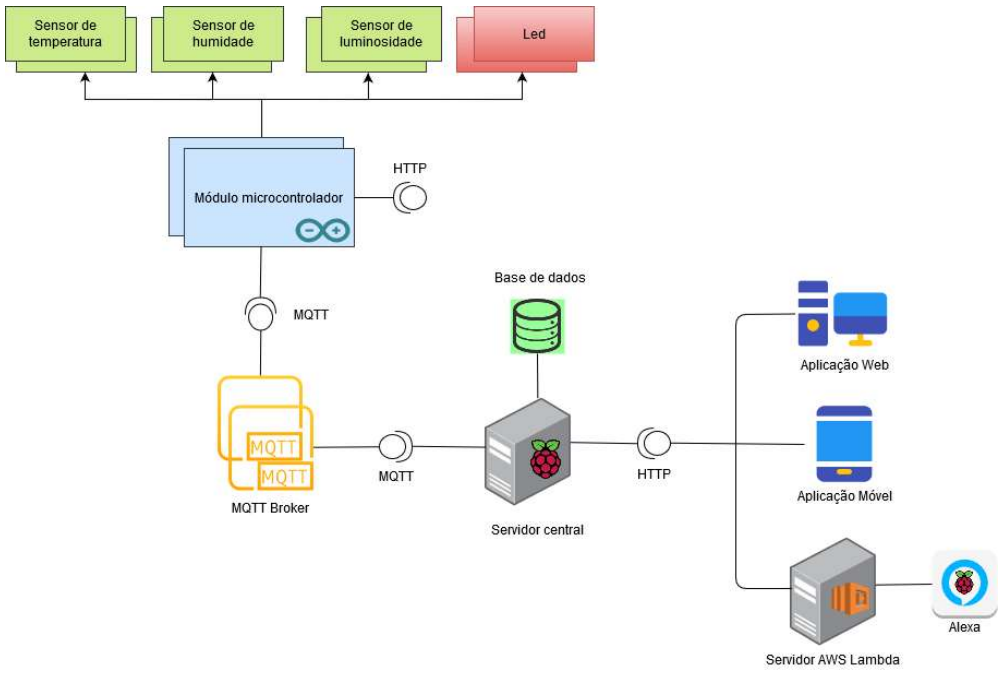


Figura 16. Arquitetura do sistema.

3.5.1. Base de dados

A base de dados selecionada foi o MongoDB [73], que permite o armazenamento de todas as leituras e ações realizadas pelo sistema. Para tal, foi usado o MongoDB Atlas [69], um serviço de base de dados na *cloud*.

A opção de usar um serviço na *cloud*, deveu-se a vários motivos. As bases de dados na *cloud* podem oferecer vantagens significativas sobre as tradicionais, incluindo maior

acessibilidade, *failover* automático e rápida recuperação automática de falhas e, potencialmente, melhor desempenho.

A segurança é também uma das suas principais vantagens, visto que, garantir a segurança dos dados é altamente crucial. A maioria dos fornecedores de bases de dados em *cloud* encriptam os dados e fornecem outras medidas de segurança. Outras vantagens são a escalabilidade e a facilidade de acesso. As bases de dados na *cloud* podem ser acedidas a partir de qualquer lugar, usando a interface web do fornecedor ou a respetiva API.

Um dos principais motivos para a escolha do MongoDB Atlas foi a escalabilidade e a natureza *Schemaless* do Mongo. O MongoDB armazena os dados em coleções e em documentos com formato em estilo JSON. Os documentos armazenados na base de dados podem ter vários conjuntos de campos com diferentes tipos para cada campo. Isto permite iterar no sistema sem ter de começar do zero para responder à evolução dos requisitos. As aplicações IoT geram um grande volume de dados, pelo que o sistema terá de escalar rapidamente.

3.5.1.1. Modelo de dados

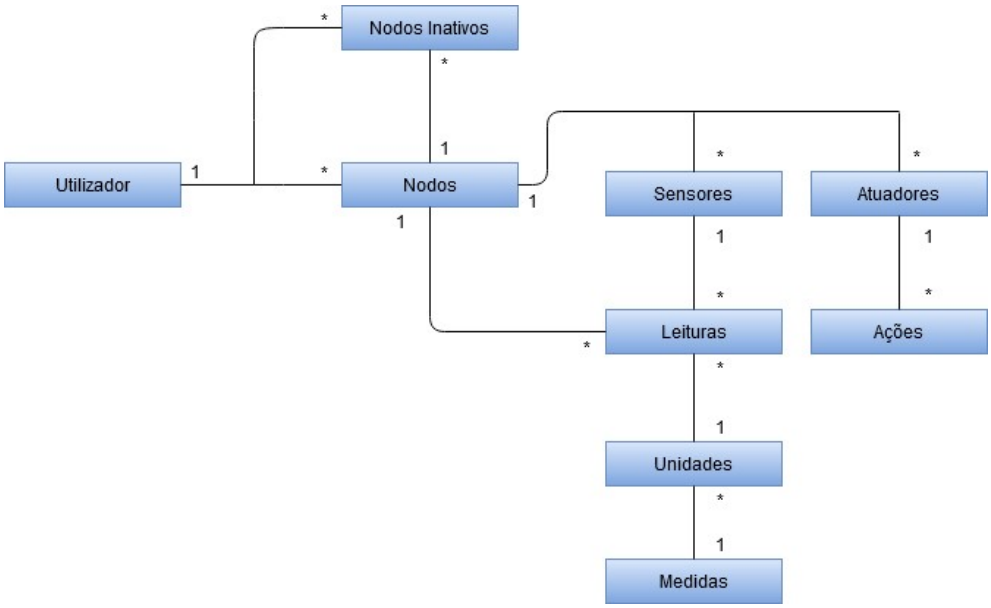


Figura 17. Modelo de dados.

Tendo em conta que a base de dados escolhida foi o MongoDB, o modelo de dados consiste em *collections* que armazenam todas as informações necessárias das várias seções do sistema. O modelo de dados foi projetado, de forma a ser o mais geral e abstrato. O modelo de dados pode ser visualizado na figura 17.

A coleção utilizador é responsável por armazenar todos os dados de uma conta de utilizador do sistema. Informações como *username*, *password* encriptada e email são armazenados. Em caso de recuperação de password, também é armazenado temporariamente um código de confirmação único. Os documentos Nodos/Nodos inativos

estão encarregues de guardar todas as informações relativas aos nodos do sistema, ou seja, os microcontroladores, que contêm um ID único e um nome que identifica a sua respetiva localização.

Todos os microcontroladores possuem sensores e atuadores. Os sensores são responsáveis por medir alterações do meio ambiente em que estão situados. No caso dos atuadores, são dispositivos capazes de reagir a comandos externos, de forma a realizar uma determinada ação. Tendo isso em conta, o modelo de dados tem as coleções sensor e atuador para armazenar os dados dos sensores e atuadores respetivamente. Atendendo a que os sensores e atuadores possuem inúmeros dados, são necessários documentos para armazenar todos esses dados. Isto é da responsabilidade da coleção leituras, que armazena todas as leituras dos sensores, e da coleção ações, que arquiva todas as ações que um determinado atuador realizou. Todas as observações de um sensor possuem uma unidade, que está armazenada na coleção unidades e a mesma faz parte de uma medida que está arquivada na coleção medidas.

3.5.2. Aplicação Web

Como é exemplificado na arquitetura, foi desenvolvida uma aplicação web de forma a que os utilizadores possam interagir com o sistema. O *backend* desta aplicação web foi definido utilizando a *framework* Flask, que é uma das *frameworks* baseadas em Python mais usadas para desenvolvimento web. O primeiro passo para o desenvolvimento da aplicação web passou pela elaboração dos protótipos de baixa fidelidade. Os protótipos de baixa fidelidade serviram de base de como a interface da aplicação deveria estar organizada. Como exemplo, na figura 18, é possível visualizar o protótipo da vista da página inicial de listagem dos nodos. A totalidade dos protótipos de baixa fidelidade encontram-se no anexo A.

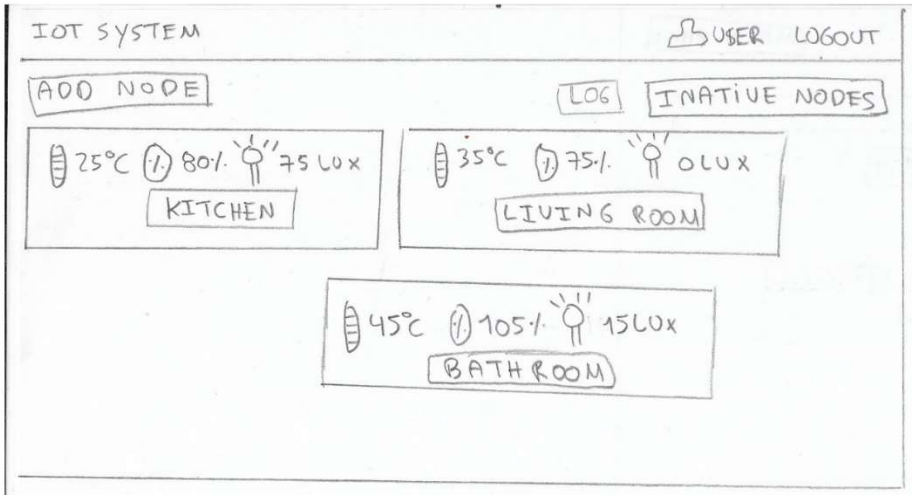
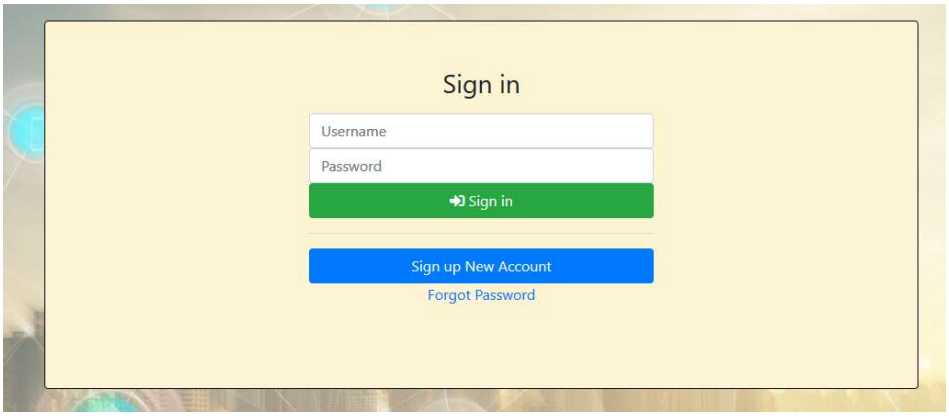
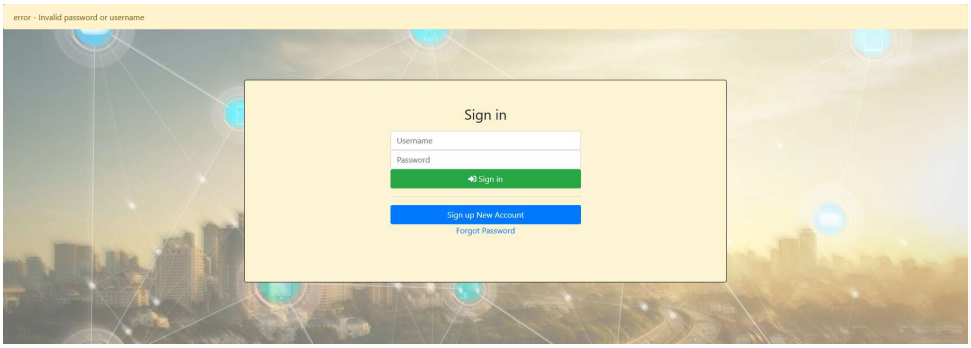


Figura 18. Vista da página inicial de listagem dos nodos da aplicação web protótipo baixa fidelidade.



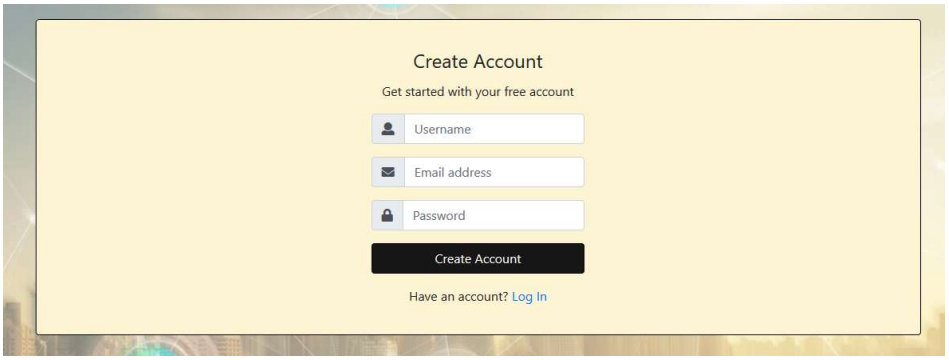
(A)



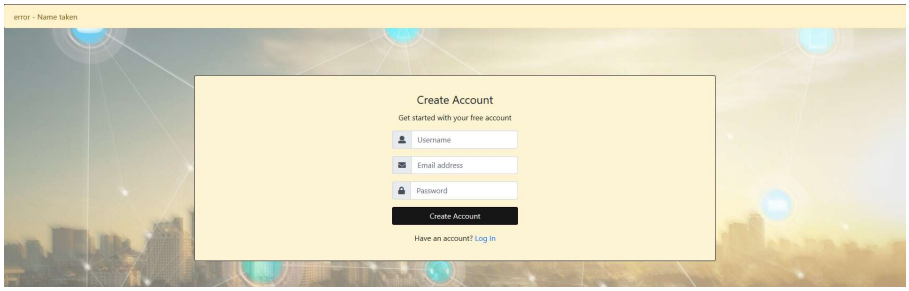
(B)

Figura 19. (A) Vista de login no website. (B) Vista de login com mensagem de erro no website.

Nas figuras 19 e 20 é possível visualizar a página de autenticação e registo do sistema. Caso o utilizador ainda não possua uma conta de utilizador, terá a oportunidade de criar uma conta. Algumas verificações são implementadas, tais como, não permitir utilizadores com o mesmo *username* e validação de *password* e *username*.



(A)



(B)

Figura 20. (A) Vista de registo no website. (B) Vista de registo com mensagem de erro no website.

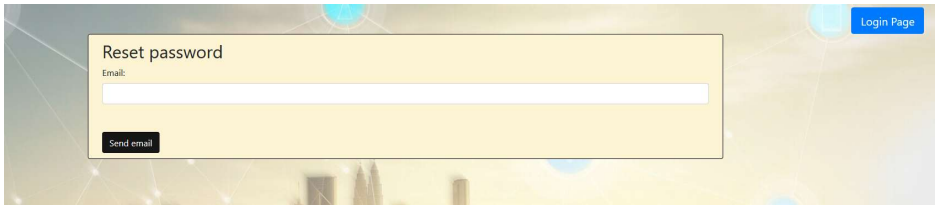
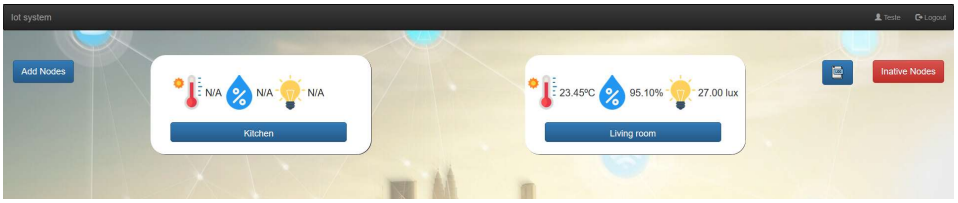
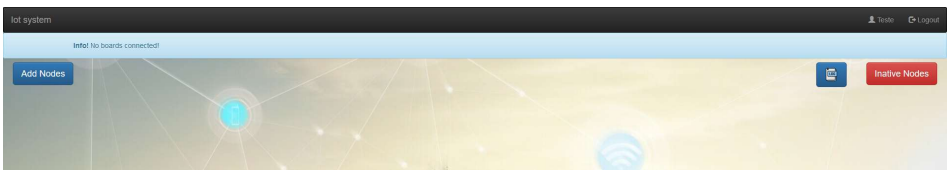


Figura 21. Vista de reset de password no website.

Na eventualidade de ter esquecido a sua *password*, o sistema oferece um mecanismo de recuperação de password, no qual o utilizador submete o seu email associado à conta. Posteriormente, recebe um email com um link para definir a sua nova password. Como medida de segurança adicional, o link enviado ao utilizador possui uma *string* codificada, que permite identificar unicamente esse utilizador. Os dados são assinados criptograficamente, para garantir que um *token* não seja violado. Outra medida adicional de segurança é o tempo de expiração para o *token* codificado, ou seja, passados 5 minutos após o envio do email, o link não é mais válido. Convém salientar que, o envio do email é feito todo de forma assíncrona. O problema, ao enviar o email de forma síncrona, é que o servidor fica bloqueado enquanto o envio se processa, o que eventualmente irá deteriorar a experiência do utilizador no caso de um envio para um servidor lento.



(A)



(B)

Figura 22. (A) Vista da página inicial com nodos listados no website. (B) Vista da página inicial sem nodos criados no website.

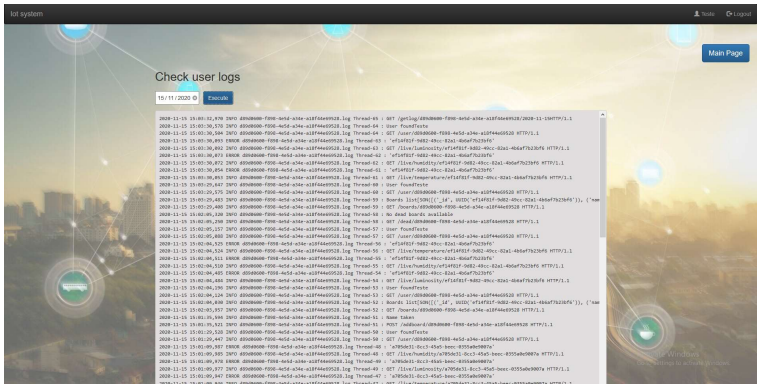
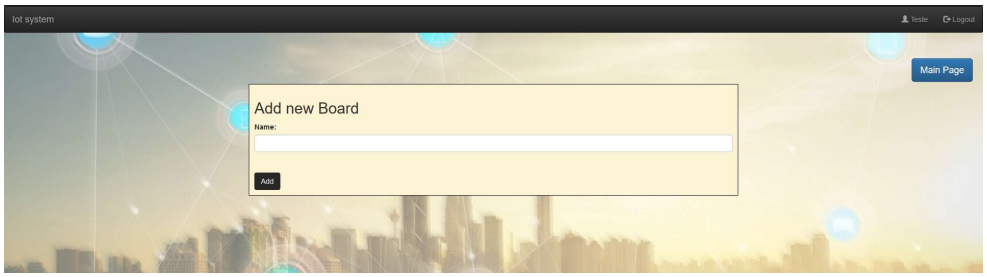
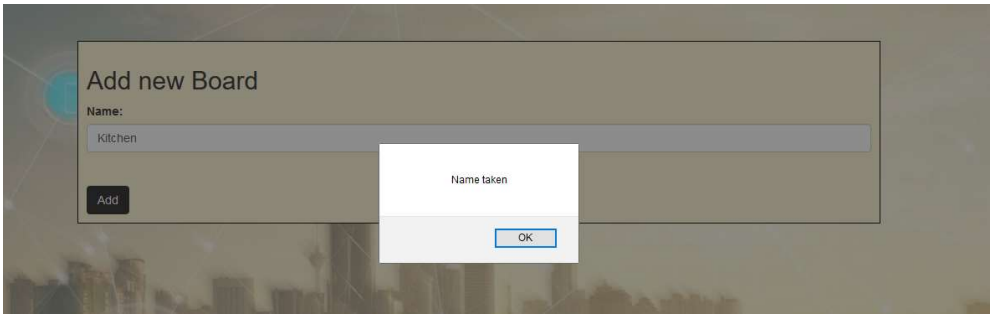


Figura 23. Vista da página de log no website.

Na figura 22 é exemplificada a página inicial do sistema após autenticação, onde são listados todos os nós ativos em tempo real. Para cada nó listado, são apresentados valores em tempo real da temperatura, humidade e luminosidade, juntamente com uma ligação (botão), que vai permitir ao utilizador controlar/monitorizar mais detalhadamente esse mesmo nó. Caso o utilizador não possua nenhum nó registado em seu nome, é apresentada da mensagem na figura 22 (B). Nessa eventualidade, o utilizador terá a possibilidade de adicionar um novo nó ao sistema ao selecionar o botão *Add Node*. Esta página também possui uma ligação à página de visualização de dados de nodos desativados, que pode ser acedida ao selecionar o botão *Inactive nodes* e uma ligação ao sistema de log, que se encontra representado na figura 23.

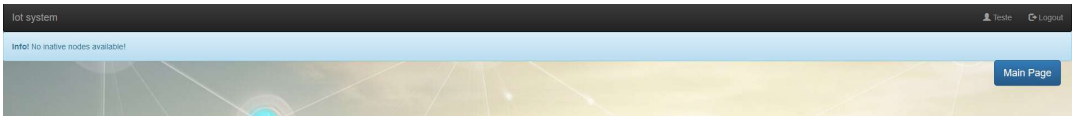


(A)



(B)

Figura 24. (A) Vista para adicionar novo nodo no website. (B) Vista para adicionar novo nodo com mensagem de erro no website.



(A)

(B)

Figura 25. (A) Vista de dados dos nós desativados sem nós no website. (B) Vista de dados dos nós desativados no website.

Na figura 24 é apresentada a página de registo de um novo nó. O utilizador, ao registar um novo nó, tem de apresentar um nome único relativo ao resto dos seus nós. No entanto, podem haver nós com nomes repetidos para diferentes utilizadores, visto que, mútiplos utilizadores podem ter nós com nomes como cozinha ou sala de estar. A página de visualização de dados de nós desativados encontra-se na figura 25. O utilizador pode, de acordo com uma data e uma medida em específico, visualizar os dados desse mesmo dia como está representado na figura 26.

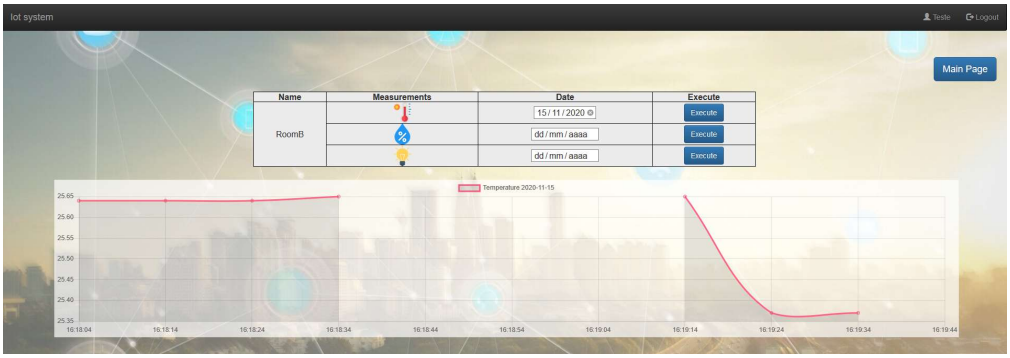


Figura 26. Vista de dados históricos de um nó desativado no website.

Caso não haja dados para esse dia, é apresentada uma mensagem ao utilizador, retratada na figura 27.

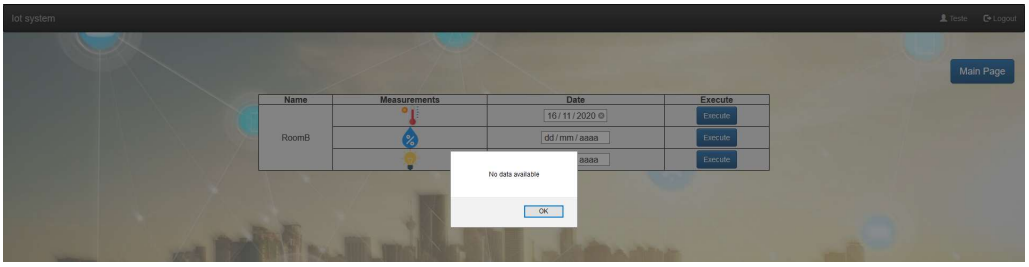
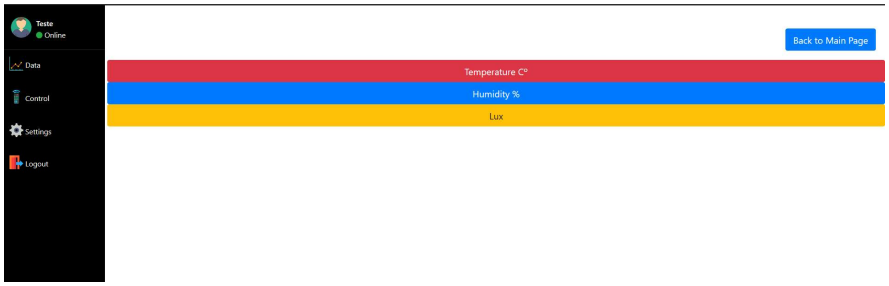
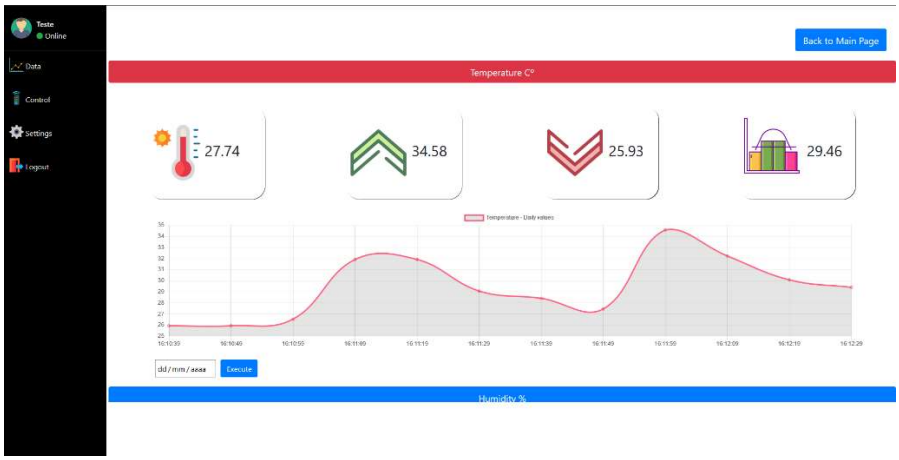


Figura 27. Vista de dados históricos de um nó desativado com notificação no website.



(A)



(B)

Figura 28. (A) Vista de dados de um nó com todas as unidades no website. (B) Vista de dados de um nó com uma unidade destacada no website.

O utilizador, ao selecionar um nó da lista, é redirecionado para a página exemplificada na figura 28. A vista *data* permite ao utilizador visualizar dados em tempo real das respectivas grandezas, oferencendo dados como o valor máximo, mínimo e a média dos últimos 12 valores registados. Tal como era possivel ao utilizador visualizar dados históricos de um nó desativado, o mesmo é permitido aqui, através de um *date picker* que pode ser utilizado para selecionar uma data em específico.

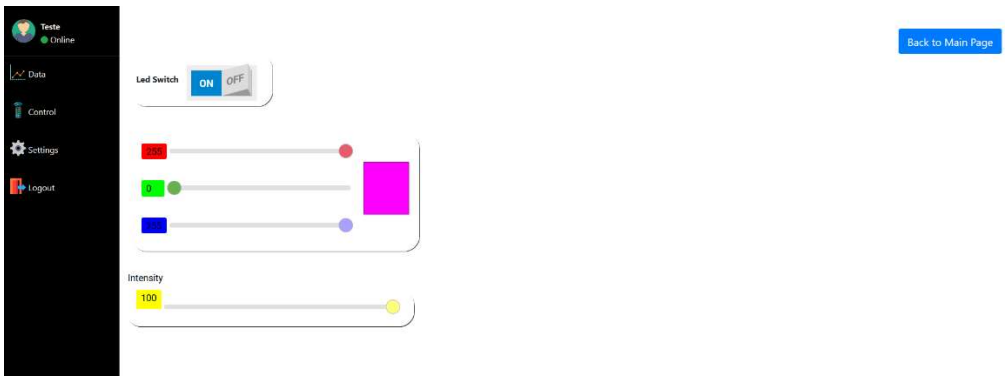


Figura 29. Vista de controlo de um nó no website.

Na figura 29, é possível verificar a vista da página *Control*, que permite ao utilizador controlar os atuadores, sendo que, de momento, o sistema tem suporte a lâmpadas LED. Todas as ações efetuadas são enviadas através de um pedido HTTP de forma assíncrona, para não bloquear a UI, onde posteriormente, o servidor central será responsável por enviar uma mensagem através do protocolo MQTT para o microcontrolador em questão. Através desta página, são possíveis ações como ligar/desligar a LED, definir a cor da LED através do código RGB (com o auxílio de um retângulo, que mostra a cor em tempo real através do valor em hexadecimal) e ajustar a intensidade.

Figura 30. Vista de definições de um nó no website.

Para finalizar, temos a vista das *definições*, onde o utilizador pode optar por mudar o nome de um nodo no sistema, com as respetivas validações, como não permitir nomes repetidos, ou apagar o nodo do sistema, onde todos os dados associados a esse nó são apagados. Ao mudar o nome de um nó, o nó assume o novo nome que corresponde a uma nova localização e o nome antigo passa a estar listado como um nó inativo. Ao apagar um nó do sistema, é enviado um pedido HTTP assíncrono para o servidor central com os dados necessários para a respetiva remoção. Posteriormente, o servidor apaga o nodo do sistema, ou seja, todos os dados associados a este nodo são removidos da base de dados na *cloud*, tais como o próprio nodo, os seus sensores, atuadores, leituras e ações. Em seguida, o servidor central envia um comando MQTT para o respetivo microcontrolador, para que este apague o seu ID e os IDs dos seus sensores e atuadores da sua memória *flash*. Existe também a possibilidade do utilizador alterar o valor predefinido da frequência (60 minutos) na qual os dados são enviados pelo módulo microcontrolador ao servidor central.

3.5.3. Aplicação Móvel

Uma aplicação móvel também foi desenvolvida, de forma a apresentar outra forma diferente de os utilizadores poderem interagir com o sistema IoT. O desenvolvimento desta aplicação foi efetuado para o sistema operativo Android, através do uso do IDE Android Studio [65] e com recurso à linguagem Java. A escolha do Java baseou-se no facto de haver um maior suporte para o desenvolvimento com esta língua.

A escolha, de usar ambientes de desenvolvimento para aplicações nativas, baseou-se nas várias vantagens que esta opção apresenta. Esta solução, em contraste com o uso de *frameworks* de desenvolvimento para aplicações multiplataforma, apresenta níveis de otimização e desempenho superiores, permitindo também um melhor nível de integração entre o software e o hardware e o uso de funções especializadas para um sistema operativo

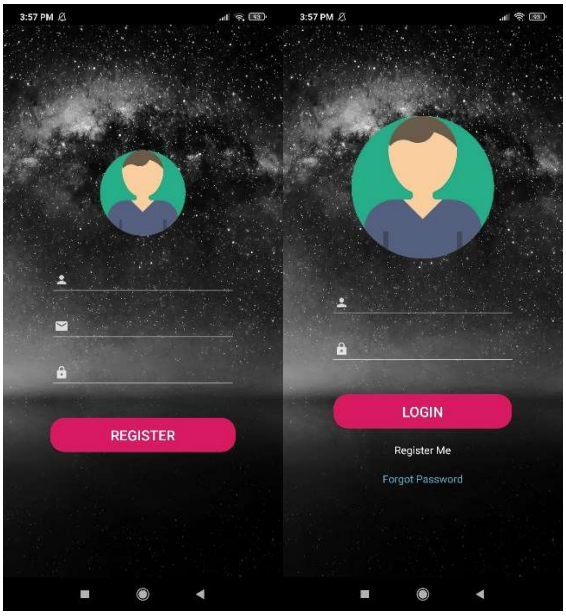
em particular. A principal motivação para esta escolha também se deveu ao facto da grande divulgação do Android, em comparação com outros sistemas, inclusive entre os utilizadores que eventualmente testaram o sistema, que possuíam todos dispositivos Android.

No entanto, convém salientar que o uso de *frameworks* para aplicações multiplataforma, também apresenta as suas vantagens, como por exemplo o facto do código desenvolvido ser executado em vários sistemas operativos, o que permite uma melhor reutilização de código entre as várias plataformas, diminuindo assim o tempo de atualização da plataforma. Contudo, esta solução também apresenta desvantagens, como anteriormente referido, a usabilidade e desempenho são inferiores às aplicações nativas e o tempo de desenvolvimento é superior, pelo facto de ser necessário ter conhecimento das diferenças entre as várias plataformas.

Tal como no caso da aplicação web, o primeiro passo no desenvolvimento da aplicação móvel foi a elaboração dos protótipos de baixa fidelidade. Na figura 31 é possível visualizar um exemplo, da vista da listagem de nós. O resto dos protótipos estão situados no anexo A.

Figura 31. Vista da listagem de na aplicação móvel protótipo de baixa fidelidade

Cada uma das vistas da aplicação móvel tem as mesmas funcionalidades que as da aplicação web. Abaixo encontra-se a descrição de cada uma das vistas.



(A) (B)

Figura 32. (A) Vista de registo da aplicação móvel. (B) Vista de login da aplicação móvel.

Nas figuras 32 e 33 encontram-se as vistas de registo, login da aplicação móvel e *reset* da *password*. Os dados inseridos pelo utilizador são enviados através de um pedido HTTP assíncrono. Na eventualidade de o utilizador não inserir os dados ou, caso os dados inseridos estejam incorretos, é usado um *Toast,* de forma a fornecer *feedback* ao utilizador. Uma notificação toast fornece um feedback simples sobre uma operação numa pequena janela pop-up. Ela ocupa somente a quantidade de espaço necessária para a mensagem, e a atividade atual continua visível e interativa. As notificações toast desaparecem automaticamente após um tempo limite.

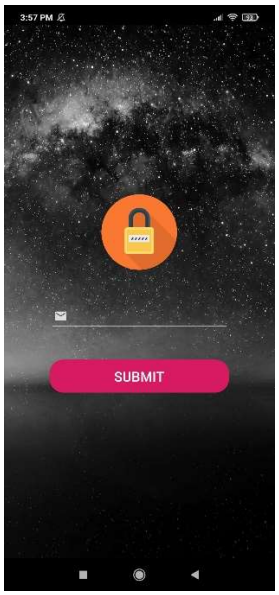
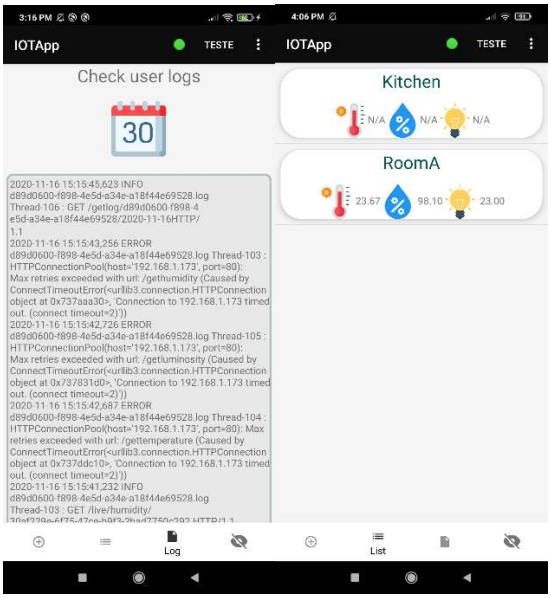


Figura 33. Vista de reset da password na aplicação móvel.



(A) (B)

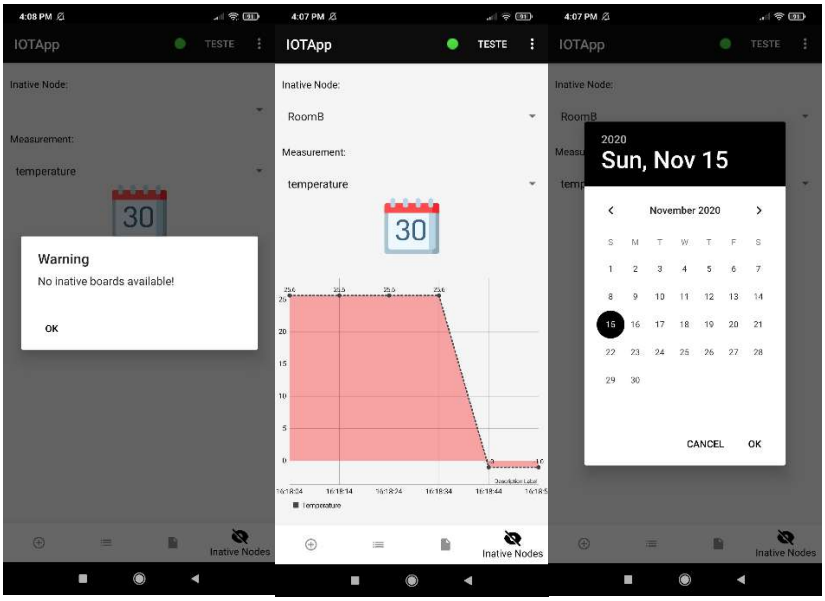
Figura 34.(A) Vista da página de log na aplicação móvel. (B) Vista de listagem dos nodos na aplicação móvel.

Na figura 34A, tal como na aplicação web, o utilizador pode visualizar os seus respetivos *logs* para um dia específico. Na figura 34B, encontra-se representada a listagem de todos os nós. Para a implementação desta funcionalidade no sistema Android, foi usado um *recyler view* juntamente com um *adapter*, sendo esta a melhor opção a implementar sempre que uma aplicação tem a necessidade de apresentar um volume de dados dinâmicos. O *adapter* tem o papel de alimentar todos os seus dados para a lista, ou seja, ele é responsável por criar visualizações para itens e substituir o conteúdo de algumas das visualizações por novos itens de dados, quando o item original não está mais visível. Isto permitiu a listagem dos nós e a possibilidade de o utilizador selecionar um nó para controlo ou visualização de dados em concreto. Para a sua implementação, foi necessário efetuar primeiro um pedido HTTP assíncrono para obter a lista de todos os nodos que pertencem ao utilizador autenticado. Ao obter estes dados, a aplicação cria a *recyler view* e define o respetivo adapter. A criação da recyclerView e do respectivo adapter pode ser visualizado na figura 35.

```
runOnUiThread(new Runnable() {
    @Override
    public void run() {
        adapter = new BoardAdapter(getApplicationContext(),boards, activity: ListActivity.this);
        recyclerView.addItemDecoration(new DividerItemDecoration(recyclerView.getContext(), DividerItemDecoration.VERTICAL));
        recyclerView.setAdapter(adapter);
    }
});
```

Figura 35. Criação da recyclerView e do respectivo adapter.

Posteriormente, caso o utilizador selecione um nó em específico, é da responsabilidade do *adapter* iniciar a actividade seguinte, onde todos os dados necessários como por exemplo o *access token* são transmitidos.



(A) (B) (C)

Figura 36. (A) Vista de dados dos nós desativados sem nós na aplicação móvel. (B) Vista de dados históricos de um nó desativado na aplicação móvel. (C) Vista de dados históricos de um nó desativado com date dialog na aplicação móvel.

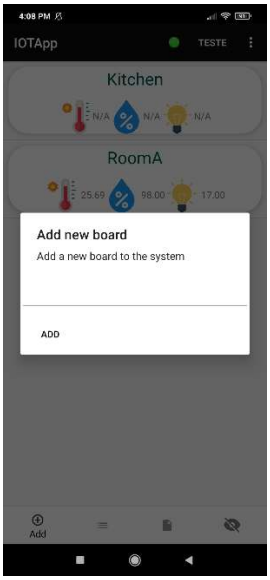
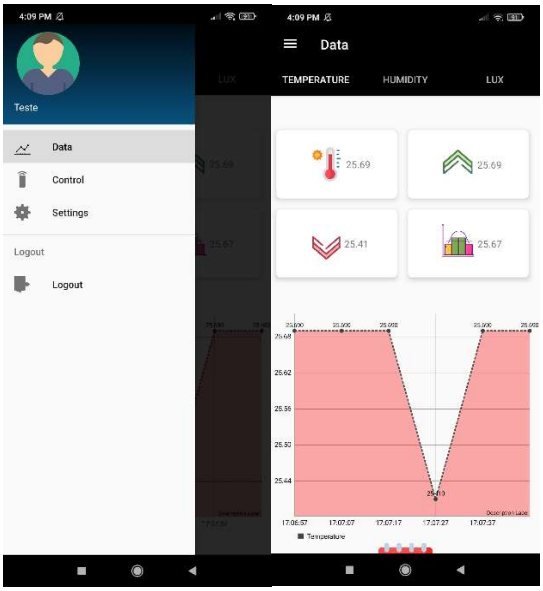


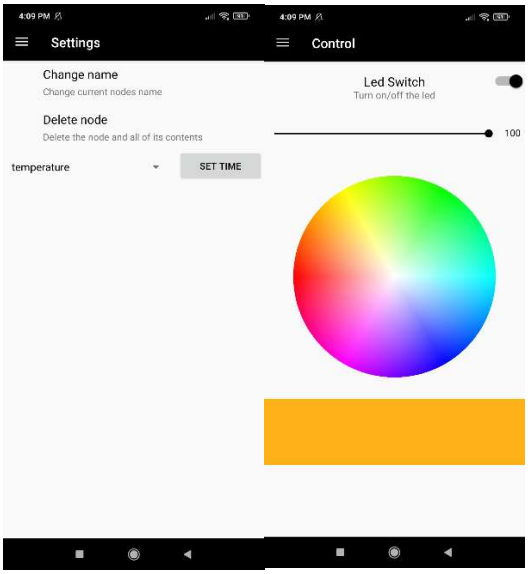
Figura 37. Vista para adicionar novo nodo na aplicação móvel.

Tal como na aplicação web, na figura 36 temos representadas as vistas onde o utilizador pode visualizar os dados de diferentes medidas em diferentes dias, de um nó desativado, recorrendo a um *date dialog*. Já na figura 37, o utilizador pode, através do *dialog,* inserir o nome de um novo nodo no sistema. Para a criação dos gráficos foi usada a biblioteca MPAndroidChart que tem suporte para uma vasta gama de gráficos.



(A) (B)

Figura 38. (A) Vista de dados de um nó com sidebar na aplicação móvel. (B) Vista de dados de um nó na aplicação móvel.



(A) (B)

Figura 39. (A) Vista de settings na aplicação móvel. (B) Vista de controlo na aplicação móvel.

Nas figuras 38 e 39 estão representadas as vistas após o utilizador selecionar um nó, que possuem as mesmas funcionalidades que a aplicação web. Para navegar entre as respetivas vistas, foi implementada uma *navigationView*, que está associada a cada um dos fragmentos implementados (*data*, *control* e *settings*), o que permite navegar entre os diferentes *items* da *sidebar*.

No fragmento dos dados, foi implementada uma *TabLayout*, que permite ao utilizador navegar entre as diferentes medidas. Na vista do controlo, inicialmente, são efetuados pedidos HTTP ao servidor central, de forma a averiguar

qual o estado de cada um dos parâmetros representados na vista, ou seja, qual o estado da LED ou qual a cor que a LED apresenta no momento. Para tal, são efetuados os pedidos assíncronos ao servidor central, que por sua vez realiza o pedido para a API do respetivo microcontrolador, sendo que, ao obter a resposta, retorna de volta à aplicação.

3.5.4. Alexa Skill

A arquitetura do sistema foi definida de forma a que múltiplas aplicações clientes pudessem servir-se dos serviços do servidor central. Desta forma, foi criada também uma *skill* na Alexa. A Alexa Skill foi definida de forma a apresentar mais uma forma distinta dos utilizadores interagirem com o sistema, diferente das tradicionais, como as aplicações móveis/web. A escolha da Alexa deveu-se ao facto de que os outros assistentes virtuais, ao contrário da Alexa, funcionam em dispositivos exclusivos, à sua grande vantagem em estender as suas funcionalidades bases através da criação de *skills* e ainda, devido à qualidade da documentação apresentada pela Amazon. Com a criação desta *skill* pretende-se averiguar se esta forma distinta de interação é a mais adequada para um sistema IoT e se problemas identificados em outros projetos [12][13] realmente se comprovam, tais como, os utilizadores terem dificuldade em efetuar pedidos, devido à falta de pistas visuais, terem de lembrar-se dos comandos, não saberem como pronunciar os pedidos ou haver falhas de pronúncia de palavras que levem a falhas de interpretação por parte do assistente.

3.5.4.1. Estrutura da Alexa

A Alexa é o serviço de voz da Amazon, baseada na *cloud,* que alimenta a família Echo bem como a aplicação nos smartphones Android e iOS. Para a criação da *skill,* ou seja, funcionalidades extra que permitem estender as funcionalidades base da Alexa*,* foi necessário criar duas contas nos serviços da Amazon: uma na *Amazon developer services* [74], que permite definir o modelo de interação e outra no *Amazon Web Services* (AWS) [75], que vai ser responsável por hospedar o *backend* da *skill*.

O ecossistema da Alexa possui duas estruturas importantes que permitem facilitar o desenvolvimento de novas *skills* e fazer a conexão aos serviços da Alexa em qualquer lugar: o Alexa Skills Kit (AKS) [76] e Alexa Voice Service (AVS) [77]. O ASK é uma coleção de APIs e ferramentas, que trata do trabalho árduo relacionado com as interfaces de voz, incluindo o reconhecimento da fala, a codificação de texto para fala e o processamento de linguagem natural. O ASK ajuda os programadores a construir *skills* de forma rápida e fácil [78]. O AVS é uma interface de programação (API), que permite aos programadores adicionar a Alexa da Amazon e o reconhecimento da linguagem natural às suas aplicações e produtos. Introduzido em 2016 como um serviço gratuito baseado na nuvem, o Alexa Voice Service (AVS) permite aos programadores adicionar reconhecimento verbal sofisticado aos seus produtos [79].

Figura 40. Ecossistema da Alexa [80].

Uma *skill* da Alexa consiste em dois componentes principais: a interface da *skill* e o serviço da *skill*.

A interface da *skill* processa o pedido de voz do utilizador e depois mapeia-o para a intenção no modelo de interação. As intenções correspondem a ações para satisfazer o pedido verbal do utilizador. Cada intenção tem pelo menos uma *utterance*, uma palavra ou frase predefinida, e o utilizador pode dizer esta frase para invocar a intenção. Se for detetada uma intenção específica, a interface da *skill* cria um evento codificado em JSON, que será passado para o serviço da *skill*.

O serviço da *skill* determina quais as ações a tomar, em resposta ao evento codificado em JSON recebido da interface da *skill*. Ao tomar uma decisão, o serviço da *skill* devolve uma resposta codificada em JSON à interface da *skill* para processamento posterior. Após o processamento, a resposta em áudio é reenviada ao utilizador através do Echo ou de outro produto com AVS [81].

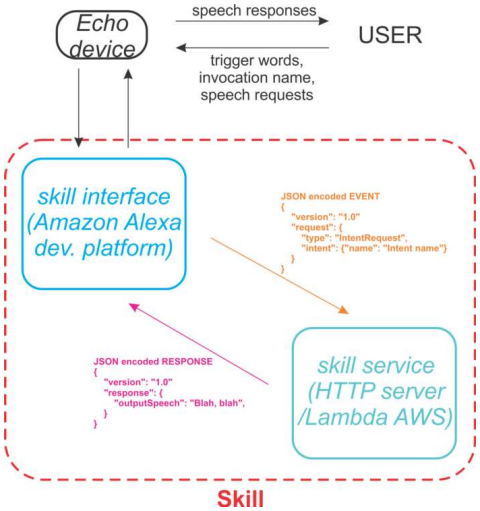


Figura 41. Comunicação utilizador-skill [82].

O modelo de interação é uma ferramenta para treinar a interface da *skill,* para que esta saiba como interpretar o que o utilizador diz. Converte a fala em eventos de intenção específicos. Ao fornecer uma lista de exemplos de *utterances*, é possível definir as palavras que devem ser mapeadas para nomes de intenções específicas no modelo de interação. As *utterances*, são uma lista de palavras que representam a forma como o utilizador pode falar com a *skill*. Este mapeamento forma o modelo de interação para a *skill*. Estas afirmações são utilizadas para gerar modelos de compreensão da linguagem natural, o que permite aos utilizadores exprimir a intenção da *skill*.

Também é possível declarar um esquema de intenções sobre o modelo de interação. Um esquema de intenções é uma estrutura JSON do modelo de interação que declara o conjunto de intenções que um serviço pode aceitar e processar. O esquema de intenções diz à interface da *skill* o que é que o serviço da *skill* implementa. Uma vez fornecidas as *utterances*, a interface da *skill* pode traduzir as palavras faladas do utilizador para os eventos específicos que o serviço da *skill* pode tratar [81].

3.5.4.2. Modelo de interação

Para a criação da *skill* foi necessário aceder à consola de desenvolvimento da Alexa. Primeiro, é requerido o nome da *skill* e o idioma, juntamente com o tipo de modelo a usar para o modelo de interação da *skill*, que pode ser um dos 4 tipos a seguir apresentados:

- Custom model (modelo de interação definido pelo utilizador; modelo usado neste projeto);
- Flash briefing model (modelo de interação pré-definido para *feeds* de notícias);
- Smart home model (modelo de interação pré-definido para aplicações domésticas inteligentes);
- Video model (modelo de interação pré-definido para aplicações vídeo).

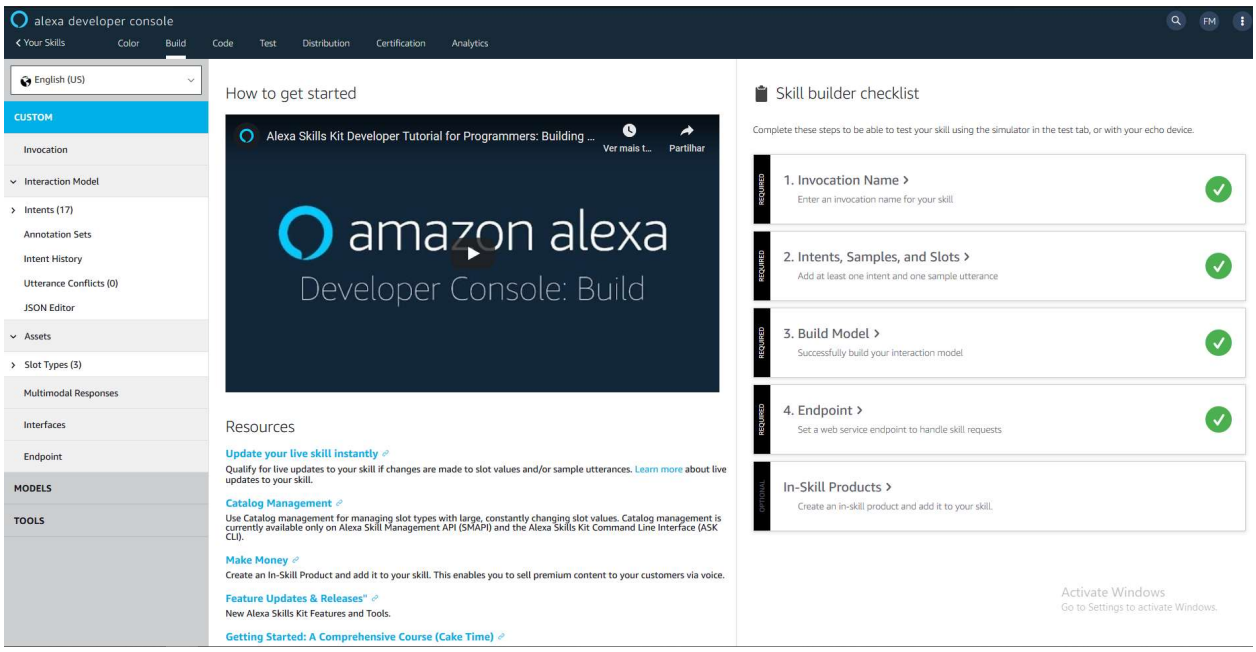


Figura 42. Alexa developer console [83].

Ao selecionar a aba *Build,* é apresentada a página da figura 42, onde o utilizador pode especificar o nome de invocação da *skill*. O nome de invocação é o que vai permitir a Alexa saber quando começar a interação com o utilizador para uma determinada *skill*, ou seja*,* sempre que o utilizador querer efetuar um determinado pedido suportado pela *skill* terá de pronunciar primeiro o nome de invocação seguido do pedido. Os requerimentos para o nome de invocação são os seguintes:

- Usar um nome de invocação com duas ou mais palavras;
- Separar as palavras com espaços;
- Usar apenas letras minúsculas;
- Colocar o nome da invocação entre aspas se usar um apóstrofo ou uma abreviatura com um ponto;
- Escrever por extenso números ou outros caracteres especiais.

Posteriormente, foi necessário criar as intenções e respetivas *utterances*, de forma a que o utilizador pudesse invocar uma intenção, expressando-se de forma diferente. Muitas das intenções utilizadas possuem parâmetros dinâmicos, tendo em conta as palavras que o utilizador menciona. Por este motivo, foram usados *slots built-in* na definição das intenções, fornecidos pelo ASK, como por exemplo AMAZON.NUMBER que converte palavras numéricas ("cinco") em dígitos (tais como "5") [84]. Caso nenhum dos tipos de *slots built* in se enquadre, o utilizador pode definir tipos de *slot* customizados. Quando se utiliza um *slot* customizado, define-se o tipo e os seus valores, e especifica-se o nome do tipo como parte da definição da intenção [85].

Para finalizar, foi necessário definir os *endpoints* do serviço web, o que corresponde ao serviço da *skill*, que é responsável por tratar os pedidos orais dos utilizadores. Há duas opções à escolha: o serviço que tratar os pedidos pode ser executado como uma função Lambda na plataforma de computação AWS ou através de HTTPS num servidor web. Caso fosse escolhida a segunda opção, seria necessário um servidor web que satisfizesse os seguintes requisitos:

- Ligação à Internet;
- HTTPS através de um certificado SSL/TLS reconhecido pela Amazon;
- A porta 443 estar disponível para consultas.

No caso deste projeto, foi usada a primeira opção. Para tal, o *endpoint* definido teve de ser o ARN da função Lambda.

3.5.4.3. *Backend da skill*

Para o desenvolvimento do *backend* da *skill,* foram usados os serviços da AWS, mais concretamente o serviço AWS Lambda, que será responsável por receber os pedidos da interface da *skill* e por efetuar pedidos à API do servidor central. Foi usado o SDK do NodeJS, que é executado no servidor Lambda. Para criar a função Lambda, foi usada a *dashboard* do AWS e selecionado “Create function”. Posteriormente são apresentadas três opções:

• author from scratch: a função lambda é criada sem nenhum modelo, mas com o exemplo “Hello world”. Esta foi a opção escolhida para este projeto;

• blueprint: a função é criada a partir de um *template* predefinido;

• browse serverless app repository: a função é criada baseando-se numa aplicação desenvolvida por outros desenvolvedores, a partir do repositório *AWS Serverless Application Repository.*

Para finalizar, é escolhido o nome da função, o ambiente de execução (que neste caso é NodeJS) e as *roles* (permissões). Após a criação da função, foi necessário definir o *trigger*, que é o modelo de interação configurado na consola de desenvolvimento Alexa. Para tal, foi selecionado o Alexa Skills Kit na secção add triggers. e no campo "Skill ID" foi passado o ID da *skill*, que se encontra na seção *endpoints* da dashboard da Alexa Developer Console.

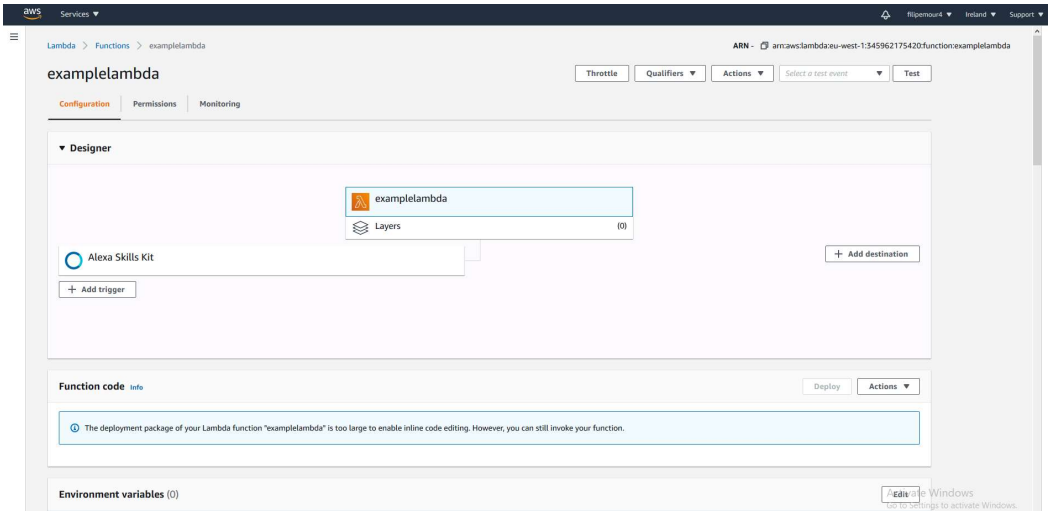
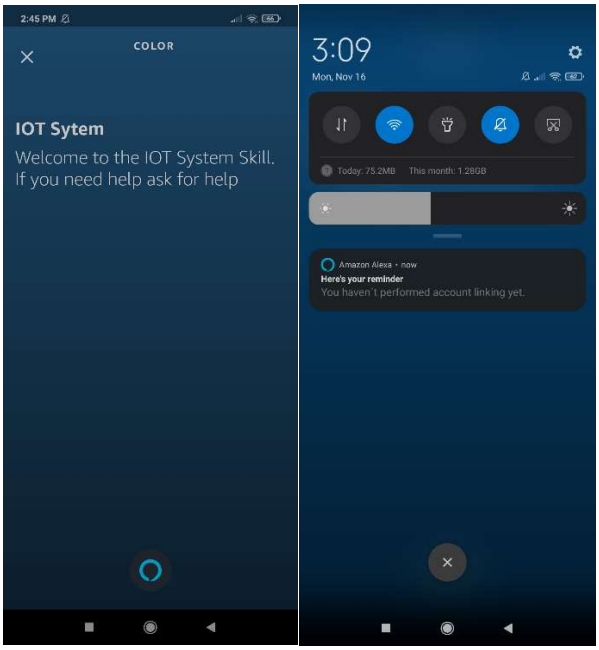


Figura 43. Página de controlo da função lambda [86].

3.5.4.4. *Account linking*

O utilizador pode iniciar a interação com a *skill,* usando a palavra de invocação *home automation*, que permite à *skill* apresentar uma mensagem de abertura e, na eventualidade do utilizador não ter efetuado o *link* entre a conta da Amazon e a conta do sistema, é efetuada uma *push notification* que notifica o utilizador que a tal ligação ainda não foi efetuada.



(A) (B)

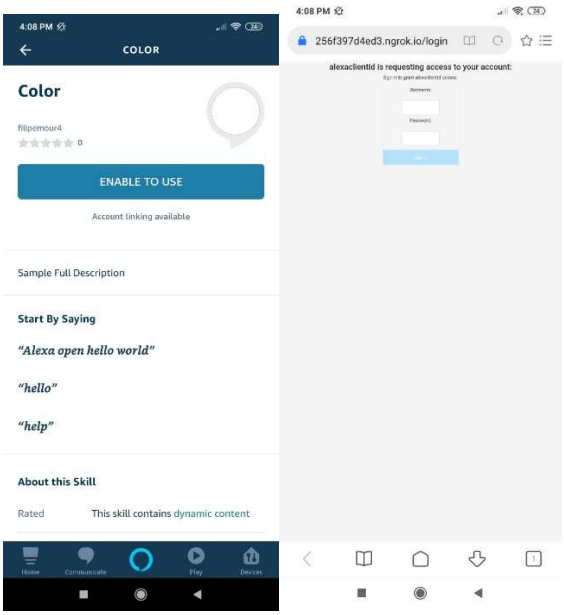
Figura 44. (A) Vista inicial do skill na aplicação da Alexa. (B) Push notification a informar que o account linking ainda não foi efetuado

Account linking permite que a *skill* possa conectar a identidade do utilizador na Amazon com a identidade do utilizador num sistema remoto. Isto é necessário, visto que todas as rotas da API do servidor central estão protegidas, sendo essencial um *access token* que identifique unicamente o utilizador a efetuar o pedido. Para criar um *link* entre a conta Amazon do utilizador da Alexa e a conta que eles possuem num serviço externo, o *Alexa Skills Kit* usa a estrutura de autenticação OAuth 2.0. O OAuth 2.0 define um meio pelo qual um serviço, com a autorização do utilizador, aceda a informações da conta que o utilizador possui no serviço externo.

No caso da *Alexa Skill Kit* são suportados dois tipos de autorização OAuth: o *Implicit Grant Type* e o *Authorization Code Grant Type*. A principal diferença entre esses dois tipos está na maneira como o *token* de acesso é obtido do servidor de autorização. No *Implicit Grant Type*, o servidor de autenticação retorna o *access token* após o login do utilizador durante o processo de ligação da conta. No caso do *Authorization Code Grant Type*, o servidor de autenticação retorna um código de autorização depois do utilizador efetuar o login durante o processo de ligação da conta. A Alexa usa então esse código, para solicitar um par de *access token / refresh token* do servidor de autorização. A Alexa pode usar o *refresh token* para solicitar um novo par de *access tokens* quando o *token* expirar. No caso desta *skill* foi usado o *Authorization Code Grant Type*, visto ser mais seguro, permitir o uso de *refresh tokens*, o que não é possível no *Implicit Grant Type* e pelo facto de ser o recomendado pela Amazon.

Para o processo de *account linking* ser possível, é exigido que alguns requisitos sejam cumpridos, como por exemplo, o servidor tem de ter suporte a OAuth 2.0, o uso de HTTPS, ou seja, o URI de autorização e de *access tokens* têm de ser encriptados e têm que estar

expostos na Internet. Para tal, recorreu-se ao uso da ferramenta ngrok, que permitiu expor o servidor local. Na figura 45, está representada as vistas para o processo de *account link* a partir da aplicação da Alexa. Na imagem 46, o utilizador é redirecionado de volta para a aplicação juntamente com uma mensagem de sucesso.



(A) (B)

Figura 45. (A) Página da skill na aplicação da Alexa. (B) Página para autenticar-se no sistema através da aplicação da Alexa.

Figura 46. Mensagem de sucesso ao efetuar account linking.

Na figura 47, é representado o processo de autorização que foi implementado para esta *skill*, ou seja, o *Authorization Code Grant Type*. Inicialmente, o servidor identifica e autêntica a identidade do utilizador da Alexa com um utilizador no sistema. Ele desempenha um papel fundamental na vinculação de contas: 1) exibe uma página de login para o utilizador efetuar o login no sistema, 2) autentica o utilizador no sistema, 3) gera um código de

autorização que identifica o utilizador e 4) passa o código de autorização para a aplicação da Alexa e, finalmente, 5) aceita o código de autorização do serviço Alexa e retorna um *access token* com validade que o serviço Alexa pode usar para aceder aos dados do utilizador noutro sistema.

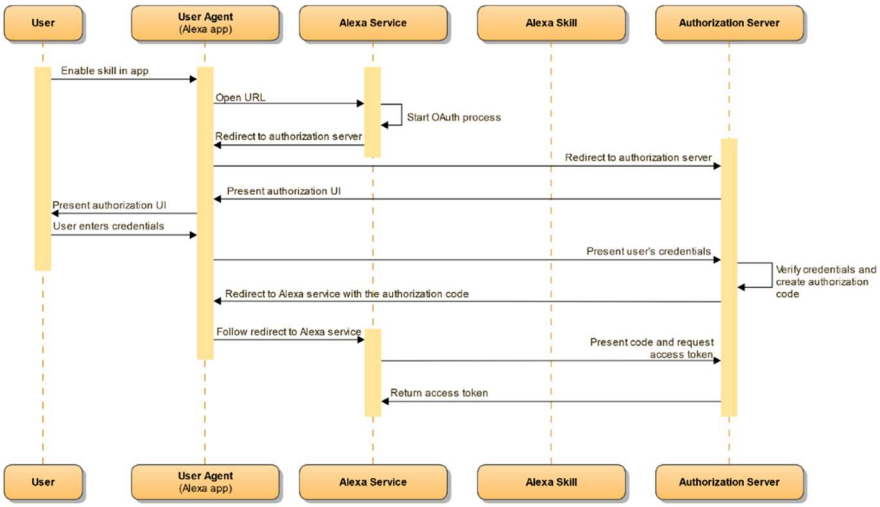


Figura 47. Diagrama do processo de account linking [87].

3.5.4.5. Home automation skill

Na figura 48, está representado um diagrama que mostra o diagrama de sequências quando o utilizador faz um pedido para a *skill* e a *skill* usa o *access token* para aceder a informações do *resource server*. A skill suporta, tanto pedidos de consulta como pedidos de atuação. Foram desenvolvidos *intents*, que representam ações que a *skill* pode executar a partir dos pedidos orais dos utilizadores. Para a interação com a skill, os utilizadores podem usar a aplicação móvel da Alexa ou o Raspberry, que têm a Alexa integrada. Uma imagem do produto com AVS pode ser encontrada no anexo C.

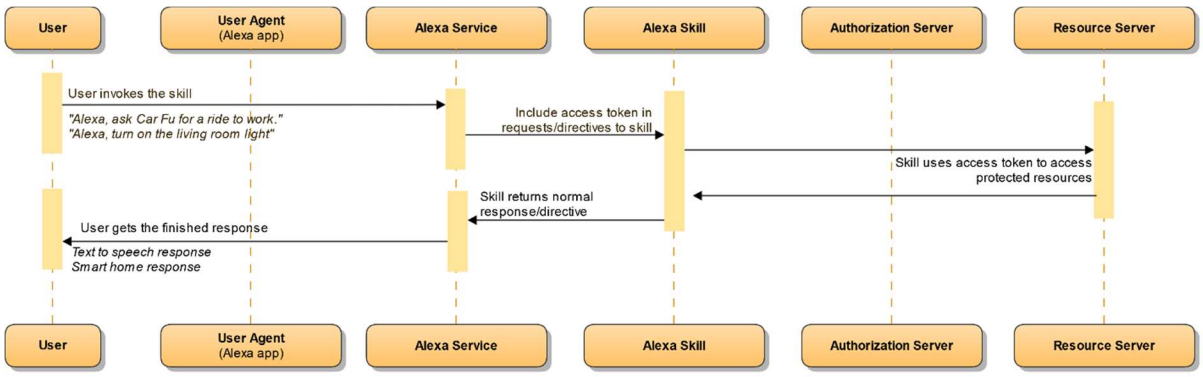
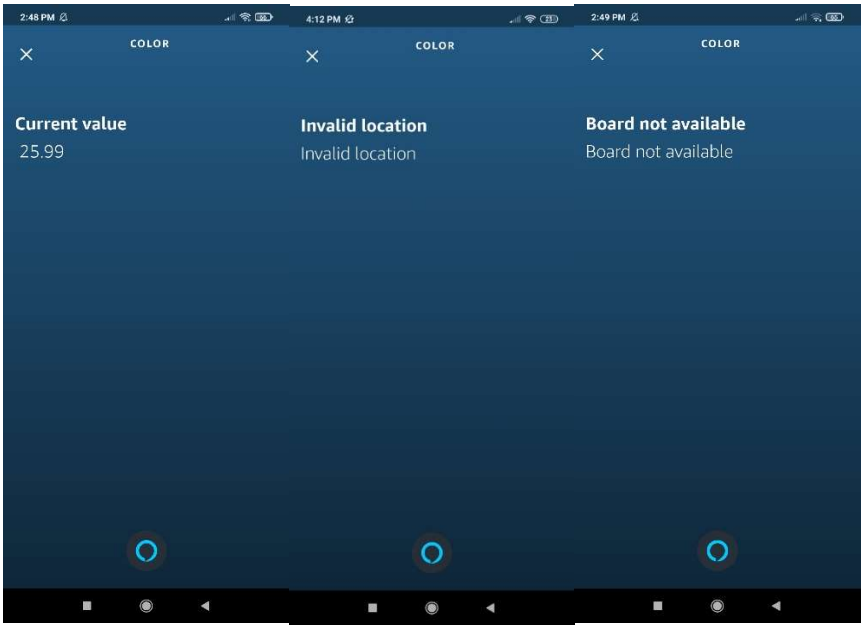


Figura 48. Diagrama de pedido do utilizador com uso do token [87].



(A) (B) (C)

Figura 49. (A) Vista de retorno do output por parte da Alexa. (B). Vista de retorno de mensagem de localização inválida por parte da Alexa. (C) Vista de retorno de mensagem de nó não ativo por parte da Alexa.

Na figura 49, é possível visualizar a resposta retornada pela Alexa no formato de *card,* através da aplicação da Alexa. Na figura 49 (A), encontra-se o output devolvido no caso do pedido ter sido executado com sucesso. Na figura 49 (B), está representada a situação em que um utilizador efetua um pedido para uma localização que não existe no seu meio, ou seja, o sistema efetua um pedido HTTP assíncrono, de forma a averiguar se o nodo mencionado pelo utilizador existe. Esta verificação é efetuada do lado do servidor central, recorrendo ao nome enviado no pedido e à identidade do utilizador enviada no *access token*. Caso esse nodo não exista, o utilizador é informado da situação e é lhe solicitado para repetir o pedido. Na figura 49 (C), corresponde à situação em que o utilizador efetua um pedido para um nodo que não se encontra ativo de momento, ou seja, está desligado ou incomunicável.

Tabela 11. Lista de intents.

TIPO	***Intent***
Consulta	LiveIntent
Consulta	MaxIntent
Consulta	MinIntent
Consulta	AvgIntent
Consulta	GetLedSwitchIntent
Consulta	GetIntensityIntent
Ação	SetSwitchIntent

Ação	SetColorIntent
Ação	SetIntensityIntent
Ação	SetNewNameIntent
Ação	SetFreqIntent
Ação	DeleteIntent
Ação	SetNewBoardIntent
Built in	CancelIntent, HelpIntent,StopIntent,NavigageHomeIntent

Na tabela 11, estão representados todos os *intents* desenvolvidos, sendo que o utilizador pode efetuar pedidos de consulta, de ação e 4 são *built in*. O fluxo de execução entre os pedidos de consulta e ação são bastante semelhantes. No caso de um pedido de consulta "*Whats the current temperature in the kitchen*", sendo que *kitchen* representa um nodo, este inicia-se com o utilizador a pronunciá-lo e com a captura por parte da Alexa. De seguida, o servidor *lambda* é responsável por efetuar o pedido HTTP assíncrono para o servidor central, com o *access token* no pedido. Este, por sua vez vai efetuar um pedido HTTP para o *endpoint* correspondente da API do microcontrolador adequado. O microcontrolador consulta o último valor registado da temperatura e reenvia a resposta ao servidor central, que por sua vez reenvia-a para o servidor *lamdba* e consequentemente para o utilizador.

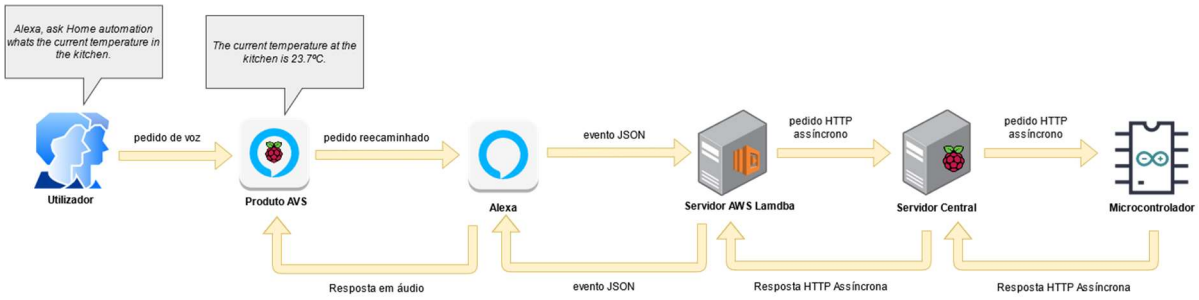


Figura 50. Fluxo de execução de um pedido de consulta na Alexa.

No caso de uma ação o fluxo é bastante semelhante. Tomando por exemplo o comando "*Change the name of kitchen to living room*", inicialmente, tal como no caso anterior, o servidor *lambda* efetua o pedido HTTP assíncrono para a API do servidor central, que por sua vez é responsável por enviar uma mensagem MQTT com o valor do novo nome para o tópico "newname/"+id". O microcontrolador dispara o *callback* para uma mensagem MQTT recebida e efetua a respetiva mudança de nome e procede, para guardar o novo nome na sua memoria flash.

3.5.5. Broker MQTT

Para a troca de mensagens entre o servidor central e os microcontroladores foram usados *brokers* MQTT. O *broker* é um intermediário no processo de comunicação entre os sensores (*publisher*) e os diversos clientes (*subscriber*). O *broker* é o elemento responsável por gerir as publicações e as subscrições do protocolo MQTT. No caso deste sistema, foram usadas duas abordagens distintas, um *broker* na *cloud* e um *broker* a ser executado localmente num Raspberry Pi. Inicialmente, o servidor central, que se encontra a ser executado no Raspberry, e os microcontroladores tentam conectar-se ao *broker* na *cloud*, que consiste numa instância do serviço CloudMQTT [68]. Para efetuar a conexão com sucesso é necessário autenticar-se com as credenciais corretas.

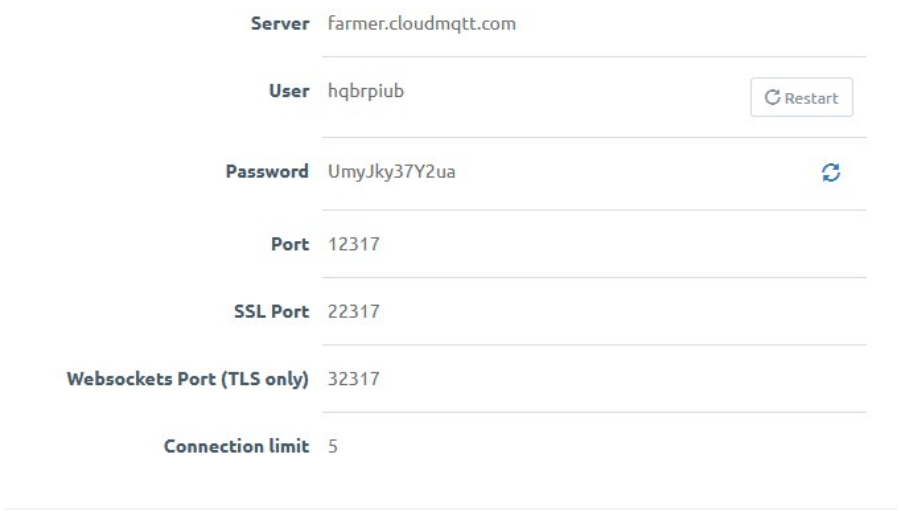


Figura 51. Definições do broker na cloud [68].

Esta opção vai aumentar a segurança do sistema, visto que previne as escutas de mensagens trocadas entre clientes. Na eventualidade da conexão não se efetuar ao broker na *cloud*, os diferentes componentes comunicam recorrendo a um *broker* Mosquitto [67] a ser executado localmente no Raspberry Pi. Tal como no caso do *broker* na *cloud*, a autenticação é necessária para efetuar a conexão.

```
mqttClient.onConnect(onMqttConnect);
mqttClient.onDisconnect(onMqttDisconnect);
mqttClient.onSubscribe(onMqttSubscribe);
mqttClient.onUnsubscribe(onMqttUnsubscribe);
mqttClient.onMessage(onMqttMessage);
mqttClient.onPublish(onMqttPublish);
mqttClient.setServer(mqttServerCloud, mqttPort);
mqttClient.setCredentials(mqttUser,mqttPassword);
mqttClient.setClientId(uuidChar_mqtt);
mqttClient.setCleanSession(false);
mqttClient.connect();
```

Figura 52. Ligação a broker na cloud.

3.5.6. Servidor central

O servidor central é responsável por receber e tratar todos os pedidos provenientes da aplicação web, da aplicação móvel, da Alexa ou dos módulos microcontrolador. Tem suporte a dois tipos de comunicação, o HTTP e MQTT. No caso do HTTP, os pedidos são efetuados para uma API REST que se encontra no anexo B, sendo que todos os pedidos são efetuados por HTTPS. Esta API suporta pedidos POST, GET e DELETE.

A criação da API fez-se recorrendo à linguagem *Python* e com a *framework* Flask, sendo que, quase todos os *endpoints* estão protegidos de acesso não autorizado através do uso de *access tokens*. Os *endpoints* não protegidos são os do login, do registo e o de mudar a password. Além da necessidade do uso de *tokens* para ser possível o acesso aos recursos, é da responsabilidade de um *middleware* verificar a identidade do *token,* ou seja*,* se corresponde ao id de um utilizador válido. Todos os *tokens* expiram num espaço de 5 minutos, sendo necessária a sua renovação através do uso de um *refresh token* criado no momento da autenticação.

No caso do protocolo MQTT, é usado para estabelecer uma comunicação com os microcontroladores. Os microcontroladores publicam os dados dos sensores periodicamente, de hora em hora, e esta frequência pode ser alterada pelo utilizador. Por exemplo, no caso do envio de dados da temperatura, o microcontrolador envia dados como o nome, o seu id, o id do sensor, a data da medição e o valor da medição. O servidor central, ao receber estes dados, armazena-os na base de dados. O utilizador, ao executar ações sobre um atuador, vai permitir ao servidor central executar o tratamento do pedido, como por exemplo efetuar *logging* localmente em ficheiros de texto e registar na base de dados a ação executada e consequentemente enviar o comando através de uma mensagem MQTT para o microcontrolador adequado. Para enviar a mensagem para o microcontrolador correto e não efetuar um *Broadcast*, esta é enviada para um tópico no qual o respetivo microcontrolador é o único subscrito.

Este servidor possui uma arquitetura modular, onde cada componente está segmentado pelas respetivas responsabilidades. O módulo da base de dados efetua a ligação à base de dados, o módulo de *logging* trata de efetuar o registo em log de todo o fluxo de dados pelo sistema, ou seja, este módulo é responsável por armazenar localmente, em ficheiros de texto, todos os dados que circulam pelo sistema, quer sejam pedidos do utilizador, quer sejam novos dados dos sensores que chegam ao sistema. O módulo MQTT, tal como o nome indica, é responsável por toda a gestão da comunicação do protocolo MQTT. Também possui o diretório da API, que está segmentado pelo módulo das rotas, pelos modelos, que são acedidos através de um ODM e pelos controladores.

Este servidor tem uma salvaguarda, na eventualidade de parar por algum motivo. Nesse caso, um *script* a ser executado em paralelo verifica se o processo do servidor está a ser executado, e caso não esteja, reinicia o servidor.

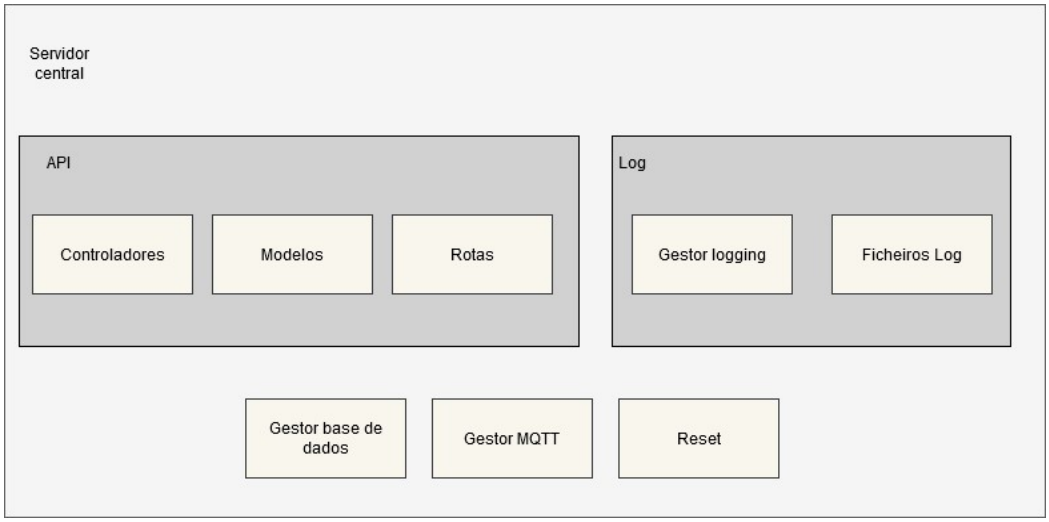


Figura 53. Arquitetura do servidor central.

3.5.7. Módulo microcontrolador

O módulo microcontrolador foi desenvolvido utilizando o microcontrolador NodeMCU, através da linguagem de programação C++ e com recurso ao IDE da plataforma Arduino. Este módulo está ligado ao servidor central e é responsável pela gestão dos sensores e atuadores. Os sensores utilizados, abordados na secção 3.4, foram o DHT22, que permite efetuar leituras da temperatura e da humidade e o sensor analógico LDR, que ao variar a luz incidida, varia a sua resistência, o que permite medir a quantidade de luz. Uma imagem do esquemático e uma imagem do circuito podem ser verificadas nas figuras 54 e 55 respetivamente.

.

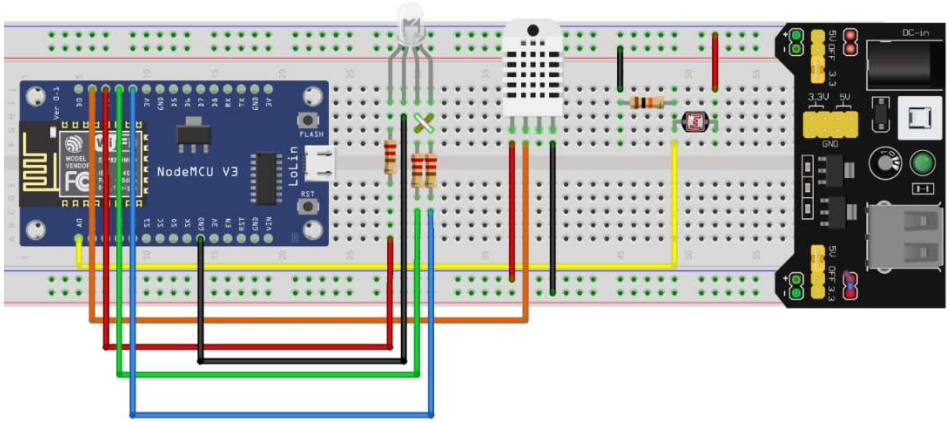


Figura 54. Esquemático do circuito do módulo microcontrolador.

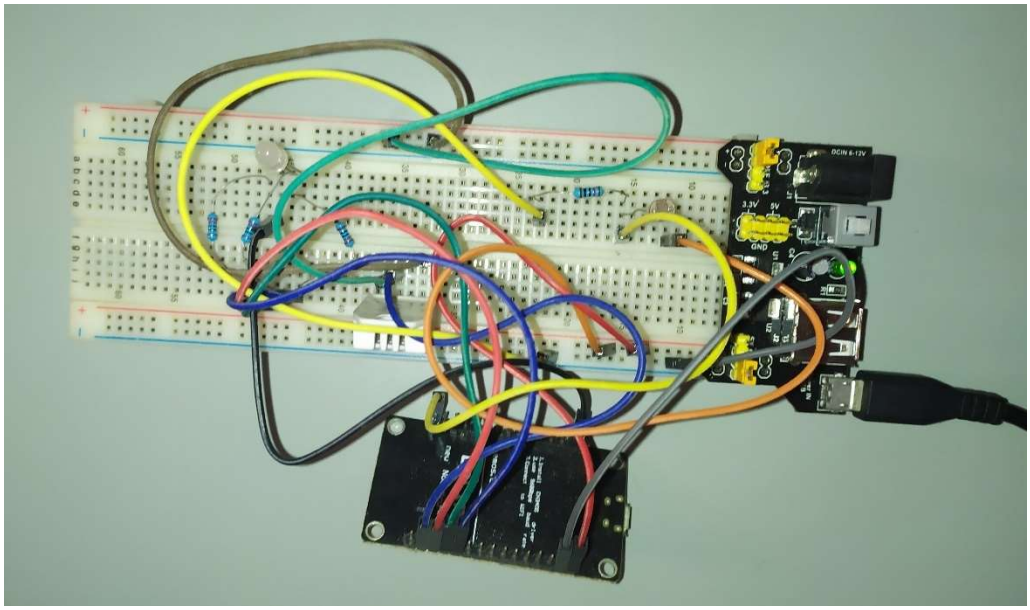


Figura 55. Circuito do módulo microcontrolador.

Como atuador foi utilizada um LED RGB, que permite ao utilizador ligar/desligar a LED, mudar a sua cor e intensidade. Também foi usado um *breadboard power supply* que se encontra no extremo direito das imagens 52 e 53. Este componente permite alimentar o microcontrolador de forma contínua, quer a 3.3 V quer a 5V, sem a necessidade de alimentar o microcontrolador através de uma porta USB ou bateria. Neste projeto o microcontrolador foi alimentado a 5V visto que os fabricantes dos sensores de temperatura e humidade recomendam esta voltagem, pois a 3V os sensores poderiam apresentar valores incorretos.

A arquitetura do módulo microcontrolador corresponde a uma arquitetura modular, onde cada componente/responsabilidade está separada pelas suas respetivas classes. Esta abordagem permite um sistema mais escalável e modificável. Cada sensor e atuador tem as suas classes, onde toda a sua gestão é efetuada. Foi implementada uma classe responsável pela ligação WiFi e outra pela ligação MQTT, sendo que, com esta implementação, é fácil a substituição por outro qualquer protocolo de comunicação. Também possui um módulo responsável pela gestão da hora/data e outro pela gestão do servidor HTTP local. Para finalizar, tem o módulo microcontrolador, que é responsável pela gestão e controlo de todos os outros módulos.

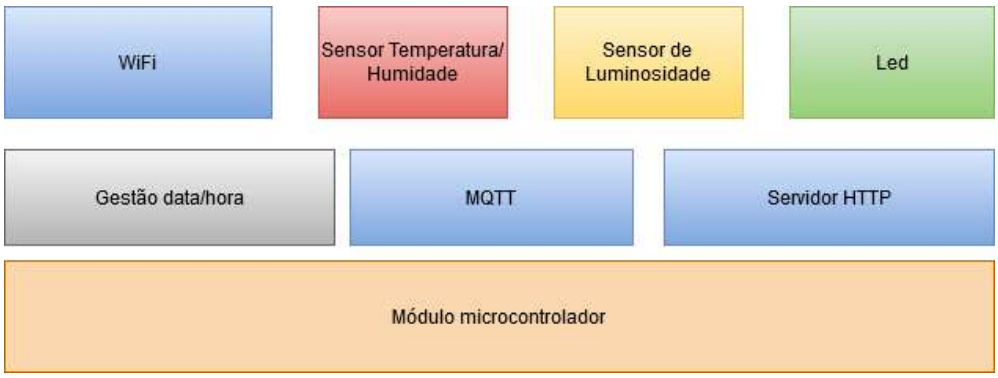


Figura 56. Arquitetura do módulo microcontrolador.

Através do suporte do módulo WiFi, inicialmente o microcontrolador fica no modo *Access Point*. O utilizador, ao aceder o IP do microcontrolador, é redirecionado para uma página web onde poderá introduzir as credenciais da sua rede local. Após o preenchimento do formulário, o microcontrolador passa ao modo *station* e conecta-se via WiFi à rede local. Na eventualidade da ligação se efetuar com sucesso, o microcontrolador armazena na sua memória as credenciais da rede local, de forma que, ao reiniciar, estabeleça a ligação automaticamente.

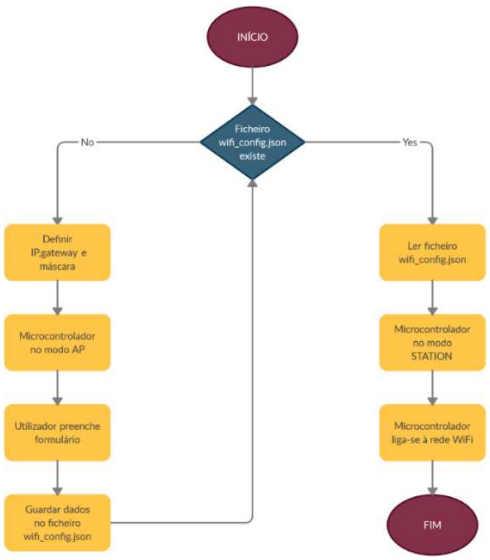


Figura 57. Fluxograma da ligação do microcontrolador à rede local.

Pelo facto de possuir suporte a ligações WiFi, foi desenvolvida uma API REST local, acessível através de pedidos HTTP. Através desta API é possível efetuar pedidos para obter valores dos sensores em tempo real ou o estado dos atuadores. Esta API foi desenvolvida de forma a que os clientes pudessem ter acesso a estes dados sempre que quisessem e não estivessem dependentes da sua entrega por parte do *publisher* do protocolo MQTT. Como medida de segurança, esta API está protegida por um mecanismo de *basic authentication*, ou seja, apenas pedidos HTTP autenticados obterão uma resposta válida. Pelo facto de possuir suporte a WiFi, foi desenvolvida uma componente OTA que permite atualizar todos os nodos remotamente através de WiFi. Isto é particularmente útil, visto que alguns nodos podem estar em lugares de difícil acesso e é algo que poupa tempo, já que permite atualizar todos os nodos de uma só vez.

Cada microcontrolador possui um ID único atribuído quando o utilizador regista o novo nodo no sistema. Desta forma, o servidor central sabe identificar e comunicar com o nodo pretendido, sem necessidade de efetuar *broadcasts*. Os sensores e atuadores também possuem ids únicos, sendo que, ao enviarem os valores lidos pelos sensores ou ao serem efetuadas ações pelos atuadores, são identificados na base de dados da *cloud*, através do seu identificador único. Sempre que o microcontrolador é reiniciado, os dados armazenados na memória local, como os ids dos sensores/atuadores, o id do nodo e o estado geral dos atuadores, são restabelecidos.

Além do HTTP, este módulo tem suporte ao protocolo MQTT, que permite uma comunicação bidirecional com o servidor central. Todos os tópicos que o microcontrolador subscreva têm concatenados o respetivo id, de forma a que o servidor central comunique apenas com o nodo pretendido. Os sensores efetuam leituras e esses dados são enviados em formato JSON para o servidor central de hora a hora, porém esta frequência pode ser alterada pelo utilizador. São enviados dados como nome, id, id do sensor, hora, data e valor. De forma a enviar o *timestamp* em cada mensagem, o microcontrolador efetua pedidos a um servidor NTP (europe.pool.ntp.org). Na ligação entre o servidor e os microcontroladores é usado um *broker*, e, no caso deste sistema, é usado um *broker* na *cloud* e um *broker* a executar localmente no Raspberry Pi como um mecanismo de redundância passiva. Como mecanismo de segurança, para poder ligar-se ao *broker,* é necessário autenticar-se. De forma a garantir que não haja perda de dados e duplicação de mensagens, é usado o nível 2 de QoS do MQTT. Desta forma, mesmo que haja uma falha na comunicação com o servidor, é garantida a entrega da mensagem.

4. Testes e Resultados

Neste capítulo são apresentados os testes de usabilidade a cada uma das aplicações desenvolvidas, através de um conjunto de tarefas, e efetuada uma análise/discussão dos resultados. Da mesma forma, é feita uma análise, através de entrevistas informais, da interação dos utilizadores com as diferentes aplicações ao longo da sua utilização doméstica mais prolongada, a fim de poder retirar conclusões sobre as diferentes formas de interagir com o sistema e assim, poder verificar qual das aplicações desenvolvidas mais se adequa a um sistema IoT, tendo em conta, por exemplo, a faixa etária dos utilizadores.

4.1. Testes de usabilidade

Nesta secção, são analisados os diferentes testes de usabilidade para cada uma das diferentes aplicações. Para tal, efetuou-se testes individualmente a cada um dos diferentes modos de interação. No total, estes testes de usabilidade foram realizados com 12 participantes, divididos em 3 grupos. Como teste de usabilidade, foi escolhido o SUS *(System Usability Scale)* [88], que consiste em 10 questões que usam a escala de Likert e que variam entre concordo fortemente a discordo fortemente. As 10 questões encontram-se listadas abaixo. Para a criação do formulário foi usada a ferramenta Google Forms.

1. Acho que gostaria de usar esse sistema com frequência.
2. Acho o sistema desnecessariamente complexo.
3. Achei o sistema fácil de usar.
4. Acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.
5. Acho que as várias funções do sistema estão muito bem integradas.
6. Acho que o sistema apresenta muita inconsistência.
7. Imagino que as pessoas aprenderão como usar esse sistema rapidamente.
8. Achei o sistema atrapalhado de usar.
9. Senti-me confiante ao usar o sistema.
10. Precisei aprender várias coisas novas antes de conseguir usar o sistema.

4.1.1. Teste de usabilidade da aplicação web

O teste de usabilidade da aplicação web foi realizado por 4 utilizadores, 2 do sexo feminino e 2 do sexo masculino. Para este teste foi pedido aos participantes para realizarem um conjunto de 10 tarefas, que se encontram listadas abaixo.

1. criar conta;
2. efetuar login;
3. adicionar novo nodo;
4. verificar última leitura da temperatura máxima de um nó;

5. verificar leituras da humidade do dia anterior de um nó;
6. ligar/desligar led de um nó;
7. mudar nome de um nó;
8. verificar leituras de um nó desativado;
9. apagar nó;
10. verificar *logs* do dia atual.

Durante a realização dos testes, o tempo que os participantes demoravam a realizar as tarefas foi monitorizado, de forma a apresentar mais uma métrica de comparação entre as diferentes plataformas. Após a realização das tarefas, os utilizadores preencheram um formulário de forma a ser possível calcular o índice de usabilidade. Para que os resultados fossem mais fidedignos, escolheu-se participantes com idades e habilitações académicas distintas. Metade dos participantes tinham idades entre os 18 e os 30 anos e a outra metade entre os 51 e os 60 anos. Relativamente às habilitações académicas, metade dos participantes tinha licenciatura e a outra metade o secundário.

Na figura 58 é possível observar o tempo médio da realização de cada uma das tarefas.

Figura 58. Tempo médio por tarefa na aplicação web em segundos.

A tabela 12 representa as respostas e os resultados de cada um dos participantes que responderam ao questionário SUS. Para o cálculo do índice de usabilidade, deve-se retirar uma unidade da pontuação das respostas ímpares e nas respostas pares deve-se subtrair 5 unidades ao valor escolhido pelo participante. Para o cálculo final, que tem valores entre 0 e 100, deve-se somar todos os valores das 10 questões e multiplicar por 2.5.

Tabela 12. Respostas do questionário SUI da aplicação web

Utilizadores	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Resultado
U1	4	1	3	1	4	1	4	3	3	1	80
U2	4	2	4	2	4	1	5	2	3	2	77.5
U3	4	1	5	1	5	2	5	1	5	1	95
U4	4	1	5	2	4	2	4	1	4	1	85

O cálculo da pontuação final, que corresponde à média dos resultados entre os participantes, deu um valor de 84.375, demonstrando que a aplicação web possui um bom índice de usabilidade.

Após a realização dos testes e do preenchimento do formulário, pediu-se a cada um dos participantes que apresentassem sugestões, alterações e propostas de melhoria para a aplicação, e daí retiraram-se algumas observações. Por exemplo, foi sugerido que se alterasse a cor das setas dos valores máximo e mínimos para vermelho e azul, ao invés de verde e vermelho. Outro aspeto diz respeito ao calendário, que não apresentava uma legenda, de forma a fornecer *feedback* sobre o seu propósito. Para finalizar, foi recomendado, que na vista dos *logs,* fosse apresentada à partida os *logs* do dia atual.

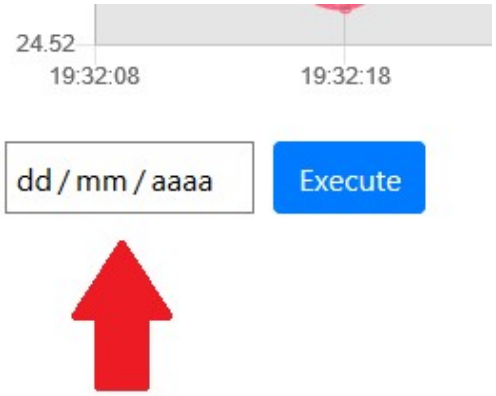


Figura 59. Problema da falta de feedback no calendário.

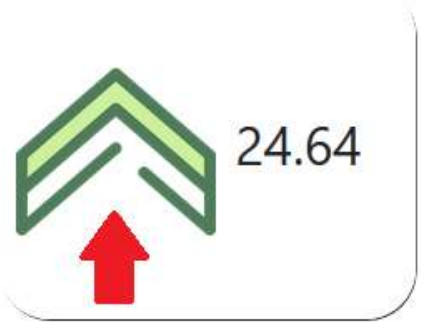


Figura 60. Sugestão da mudança de cor da seta para vermelho na aplicação web.

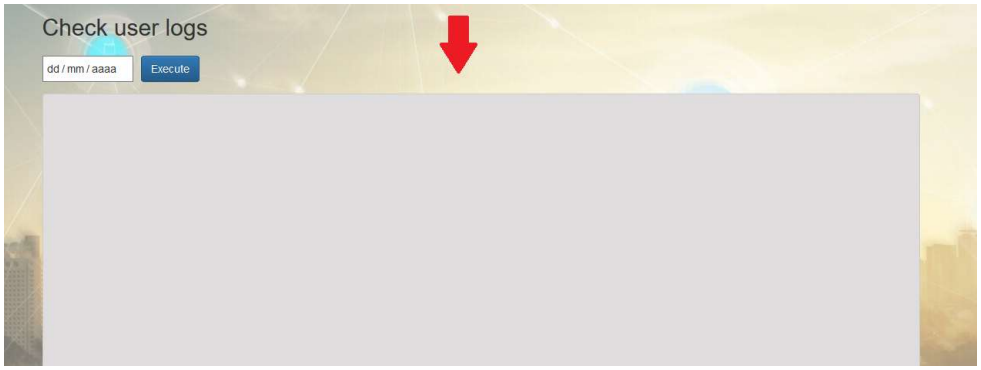


Figura 61. Sugestão que apareçam os logs do dia atual ao carregar na vista dos logs.

4.1.2. Teste de usabilidade da aplicação móvel

O teste de usabilidade da aplicação móvel, tal como na aplicação web, foi realizado com 4 participantes, 3 do sexo feminino e 1 do sexo masculino. A distribuição por faixa etária pode ser visualizada na figura 62. Os participantes realizaram um conjunto de 10 tarefas, as mesmas da aplicação web, onde foi monitorizado o tempo para a realização de cada tarefa.

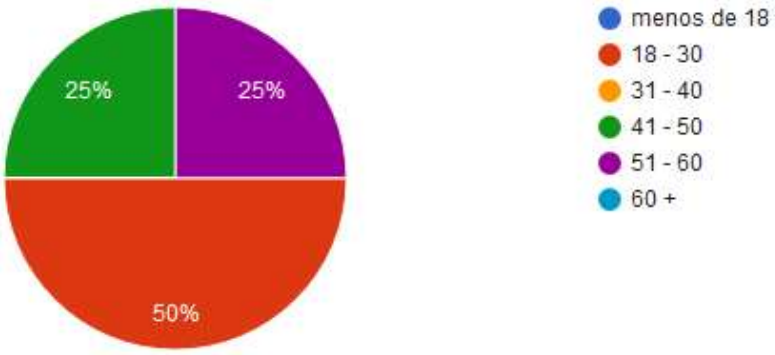


Figura 62. Distribuição etária dos participantes no teste de usabilidade da aplicação móvel.

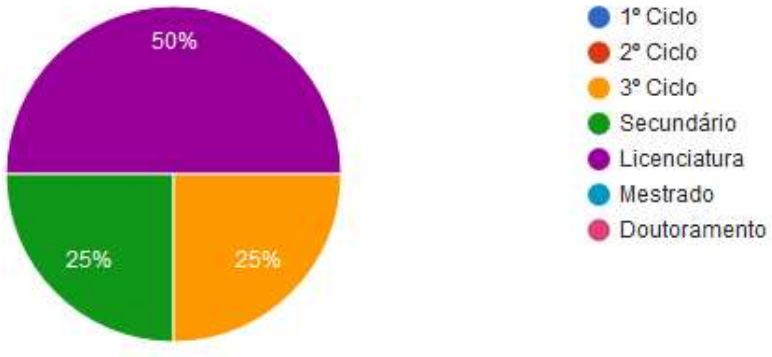


Figura 63. Habilitações académicas dos participantes no teste de usabilidade da aplicação móvel.

Na figura 64 é possível observar o tempo médio da realização de cada uma das tarefas.

Figura 64. Tempo médio por tarefa na aplicação móvel em segundos.

A tabela 13 apresenta as respostas e os resultados de cada um dos participantes que responderam ao questionário SUS.

Tabela 13. Respostas do questionário SUI da aplicação móvel

Utilizadores	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Resultado
U1	4	2	4	1	5	1	4	1	4	1	87.5
U2	3	4	3	1	3	2	4	3	3	2	60
U3	4	1	5	1	5	2	4	1	4	1	90
U4	5	1	4	2	5	1	4	2	4	2	85

O cálculo da pontuação final deu um valor de 80.625 o que indica que a aplicação móvel possui um bom índice de usabilidade.

Posteriormente, foi pedido a cada um dos participantes que apresentassem sugestões, alterações e melhorias para a aplicação. Tal como na aplicação web, foi sugerido alterar as cores das setas dos valores máximos para vermelho e dos mínimos para azul e apresentar os *logs* do dia atual ao carregar a vista dos *logs*.

Figura 65. Sugestão de mudança de cor da seta para vermelho na aplicação móvel.

4.1.3. Teste de usabilidade da *skill* da Alexa

Este teste consistiu num conjunto de tarefas, através de uma interface de voz, realizada por 4 participantes, todos do sexo masculino. Todos os participantes tinham idades entre os 18 e os 30 anos. No que diz respeito às habilitações, 75% eram licenciados e 25% mestres. Foi pedido aos participantes para realizarem um conjunto de 5 tarefas, que se encontram abaixo listadas.

1. criar um nodo;
2. verificar temperatura máxima de um nó;
3. ligar/desligar led de um nó;
4. mudar frequência de envio de dados da humidade num nó;
5. apagar nó.

Durante a realização dos testes, além do tempo decorrido a realizar cada tarefa, foi também monitorizado o número de falhas cometidas pelos participantes ao realizarem os seus pedidos por fala. Para a realização destes testes foi usada a aplicação móvel da Alexa.

Nas figuras 66 e 67 é possível verificar que a maioria dos participantes não usa assistentes virtuais com frequência, mas já usaram alguns deles por curiosidade.

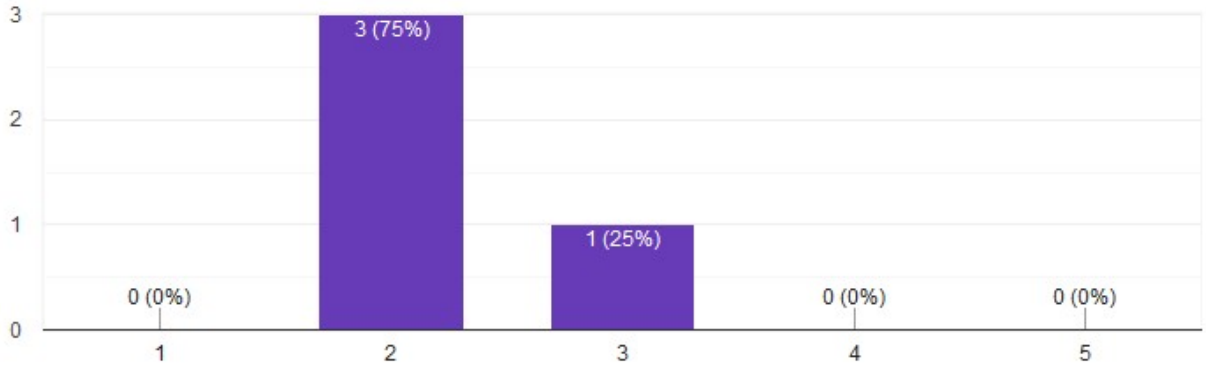


Figura 66. Frequência de utilização de assistentes virtuais.

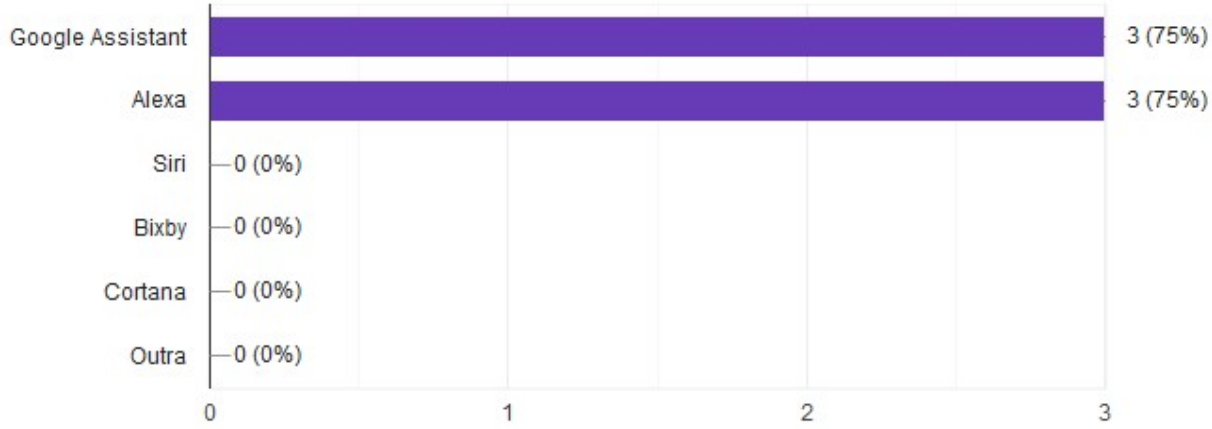


Figura 67. Assistentes virtuais usados pelos participantes.

Como é possível visualizar na figura 67, a maioria dos participantes está familiarizada com os assistentes virtuais da Google e da Alexa, principalmente com o Google Assistant, por este já estar pré-instalado em dispositivos Android.

Durante o decorrer dos testes, foi possível fazer algumas observações, tais como o tempo de execução da tarefa e o número de erros cometidos a realizar cada tarefa. Estes parâmetros encontram-se representados nas figuras 68 e 69.

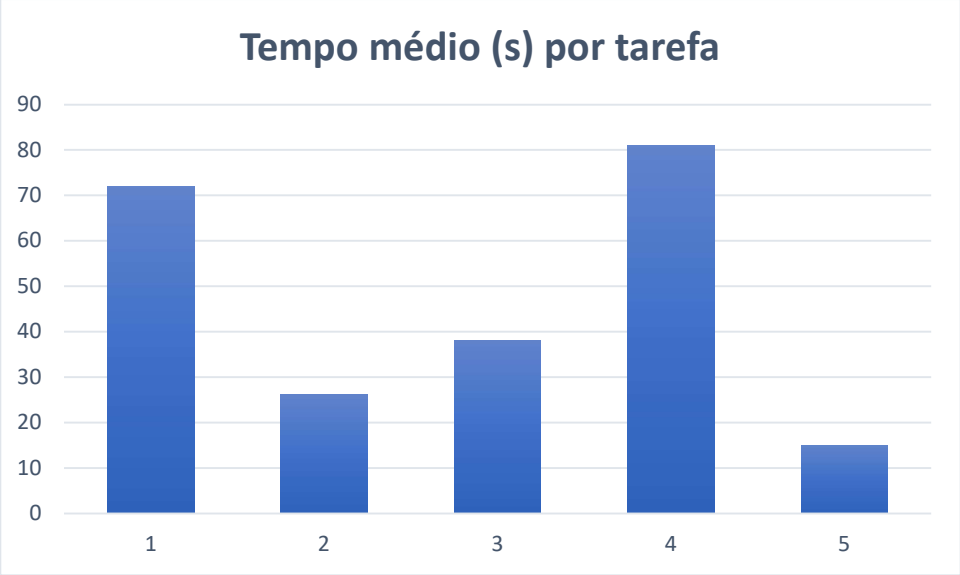


Figura 68. Tempo médio por tarefa na skill da Alexa em segundos.

Figura 69. Média de erros na pronunciação do pedido à Alexa.

Os erros, ao pronunciar os pedidos ao assistente, aconteceram devido a falhas de interpretação de *utterances* que não constavam no modelo de interação da *skill* ou devido a uma falha ao interpretar o que o utilizador pediu.

Na tabela 14 estão apresentadas as respostas e os resultados de cada um dos participantes que responderam ao questionário SUS.

Tabela 14. Respostas do questionário SUI da Alexa

Utilizadores	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Resultado
U1	4	2	3	2	4	2	3	2	3	1	70
U2	5	1	3	1	4	1	3	2	4	2	80
U3	4	1	3	1	4	1	3	2	4	1	80
U4	4	2	4	2	4	2	2	2	4	2	70

Após analisar os resultados, foi calculado o valor do índice de usabilidade, que atingiu o valor de 75%, indicando uma boa usabilidade, apesar de inferior às outras duas. A principal sugestão de melhoria por parte dos participantes foi a extensão do modelo de interação, de forma a estar mais bem preparado para interpretar mais expressões mencionadas pelos participantes, o que levaria a uma minimização de erros.

4.2. Comparação dos resultados

Tabela 15. Comparação dos resultados das diferentes aplicações.

Teste de usabilidade	SUS	Tempo médio(s)
App web	84.375	16.51
App móvel	80.625	23.57
Alexa *skill*	75	46.94

Observando os valores da tabela 15 é possível concluir que a aplicação web foi a que teve melhores resultados globais. Em contraste, a *skill* da Alexa obteve os resultados com menor qualidade, principalmente no que diz respeito ao tempo médio na execução de cada tarefa. Estes resultados vão de encontro às conclusões dos artigos [12] e [13], onde foram apresentados problemas relativos à utilização de assistentes virtuais, como por exemplo, os utilizadores não saberem como efetuar pedidos, uma vez que não há pistas visuais, terem que se lembrar da lista de comandos possíveis e efetuarem pedidos com má pronúncia. Estes problemas levaram a um aumento considerável do tempo de execução de algumas tarefas, visto que, nestas situações, os utilizadores pronunciavam os seus pedidos de uma forma que o assistente não estava preparado para interpretar e também por pronunciarem algumas palavras de forma incorreta. Uma forma de mitigar estes problemas seria estender o modelo de interação da Alexa de modo a reduzir estes erros de interpretação e, consequentemente, melhorar o valor de usabilidade e o tempo médio de realização das tarefas.

4.3. Observações dos utilizadores

Além dos testes de usabilidade, também foram realizadas instalações em três casas, para os utilizadores interagirem com o sistema de forma mais prolongada, durante 5 dias. O objetivo destas instalações foi inferir, a partir da utilização das diferentes aplicações por parte dos utilizadores, qual delas é a melhor em termos de usabilidade para um utilizador final interagir com um sistema desta natureza e assim verificar qual das aplicações desenvolvidas mais se adequa a um sistema IoT.

A partir de entrevistas informais pós utilização, foi possível concluir que, apesar do assistente virtual ter apresentado um índice de usabilidade inferior à aplicação web/móvel, foi o método que despertou mais curiosidade e foi o mais usado pelos utilizadores ao longo da experiência, por ter sido uma forma de comunicação mais natural e intuitiva. Este último ponto foi mais acentuado nos utilizadores de idade mais avançada (40-60 anos), visto que muitos deles não utilizavam com muita frequência aplicações web/móveis. Por este motivo, muitos deles recorriam ao assistente virtual, por considerarem uma forma mais natural e fácil de efetuarem pedidos.

No caso dos utilizadores mais jovens, foi relatado pelos próprios nas entrevistas informais, que, inicialmente possuíam um maior interesse pelo assistente virtual, visto que usavam com mais frequência esta forma de interação, interesse que foi diminuindo gradualmente, pelo que, passaram a recorrer mais à aplicação móvel para interagir com o sistema. O principal motivo apresentado pelos participantes em questão, foi o facto de estarem mais habituados a trabalhar com aplicações web/móvel e por ser algo quase instintivo/habitual utilizar o telemóvel para qualquer tarefa. Outro motivo apontado, foram certas limitações apresentadas pelo assistente virtual, principalmente na exposição de dados temporais. Alguns utilizadores recorriam sempre às outras plataformas para visualizar dados temporais, visto que a *skill* da Alexa não apresentava essa funcionalidade. Outro ponto negativo relacionado com a utilização da *skill* foram as falhas de interpretação por parte da Alexa. Estas falhas e o facto de o assistente virtual não perceber o pedido, eram mais frustrantes do ponto de vista de usabilidade para os participantes, do que estar à procura da funcionalidade nas outras aplicações. A grande maioria destes erros deveu-se a falhas de interpretação de *utterances* que não constavam no modelo de interação da *skill*, mas também devido a falhas na compreensão do que foi pedido pelo utilizador, pois alguns utilizadores pronunciavam mal algumas palavras em inglês.

5. Conclusão e Trabalho Futuro

Nesta secção são apresentadas as conclusões finais do projeto desenvolvido, um resumo do processo de desenvolvimento e propostas para trabalho futuro.

Este projeto teve como objetivo principal o estudo de diferentes formas de interação de um utilizador com um sistema IoT, para apurar qual a forma mais prática e adequada de interação. Para tal, foi desenvolvido um sistema IoT, com suporte a diferentes formas de interação do utilizador com o sistema, tais como, uma aplicação web, uma aplicação móvel e um assistente virtual. Inicialmente, foram analisados os diversos protocolos mais usados em sistemas IoT, mais concretamente protocolos da camada de aplicação e da camada de ligação de dados e física. Foram também analisados os diversos assistentes virtuais disponíveis no mercado, tais como a Alexa, assistente virtual escolhido para o desenvolvimento da aplicação de voz, devido à boa documentação apresentada e por permitir a extensão das suas funcionalidades bases através do desenvolvimento de *skills*.

Posteriormente, foram estipulados os requisitos funcionais e não funcionais, sendo que, no total, o sistema teve 32 requisitos, 19 não funcionais e 13 funcionais. Subsequentemente, foi desenhada a arquitetura do sistema, que consistiu num módulo microcontrolador, num servidor central, em brokers MQTT, numa base de dados remota na *cloud*, em sensores e atuadores, num website, numa aplicação móvel e numa *skill* na Alexa. Para a escolha dos microcontroladores, foi selecionado o NodeMCU, devido ao seu baixo custo e por ter suporte a ligações WiFi através de um módulo pré-montado. O módulo microcontrolador foi encarregue pela gestão dos sensores e atuadores. No que diz respeito ao servidor central, ficou hospedado num Raspberry Pi e foi responsável por receber e tratar todos os pedidos da aplicação web, da aplicação móvel, da Alexa ou dos microcontroladores. Tem suporte a dois tipos de comunicação, o HTTP e MQTT, sendo que, no caso do HTTP os pedidos são efetuados para uma API REST. Para a escolha da base de dados, foi selecionado o MongoDB, uma base de dados NoSQL que armazena, por coleções de documentos JSON, todas as leituras e ações realizadas pelo sistema. Para esse fim, foi usado o MongoDB Atlas, um serviço de base de dados na *cloud*.

Realizaram-se testes de usabilidade para as três aplicações desenvolvidas, e o teste escolhido foi o SUS. A partir destes testes, foi possível calcular o índice de usabilidade para cada um dos casos, sendo que o maior foi a da aplicação web, com um valor de 84.374 e o menor foi o da *skill*, com 75 valores. Em ambos os casos, estes valores são classificados como bons. Os testes permitiram ainda verificar alguns problemas identificados pelos participantes, tais como, a mudança de cor de alguns ícones, o fornecimento de feedback em alguns botões nas aplicações web e móvel e o aumento do número de *uterrances* para cada intenção do modelo de interação da *skill* da Alexa, de forma a minimizar erros e falhas de interpretação dos pedidos. Além dos testes de usabilidade, instalou-se o sistema desenvolvido em 3 casas, para que os utilizadores interagissem com o sistema por um maior período de tempo. Assim, foi possível, através de entrevistas informais, dispor de mais informação que permitisse concluir sobre a melhor forma de interação. O assistente virtual, apesar de ter apresentado um índice de usabilidade inferior às outras soluções, foi, segundo o relato dos utilizadores nas entrevistas informais, o que mais captou a curiosidade, pelo facto de ser uma tecnologia e forma de interação relativamente novas, por comparação às restantes aplicações e por ser

uma forma mais natural e intuitiva de interação. Contudo, os utilizadores encontraram algumas barreiras na utilização do assistente virtual, como por exemplo, erros de pronúncia ao efetuar os pedidos orais ou desconhecimento de como efetuá-los. Outra limitação, foi a visualização de dados temporais, pois os participantes relataram que eram forçados a usar a aplicação web/móvel para este tipo de pedido.

Como trabalho futuro há alguns pontos que podem ser melhorados, por exemplo, acrescentar suporte a outras línguas na *skill* da Alexa. A opção por desenvolver a *skill* em inglês, foi por haver um maior suporte por parte da Amazon para esta língua e por ser uma língua com grande divulgação. No entanto, no futuro, poderia ser fornecido suporte à versão do português do Brasil. Outra melhoria seria a extensão do modelo de interação da *skill*, de forma a minimizar erros ao interpretar pedidos dos utilizadores. No que diz respeito à aplicação móvel, poderia ser desenvolvida uma aplicação para o sistema iOS através de desenvolvimento multiplataforma.

Em suma, o objetivo do projeto para a averiguação da melhor forma de interação foi atingido, já que, através dos testes de usabilidade e das experiências prolongadas com os utilizadores, foi possível concluir que as aplicações web/móvel estão mais bem preparadas para a realidade atual, visto que os assistentes virtuais ainda apresentam algumas limitações. Todavia, para apresentar uma resposta definitiva sobre a questão, seria necessário um maior número de utilizadores e de testes. Relativamente ao desenvolvimento do sistema IoT, a sua implementação foi executada com sucesso, uma vez que todos os requisitos inicialmente delineados foram implementados e cumpridos.

6. Referências

[1] C. R. da Junqueira 69, “A Internet das Coisas (IoT – Internet of Things).” https://www.cncs.gov.pt/a-internet-das-coisas-iot-internet-of-things/ (acedido Aug. 21, 2020).

[2] “Tecnologia da Informação e Comunicação,” InfoEscola. https://www.infoescola.com/informatica/tecnologia-da-informacao-e-comunicacao/ (accessed Apr. 17, 2021).

[3] “Conheça 6 aplicações da internet das coisas que já estão tornando o mundo melhor,” *Época Negócios*. https://epocanegocios.globo.com/Tecnologia/noticia/2019/03/conheca-6-aplicacoes-da-internet-das-coisas-que-ja-estao-tornando-o-mundo-melhor.html (acedido Aug. 21, 2020).

[4] “Internet das Coisas: exemplos, aplicações e vantagens – CCG.” http://www.ccg.pt/internet-das-coisas-exemplos-aplicacoes-e-vantagens/ (acedido Aug. 21, 2020).

[5] “IoT Advantage and Disadvantage - Javatpoint,” *www.javatpoint.com*. https://www.javatpoint.com/iot-advantage-and-disadvantage (acedido Aug. 21, 2020).

[6] “abrantesasf/cofre-IoT,” *GitHub*. https://github.com/abrantesasf/cofre-IoT (acedido Aug. 21, 2020).

[7] “Internet das coisas,” *Wikipédia, a enciclopédia livre*. Jul. 15, 2020, Acedido: Aug. 21, 2020. [Online]. Available: https://pt.wikipedia.org/w/index.php?title=Internet_das_coisas&oldid=58772209.

[8] M. Ali and A. M. Hassan, “Developing Applications for Voice Enabled IoT Devices to Improve Classroom Activities,” in 2018 21st International Conference of Computer and Information Technology (ICCIT), Dhaka, Bangladesh, Dec. 2018, pp. 1–4

[9] V. Chayapathy, G. S. Anitha, and B. Sharath, “IOT based home automation by using personal assistant,” in 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon), Bangalore, Aug. 2017, pp. 385–389

[10] M. Majchrowicz and M. Hufnagiel, “Management of IOT Devices in a Smart Home Through the Application of an Interactive Mirror,” Image Processing & Communications, vol. 22, no. 4, pp. 43–50, Dec. 2017

[11] M. Mikusz, S. Houben, N. Davies, K. Moessner, and M. Langheinrich, “Raising Awareness of IoT Sensor Deployments,” in Living in the Internet of Things: Cybersecurity of the IoT - 2018, London, UK, 2018, p. 9 (8 pp.)-9 (8 pp.)

[12] R. Kang, A. Guo, G. Laput, Y. Li, and X. “Anthony” Chen, “Minuet: Multimodal Interaction with an Internet of Things,” in *Symposium on Spatial User Interaction*, New Orleans LA USA, Oct. 2019, pp. 1–10

[13] C. Murad, C. Munteanu, B. R. Cowan, and L. Clark, “Revolution or Evolution? Speech Interaction and HCI Design Guidelines,” *IEEE Pervasive Comput.*, vol. 18, no. 2, pp. 33–45, Apr. 2019

[14] R. Xiao, G. Laput, Y. Zhang, and C. Harrison, “Deus EM Machina: On-Touch Contextual Functionality for Smart IoT Appliances,” in Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver Colorado USA, May 2017, pp. 4000–4008

[15] A. A. de Freitas, M. Nebeling, X. “Anthony” Chen, J. Yang, A. S. K. Karthikeyan Ranithangam, and A. K. Dey, “Snap-To-It: A User-Inspired Platform for Opportunistic Device Interactions,” in Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose California USA, May 2016, pp. 5909–5920

[16] “Communication protocol,” *Wikipedia*. Aug. 08, 2020, Acedido: Aug. 21, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Communication_protocol&oldid=971786680.

[17] “What is a Communication Protocol? - Definition from Techopedia,” *Techopedia.com*. http://www.techopedia.com/definition/25705/communication-protocol (acedido Aug. 21, 2020).

[18] S. Gautam, “Nginx WebServer with basic HTTP Protocol,” *Medium*, Dec. 28, 2018. https://medium.com/@me.sanjeev3d/https-medium-com-me-sanjeev3d-nginx-webserver-215b636afcc5 (acedido Nov. 25, 2020).

[19] “HTTP: Definition & How it Works | Protocol Support Library | ExtraHop.” https://www.extrahop.com/resources/protocols/http/ (acedido Aug. 25, 2020).

[20] C. Cimpanu, “The CoAP protocol is the next big thing for DDoS attacks,” *ZDNet*. https://www.zdnet.com/article/the-coap-protocol-is-the-next-big-thing-for-ddos-attacks/ (acedido Aug. 22, 2020).

[21] “CoAP Protocol: Step-by-Step Guide - DZone IoT,” dzone.com. https://dzone.com/articles/coap-protocol-step-by-step-guide (acedido Aug. 22, 2020).

[22] “CoAP.” https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/coap/funcionamento.html (acedido Aug. 22, 2020).

[23] “What is MQTT and How Does it Work?,” IoT Agenda. https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport (acedido Aug. 22, 2020).

[24] “Conhecendo o MQTT,” IBM Developer, Oct. 03, 2017. https://developer.ibm.com/br/technologies/iot/articles/iot-mqtt-why-good-for-iot/ (acedido Aug. 22, 2020).

[25] admin, “Configure MQTT runing on ESP8266 for Home Automation,” ESP8266 Shop, Jan. 02, 2019. https://esp8266-shop.com/blog/configure-mqtt-runing-on-esp8266-for-home-automation/ (acedido Nov. 25, 2020).

[26] “MQTT Publish/Subscriber - Protocolos para IoT - Embarcados,” Embarcados - Sua fonte de informações sobre Sistemas Embarcados, Jun. 15, 2015. https://www.embarcados.com.br/mqtt-protocolos-para-iot/ (acedido Aug. 22, 2020).

[27] “Protocolo MQTT - Redes 1.” https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/mqtt/ (acedido Nov. 25, 2020).

[28] “6 most commonly used IoT communication protocols,” TechGenix, Nov. 02, 2019. http://techgenix.com/iot-communication-protocols/ (acedido Aug. 24, 2020).

[29] breadware, “Top 7 Internet of Things (IoT) Communication Protocols,” Breadware, Dec. 11, 2018. http://dbz.164.myftpupload.com/2018/12/iot-communication-protocols/ (acedido Aug. 24, 2020).

[30] “8 Most Popular IoT Protocols and Standards You Need to Know | SaM Solutions.” https://www.sam-solutions.com:443/blog/internet-of-things-iot-protocols-and-connectivity-options-an-overview/ (acedido Aug. 24, 2020).

[31] “11 Internet of Things (IoT) Protocols You Need to Know About.” https://www.rs-online.com/designspark/eleven-internet-of-things-iot-protocols-you-need-to-know-about (acedido Aug. 25, 2020).

[32] “Virtual assistant,” *Wikipedia*. Nov. 28, 2020, Acedido: Nov. 29, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Virtual_assistant&oldid=991058879.

[33] “Artificial Intelligence in Customer Care: What, Why, and How,” Noble Systems, Jul. 15, 2019. https://www.noblesystems.com/artificial-intelligence-in-customer-care-what-why-and-how/ (accessed Apr. 20, 2021).

[34] Stephanie, “An Introduction To Intellligent Personal Assistants,” *Chatbot and Voice Assistant Solutions from Onlim.*, Mar. 20, 2018. https://onlim.com/en/introduction-to-intellligent-personal-assistants/ (acedido Nov. 29, 2020).

[35] “Can You Trust Intelligent Virtual Assistants? | No Jitter.” https://www.nojitter.com/can-you-trust-intelligent-virtual-assistants (acedido Nov. 29, 2020).

[36] “Amazon Alexa Official Site: What Is Alexa?” /alexa (acedido Nov. 25, 2020).

[37] “Amazon Echo Dot (3rd Generation), Anthracite | Lufthansa WorldShop.” https://www.worldshop.eu/en/product/Amazon-Echo-Dot-3rd-Generation-Anthracite/1756294?p=r6FbMSW3hDU (acedido Dec. 09, 2020).

[38] “Siri Vs Cortana Vs Alexa Vs Google Assistant Vs Bixby,” *Reach Byte*. https://reachbyte.com/siri-cortana-alexa-google-assistant-bixby/ (acedido Nov. 29, 2020).

[39] “Siri,” *Apple*. https://www.apple.com/siri/ (acedido Dec. 09, 2020).

[40] “Bixby | Toda a informação | Samsung Portugal.” https://www.samsung.com/pt/apps/bixby/ (acedido Dec. 09, 2020).

[41] “Bixby (software),” Wikipedia. Nov. 25, 2020, Acedido: Nov. 29, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Bixby_(software)&oldid=990558825.

[42] A. N. Voice, “Bixby fica mais acessível para deficientes visuais,” *NewVoice*, May 26, 2020. https://newvoice.ai/2020/05/26/bixby-fica-mais-acessivel-para-deficientes-visuais/ (acedido Dec. 09, 2020).

[43] “Cortana - Your personal productivity assistant,” Cortana - Your personal productivity assistant. https://www.microsoft.com/en-us/cortana (acedido Dec. 09, 2020).

[44] “Cortana,” *Wikipedia*. Nov. 02, 2020, Acedido: Nov. 29, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Cortana&oldid=986773931.

[45] “Futuro incerto? A Microsoft mudou a Cortana para a sua loja de aplicações,” *Pplware*, Jun. 30, 2019. https://pplware.sapo.pt/microsoft/microsoft-cortana-loja-aplicacoes/ (acedido Dec. 09, 2020).

[46] “Google Assistant, your own personal Google,” *Assistant*. https://assistant.google.com/ (acedido Dec. 09, 2020).

[47] “Google Assistant,” *Wikipedia*. Nov. 28, 2020, Acedido: Nov. 29, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Google_Assistant&oldid=991072926.

[48] “Virtual assistant services comparison: Alibaba Cloud, AWS, Google Cloud, IBM and Microsoft,” Latest Digital Transformation Trends | Cloud News | Wire19, Oct. 28, 2020. https://wire19.com/virtual-assistant-services-comparison/ (accessed Apr. 19, 2021).

[49] “Google Assistant já é tão usado como a Siri, 3 anos depois de lançado,” Pplware, Apr. 30, 2019. https://pplware.sapo.pt/google/google-assistant-siri-assistente-virtual/ (acedido Dec. 09, 2020).

[50] “NodeMCU: A placa de desenvolvimento muito low cost,” *Pplware*, Aug. 01, 2017. https://pplware.sapo.pt/gadgets/hardware/nodemcu-a-placa-de-desenvolvimento-muito-low-cost/ (acedido Aug. 26, 2020).

[51] J. B. at C.-Psap. in C. S. / S. T. E. Researcher, “ESP8266 NodeMCU pinout for Arduino IDE,” *Mechatronics Blog*, Nov. 19, 2019. https://mechatronicsblog.com/esp8266-nodemcu-pinout-for-arduino-ide/ (acedido Nov. 25, 2020).

[52] “DHT22 - Sensor de Umidade e Temperatura,” *Portugues*. https://www.vidadesilicio.com.br/dht22 (acedido Aug. 26, 2020).

[53] “buyhere22 - DHT22 Humidity and Temperature Sensor.” https://buyhere22.com/DHT22_Humidity_and_Temperature_Sensor (acedido Nov. 25, 2020).

[54] “DHT11, DHT22 and AM2302 Sensors,” *Adafruit Learning System*. https://learn.adafruit.com/dht/overview (acedido Aug. 26, 2020).

[55] “5MM GL5528 LIGHT DEPENDENT RESISTOR (LDR) – e2lab.” https://www.e2lab.tech/product/5mm-gl5528-light-dependent-resistor-ldr/ (acedido Nov. 25, 2020).

[56] “LDR,” *Wikipédia, a enciclopédia livre*. Apr. 10, 2019, Acedido: Aug. 26, 2020. [Online]. Available: https://pt.wikipedia.org/w/index.php?title=LDR&oldid=54774797.

[57] C. Fritzen, “Como funciona um LDR (resistor dependente de luz),” *FritzenLab*, Jan. 19, 2016. https://fritzenlab.com.br/2016/01/19/como-funciona-um-ldr-resistor-dependente-de-luz/ (acedido Aug. 26, 2020).

[58] T. R. P. Foundation, “Teach, Learn, and Make with Raspberry Pi,” Raspberry Pi. https://www.raspberrypi.org/ (accessed Apr. 23, 2021).

[59] “Beginner’s Guide: How to Get Started With Raspberry Pi,” PCMAG. https://www.pcmag.com/how-to/beginners-guide-how-to-get-started-with-raspberry-pi (accessed Apr. 23, 2021).

[60] “Raspberry Pi,” Wikipédia, a enciclopédia livre. Aug. 03, 2020, Accessed: Apr. 23, 2021. [Online]. Available: https://pt.wikipedia.org/wiki/Raspberry_Pi

[61] “ngrok - secure introspectable tunnels to localhost.” https://ngrok.com/ (acedido Dec. 09, 2020).

[62] R. Obeyesekera, “Getting Started with Ngrok,” *Medium*, Jul. 02, 2019. https://medium.com/aeturnuminc/getting-started-with-ngrok-ed67891a74bd (acedido Aug. 28, 2020).

[63] “Welcome to Flask — Flask Documentation (1.1.x).” https://flask.palletsprojects.com/en/1.1.x/ (accessed Apr. 17, 2021).

[64] “Django ou Flask, eis a questão.,” *Blog da TreinaWeb*, Feb. 12, 2019. https://www.treinaweb.com.br/blog/django-ou-flask-eis-a-questao/ (acedido Aug. 28, 2020).

[65] “Download Android Studio and SDK tools,” Android Developers. https://developer.android.com/studio?hl=pt-br (accessed Apr. 17, 2021).

[66] “What is Android Studio? - Definition from WhatIs.com,” SearchMobileComputing. https://searchmobilecomputing.techtarget.com/definition/Android-Studio (accessed Apr. 23, 2021).

[67] “Eclipse Mosquitto,” Eclipse Mosquitto, Jan. 08, 2018. https://mosquitto.org/ (accessed Apr. 23, 2021).

[68] “CloudMQTT - Hosted message broker for the Internet of Things.” https://www.cloudmqtt.com/ (acedido Nov. 25, 2020).

[69] “Managed MongoDB Hosting | Database-as-a-Service,” MongoDB. https://www.mongodb.com/cloud/atlas (accessed Apr. 17, 2021).

[70] “MongoDB Atlas — MongoDB Atlas,” https://github.com/mongodb/docs-bi-connector/blob/DOCSP-3279/source/index.txt. https://docs.atlas.mongodb.com/ (accessed Apr. 23, 2021).

[71] “What is AWS Lambda? - AWS Lambda.” https://docs.aws.amazon.com/lambda/latest/dg/welcome.html (acedido Oct. 11, 2020).

[72] “Amazon Lambda - AWS,” Amazon Web Services, Inc. https://aws.amazon.com/pt/lambda/ (accessed Apr. 19, 2021).

[73] “The most popular database for modern apps,” MongoDB. https://www.mongodb.com (accessed Apr. 17, 2021).

[74] “Amazon Developer Services.” https://developer.amazon.com/ (acedido Oct. 11, 2020).

[75] “Amazon Web Services (AWS) − Serviços de computação em nuvem,” *Amazon Web Services, Inc.* https://aws.amazon.com/pt/ (acedido Nov. 25, 2020).

[76] “Alexa Skills Kit Official Site: Build Skills for Voice.” /alexa/alexa-skills-kit (acedido Dec. 09, 2020).

[77] “Alexa Built-in Overview | Alexa Voice Service.” /alexa/techdoc-template (acedido Dec. 09, 2020).

[78] “What is Alexa Skills Kit? - Definition from WhatIs.com,” *SearchAWS*. https://searchaws.techtarget.com/definition/Alexa-Skills-Kit (acedido Oct. 11, 2020).

[79] “Alexa Voice Service v20160207 | Alexa Voice Service.” /alexa/techdoc-template (acedido Oct. 11, 2020).

[80] A. Channe, “Amazon Alexa,” *Medium*, Aug. 17, 2020. https://medium.com/datadriveninvestor/amazon-alexa-aaeb0dca9274 (acedido Nov. 25, 2020).

[81] Ashish, “Understanding Amazon’s Alexa and Building Alexa Skill,” *Medium*, Jun. 12, 2017. https://medium.com/@ashish_fagna/understanding-amazons-alexa-and-alexa-skill-6749785683b0 (acedido Oct. 11, 2020).

[82] “(3) How are apps for Amazon Alexa written? - Quora.” https://www.quora.com/How-are-apps-for-Amazon-Alexa-written (acedido Nov. 25, 2020).

[83] “Alexa Developer Console.” https://developer.amazon.com/alexa/console/ask (acedido Nov. 25, 2020).

[84] “Slot Type Reference | Alexa Skills Kit.” /alexa/techdoc-template (acedido Oct. 11, 2020).

[85] “Create and Edit Custom Slot Types | Alexa Skills Kit.” /alexa/techdoc-template (acedido Oct. 11, 2020).

[86] “Console de Gerenciamento da AWS,” *Amazon Web Services, Inc.* https://aws.amazon.com/pt/console/ (acedido Oct. 11, 2020).

[87] “Configure an Authorization Code Grant | Alexa Skills Kit.” /alexa/techdoc-template (acedido Aug. 31, 2020).

[88] A. S. for P. Affairs, “System Usability Scale (SUS),” Sep. 06, 2013. system-usability-scale.html (acedido Nov. 25, 2020).

Anexos

Anexo A – Protótipos de baixa fidelidade

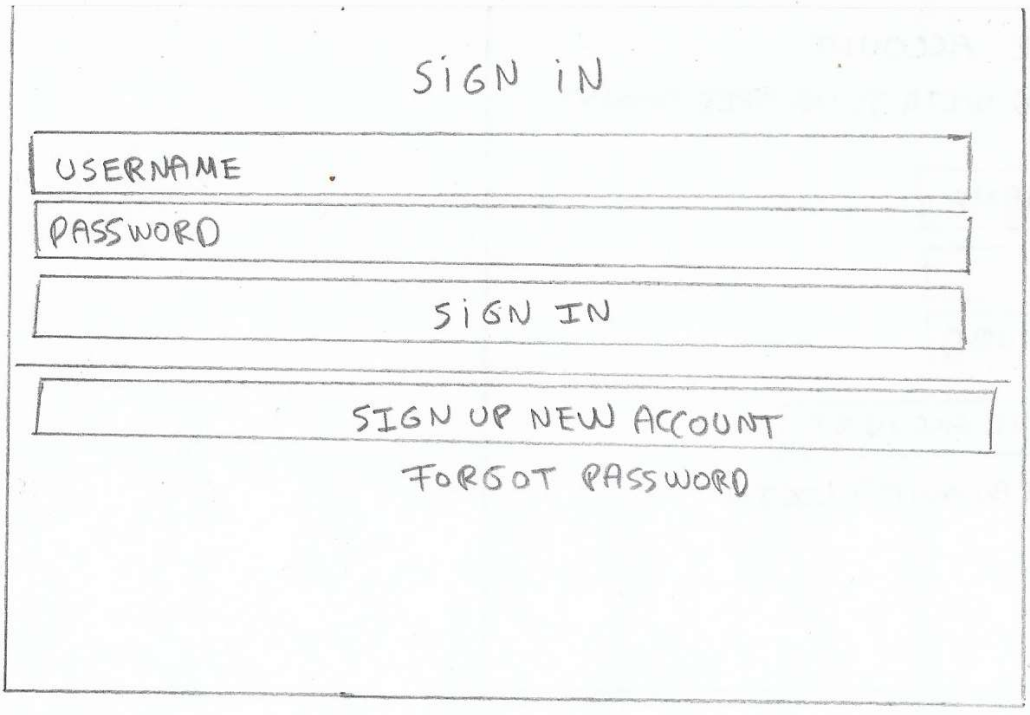


Figura 70. Vista de login da aplicação web protótipo baixa fidelidade

Figura 71. Vista de registo da aplicação web protótipo baixa fidelidade.

Figura 72. Vista de reset password da aplicação web protótipo baixa fidelidade.

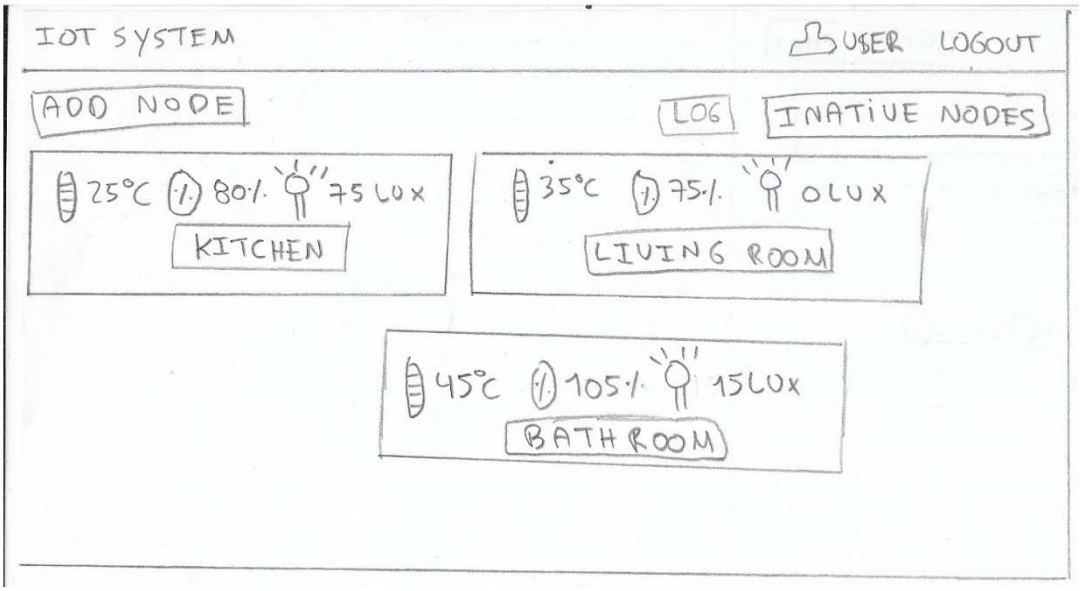


Figura 73. Vista da página inicial de listagem dos nodos da aplicação web protótipo baixa fidelidade.

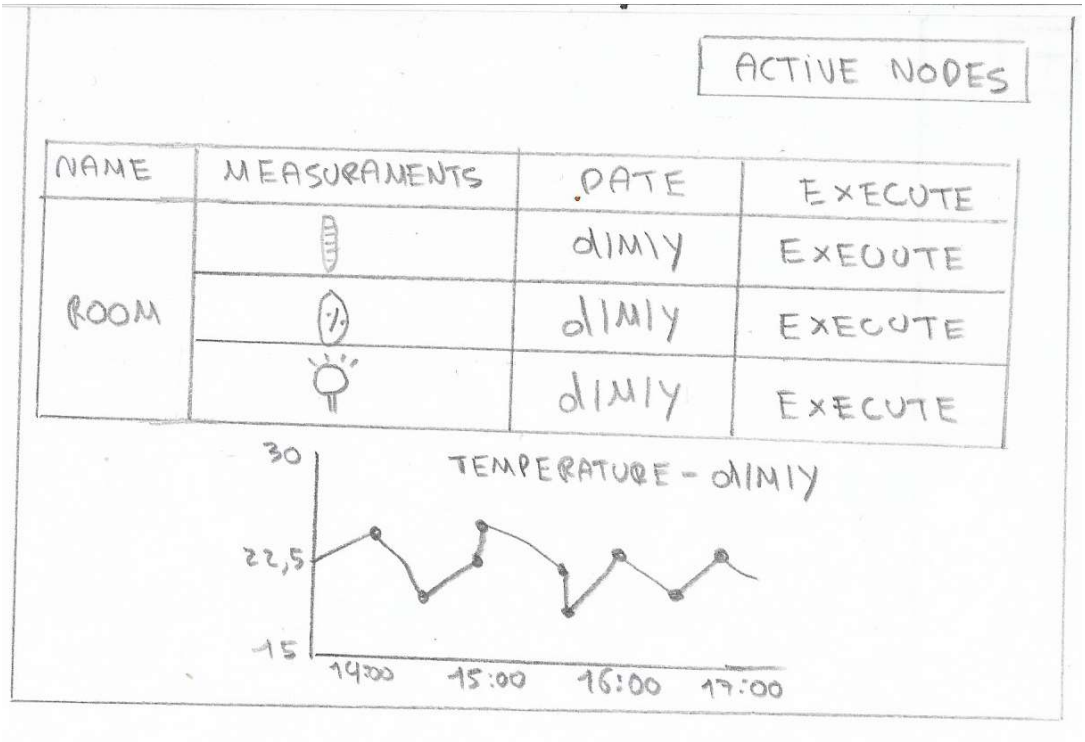


Figura 74. Vista da visualização de dados de nós inativos da aplicação web protótipo baixa fidelidade.

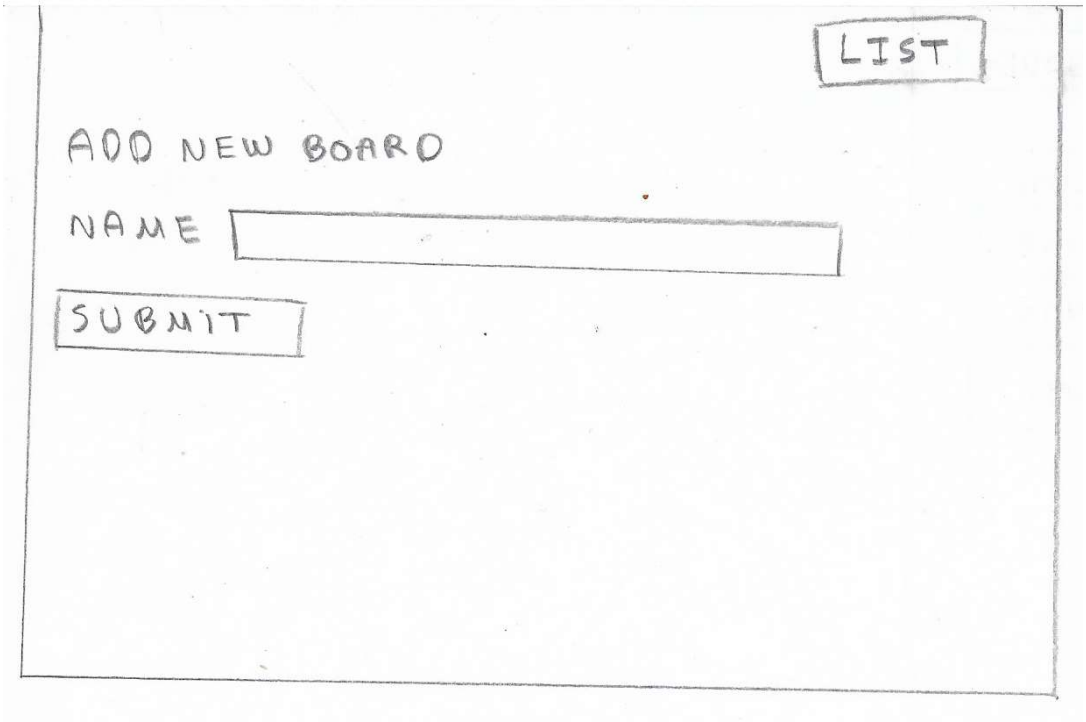


Figura 75. Vista de registo de novo nodo da aplicação web protótipo baixa fidelidade.

Figura 76. Vista de logs da aplicação web protótipo baixa fidelidade.

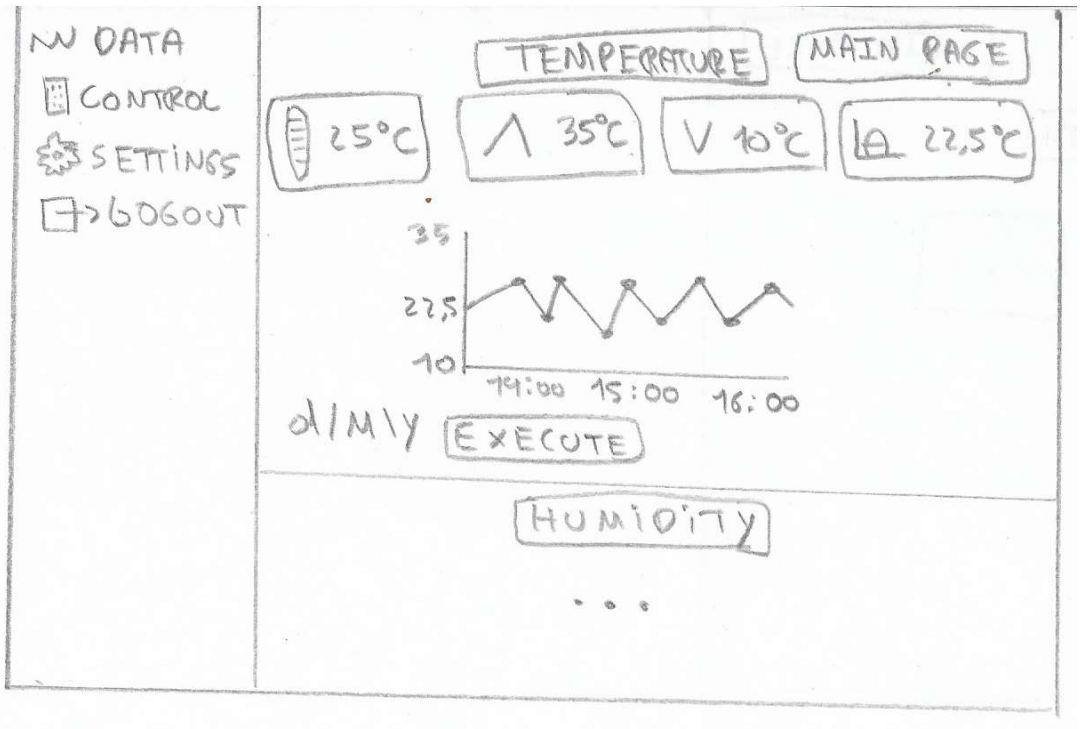


Figura 77. Vista de dados de um nó da aplicação web protótipo baixa fidelidade.

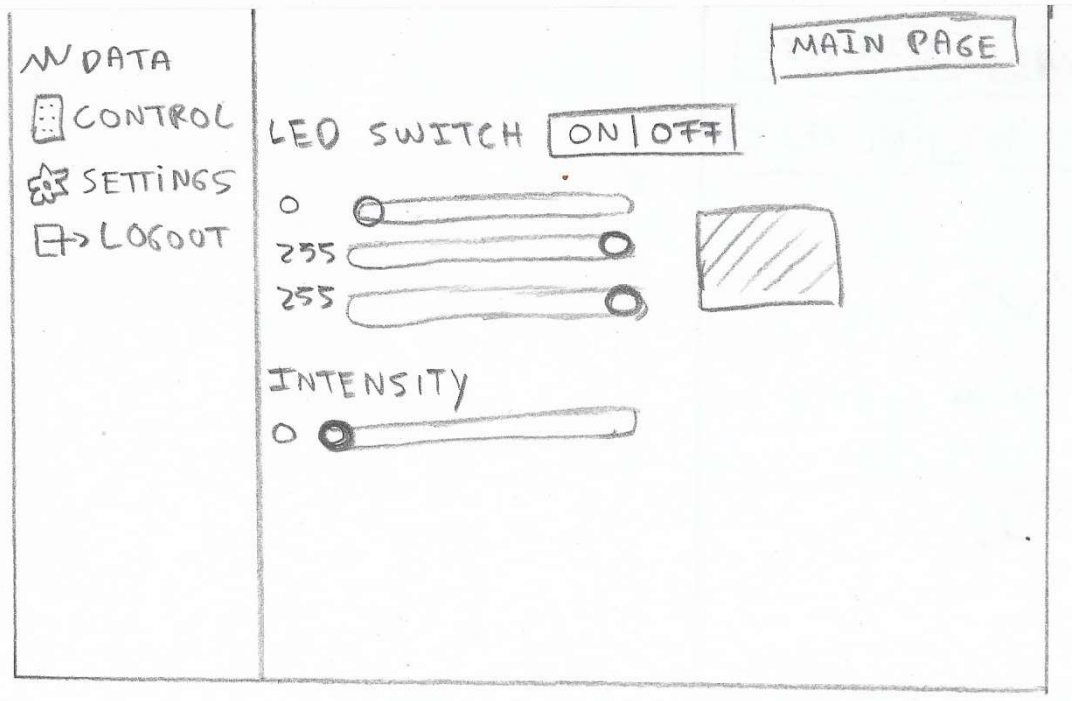


Figura 78. Vista de controlo de um nó da aplicação web protótipo baixa fidelidade.

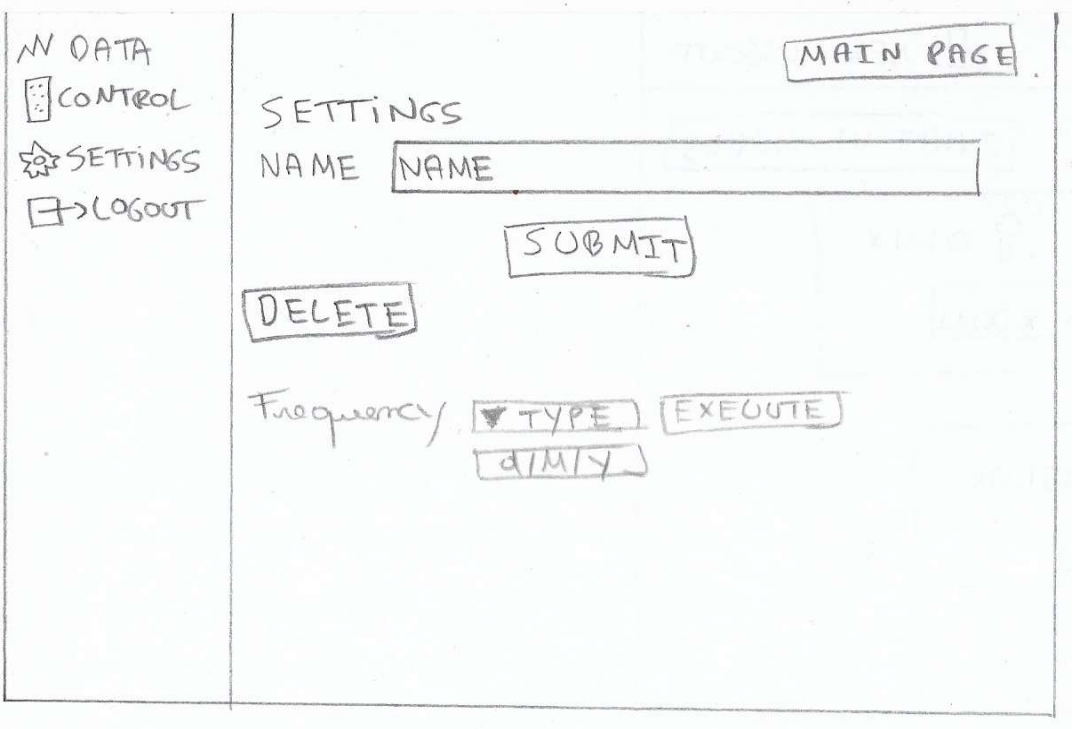


Figura 79. Vista de settings de um nó da aplicação web protótipo baixa fidelidade.

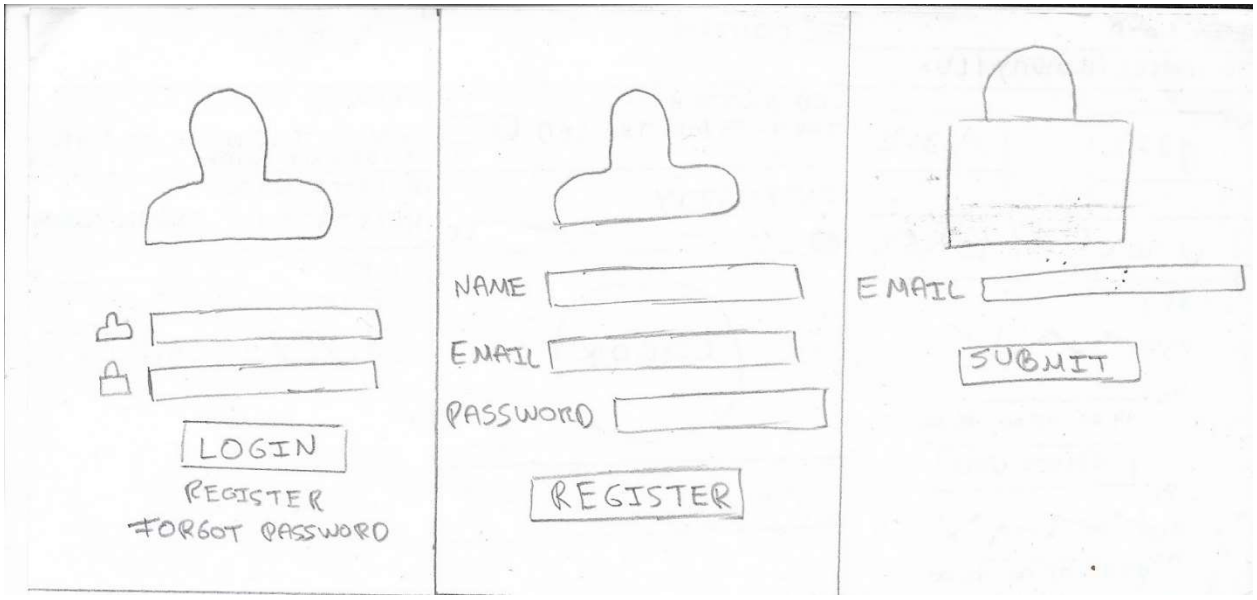


Figura 80. Vista de login,registo e reset password da aplicação móvel protótipo de baixa fidelidade.

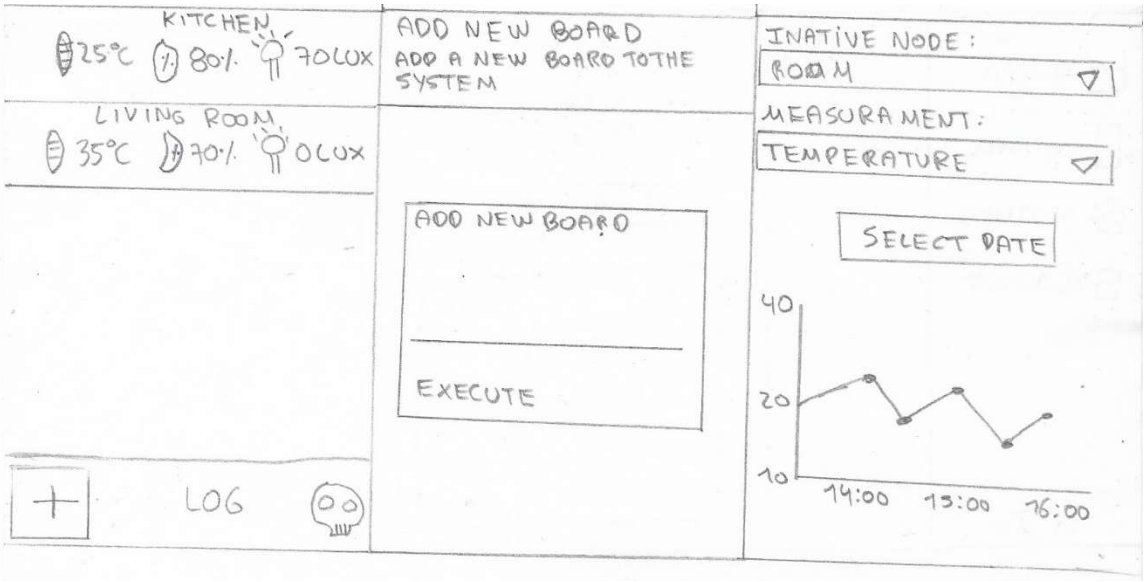


Figura 81. Vista da listagem de nós, de adicionamento de novo nó e visualização de dados de nós inativos na aplicação móvel protótipo de baixa fidelidade.

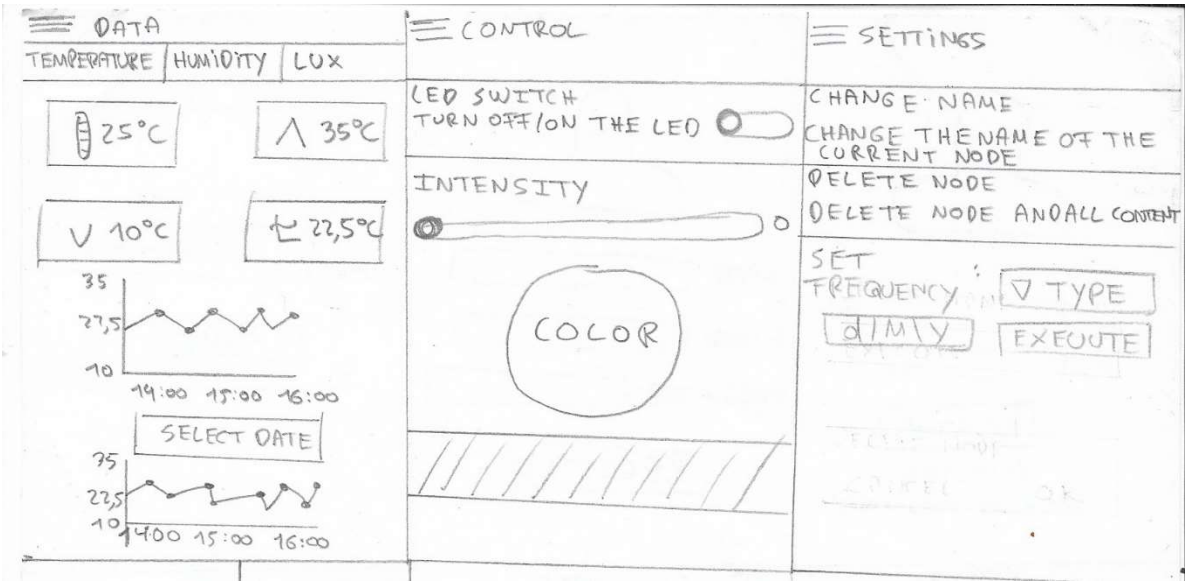
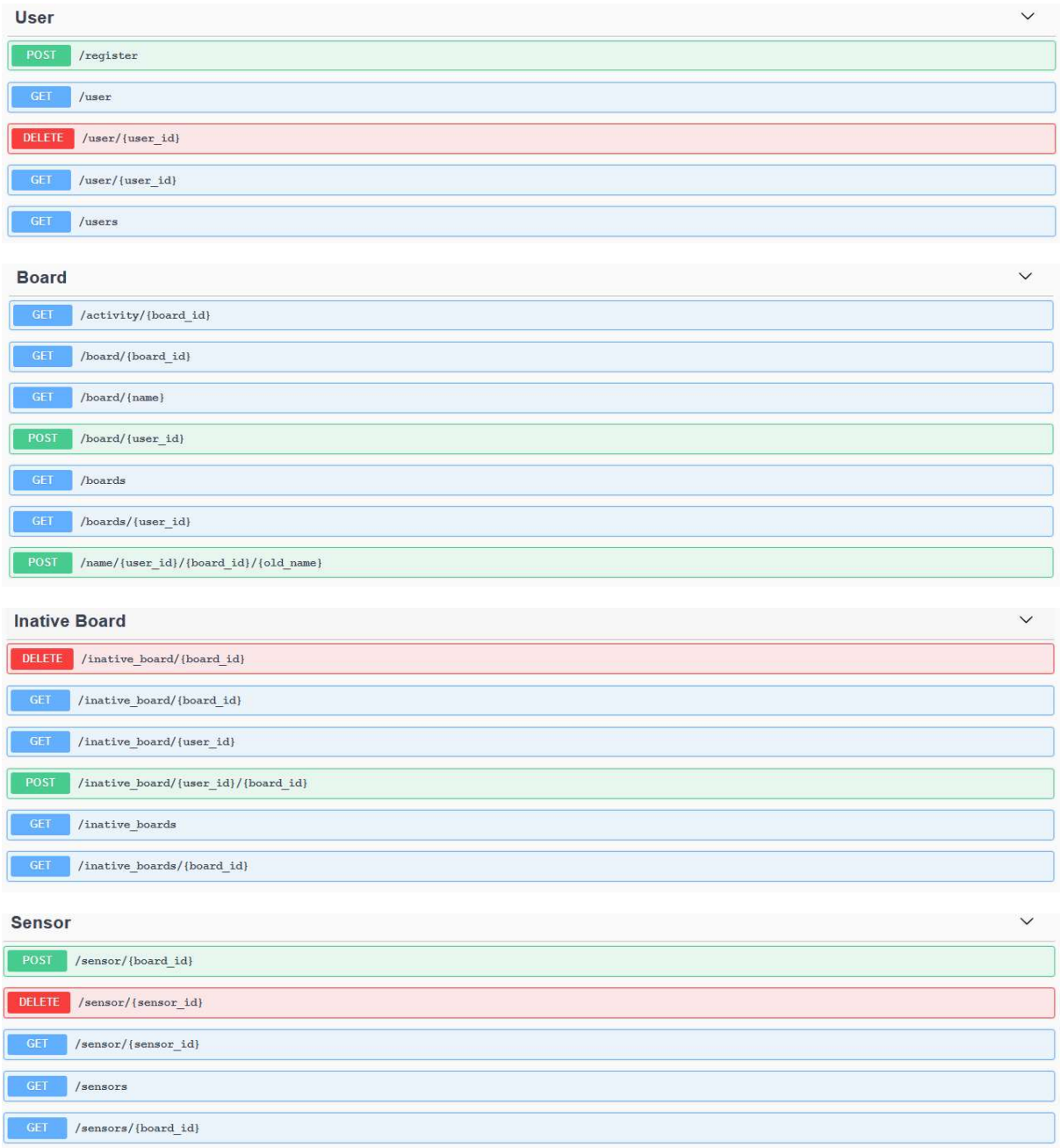


Figura 82. Vista dos dados de um nó, do controlo de um nó e dos settings de um nó ná aplicação móvel protótipo de baixa fidelidade.

Figura 83. Vista da sidebar e logs da aplicação móvel protótipo de baixa fidelidade.

Anexo B – API do servidor central



Reading

GET	/avg/{type_read}/{board_id}/{name}
GET	/hist/{type_read}/{board_id}/{name}/{date}
GET	/last12/{type_read}/{board_id}/{name}
GET	/live/{type_read}/{board_id}
GET	/max/{type_read}/{board_id}/{name}
GET	/min/{type_read}/{board_id}/{name}
DELETE	/reading/{reading_id}
GET	/reading/{reading_id}
GET	/readings
GET	/readings/{sensor_id}
GET	/readings/{sensor_id}/{type_read}

Actuator

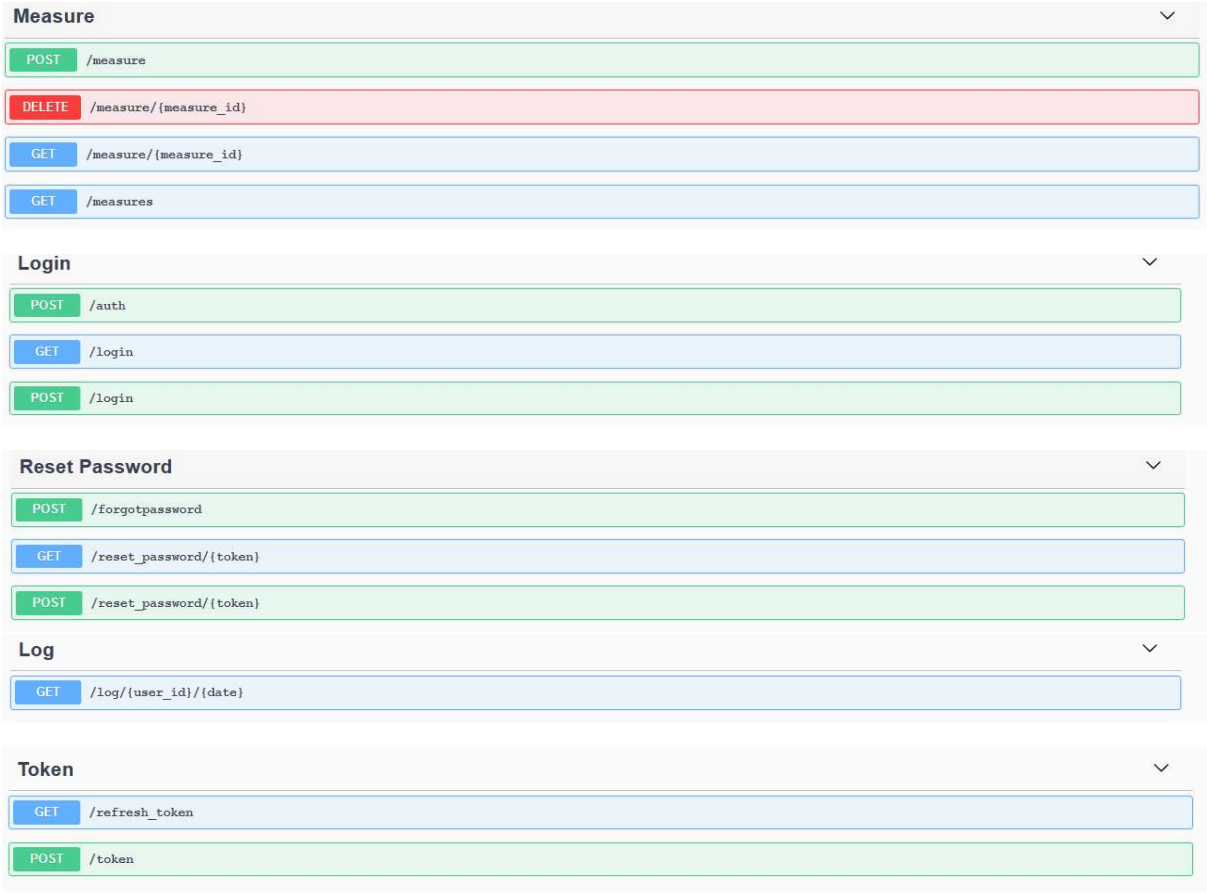
DELETE	/actuator/{actuator_id}
GET	/actuator/{actuator_id}
POST	/actuator/{board_id}
GET	/actuators
GET	/actuators/{board_id}
GET	/intensity/{board_id}
GET	/ledswitchsets/{board_id}
GET	/rgb/{board_id}

Action

DELETE	/action/{action_id}
GET	/action/{action_id}
GET	/action/{actuator_id}
GET	/actions
POST	/frequency/{type_measure}/{board_id}
POST	/intensity/{board_id}
POST	/ledswitch/{board_id}
POST	/rgb/{board_id}
POST	/rgb/{color}/{board_id}

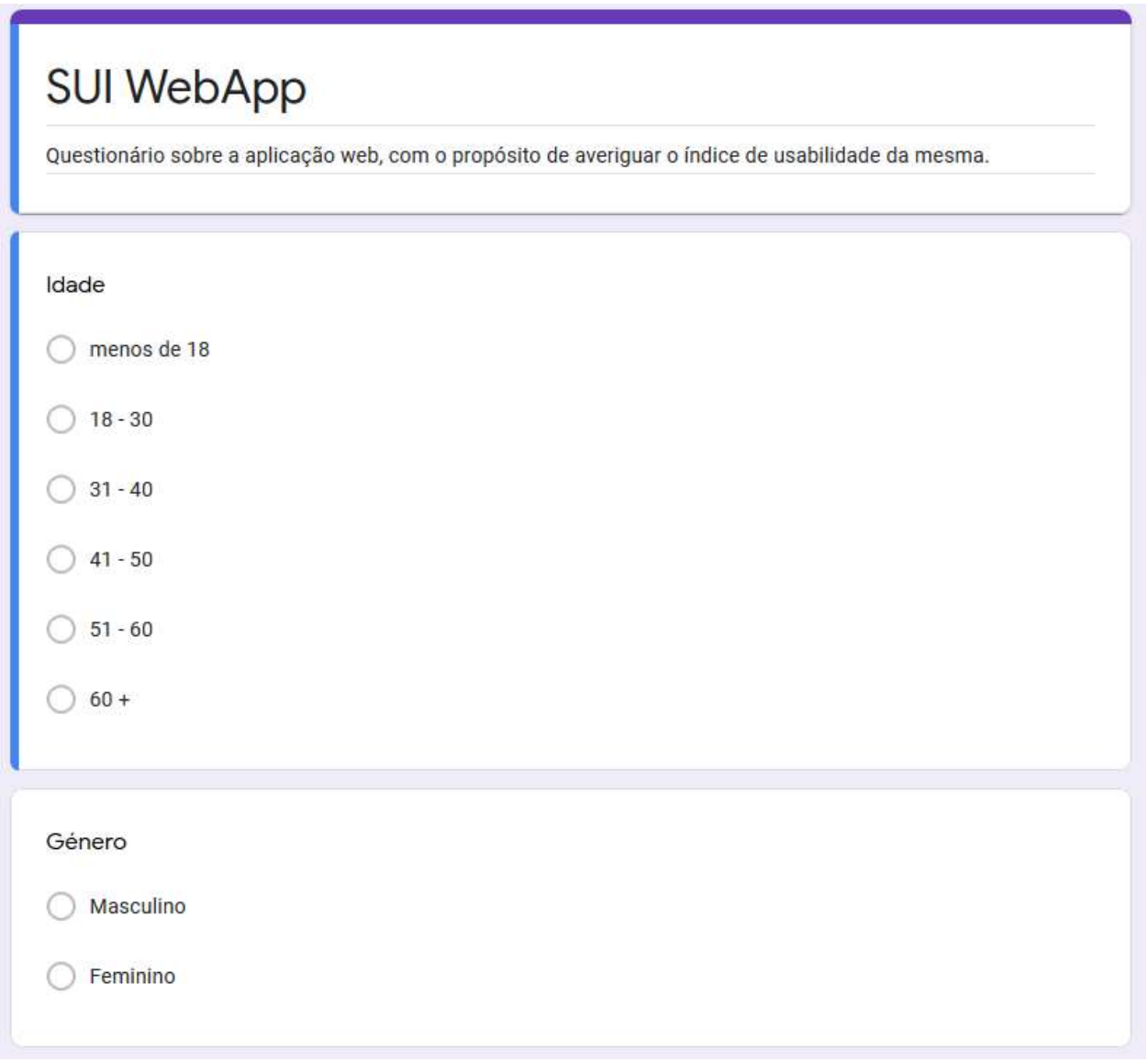
Unit

GET	/unit/{measure_id}
POST	/unit/{measure_id}
DELETE	/unit/{unit_id}
GET	/unit/{unit_id}
GET	/units


Measure
POST /measure
DELETE /measure/{measure_id}
GET /measure/{measure_id}
GET /measures
Login
POST /auth
GET /login
POST /login
Reset Password
POST /forgotpassword
GET /reset_password/{token}
POST /reset_password/{token}
Log
GET /log/{user_id}/{date}
Token
GET /refresh_token
POST /token

Anexo C – Produto com AVS

Anexo D – Questionário SUI aplicação web



SUI WebApp

Questionário sobre a aplicação web, com o propósito de averiguar o índice de usabilidade da mesma.

Idade

- menos de 18
- 18 - 30
- 31 - 40
- 41 - 50
- 51 - 60
- 60 +

Género

- Masculino
- Feminino

Habilitações académicas

- ◯ 1º Ciclo
- ◯ 2º Ciclo
- ◯ 3º Ciclo
- ◯ Secundário
- ◯ Licenciatura
- ◯ Mestrado
- ◯ Doutoramento

Penso que gostaria de utilizar este sistema frequentemente

	1	2	3	4	5	
discordo fortemente	◯	◯	◯	◯	◯	concordo fortemente

Achei o sistema desnecessariamente complexo

	1	2	3	4	5	
discordo fortemente	◯	◯	◯	◯	◯	concordo fortemente

Pensei que o sistema era fácil de usar.

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Penso que precisaria do apoio de uma pessoa técnica para poder utilizar este sistema

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Achei que as várias funções deste sistema estavam bem integradas.

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Pensei que havia demasiada incoerência neste sistema

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Imagino que a maioria das pessoas aprenderia a utilizar este sistema muito rapidamente

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Achei o sistema muito complicado de usar

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Senti-me muito confiante ao utilizar o sistema

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

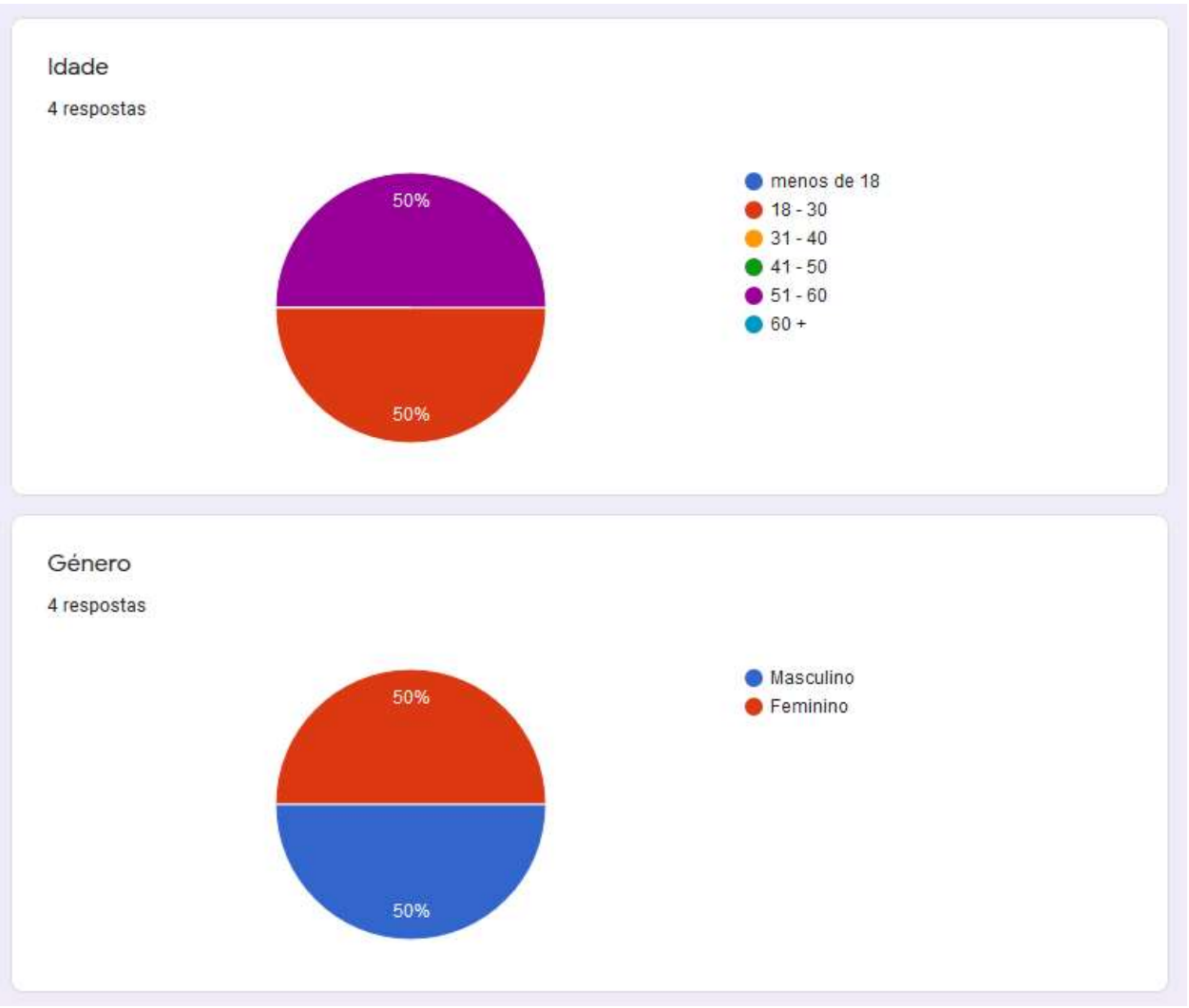
Precisava de aprender muitas coisas antes de poder avançar com este sistema

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Sugestões, alterações, melhorias

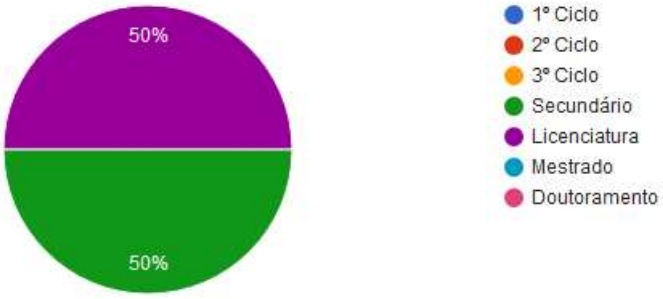
Texto de resposta longa

Anexo E – Respostas Questionário SUI aplicação web



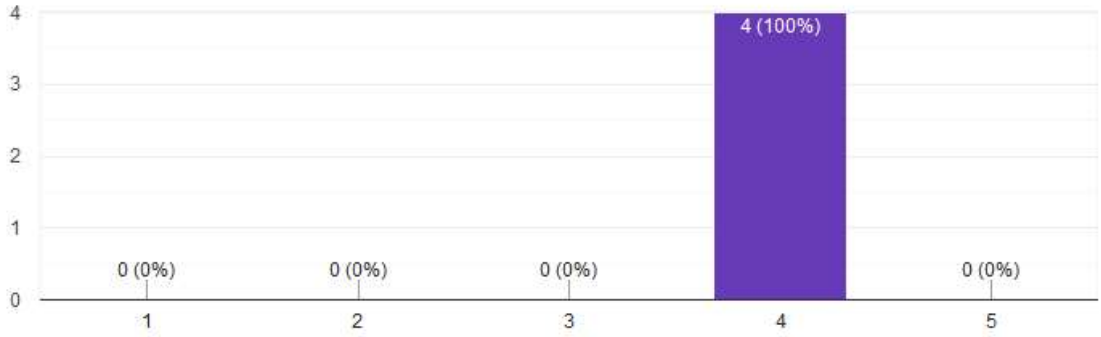
Habilitações académicas

4 respostas



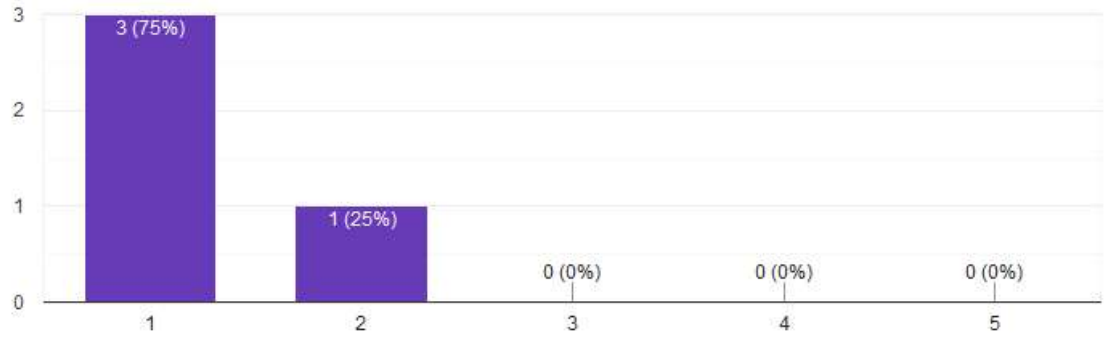
Penso que gostaria de utilizar este sistema frequentemente

4 respostas



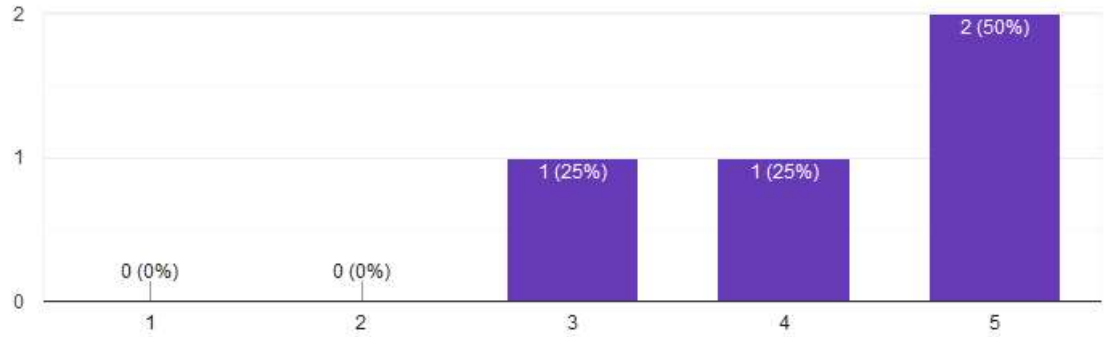
Achei o sistema desnecessariamente complexo

4 respostas



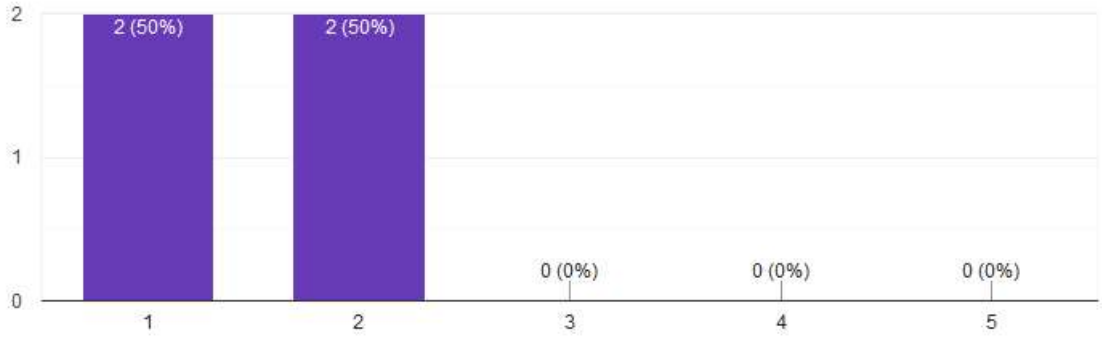
Pensei que o sistema era fácil de usar.

4 respostas



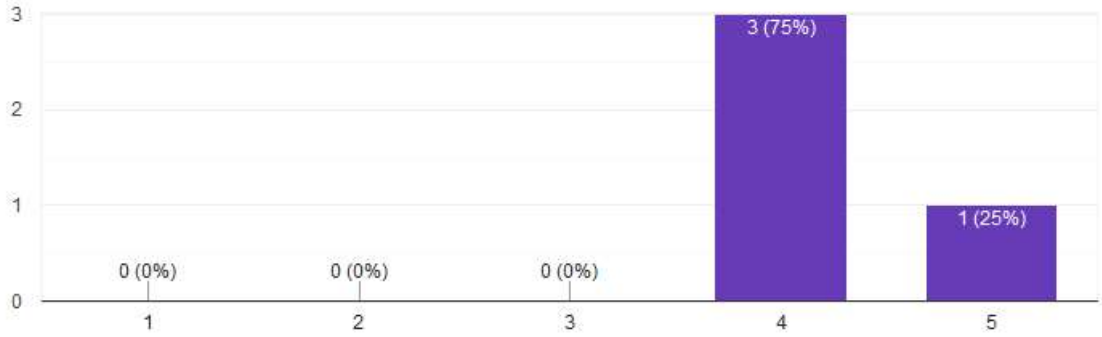
Penso que precisaria do apoio de uma pessoa técnica para poder utilizar este sistema

4 respostas



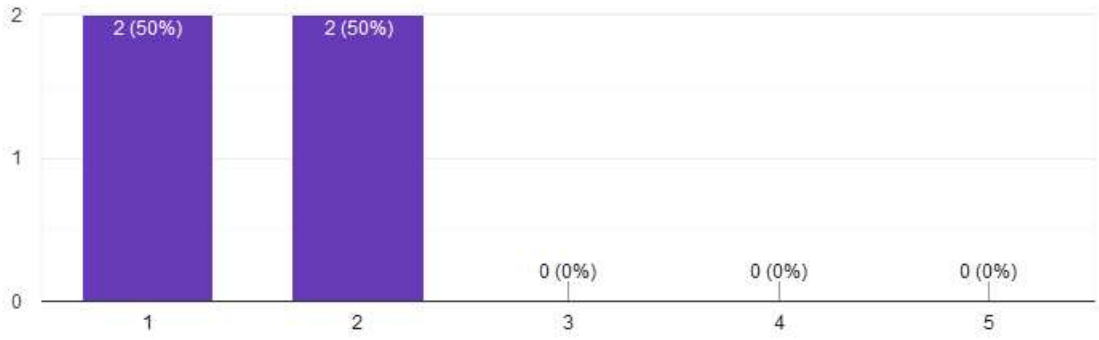
Achei que as várias funções deste sistema estavam bem integradas.

4 respostas



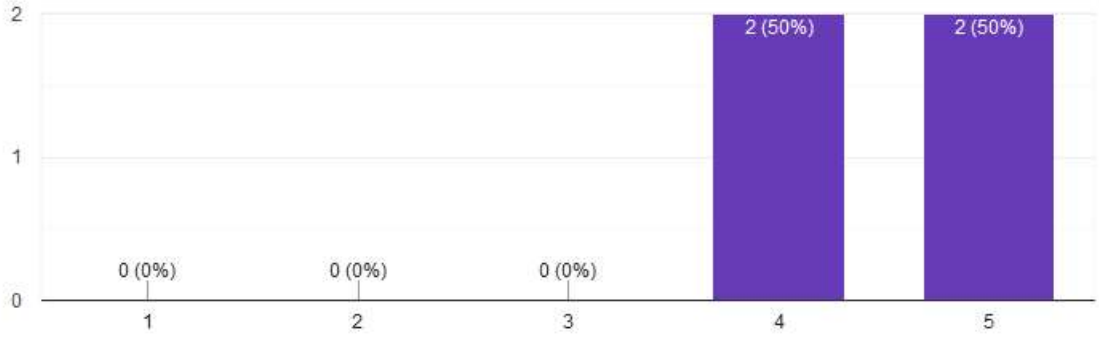
Pensei que havia demasiada incoerência neste sistema

4 respostas



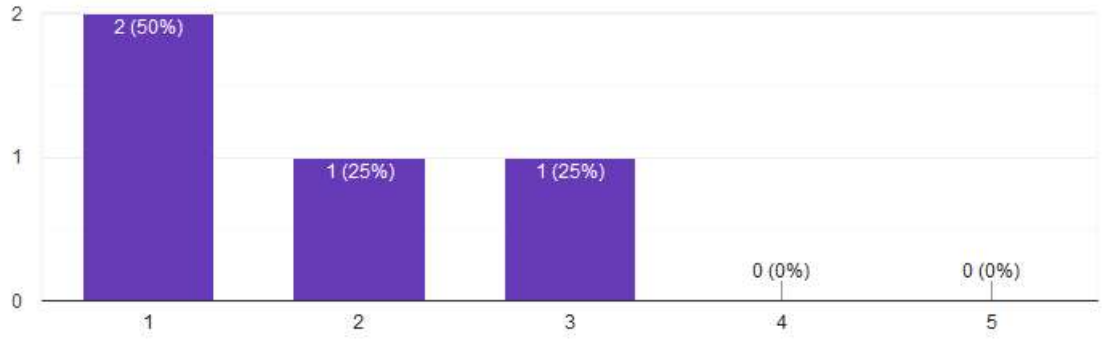
Imagino que a maioria das pessoas aprenderia a utilizar este sistema muito rapidamente

4 respostas



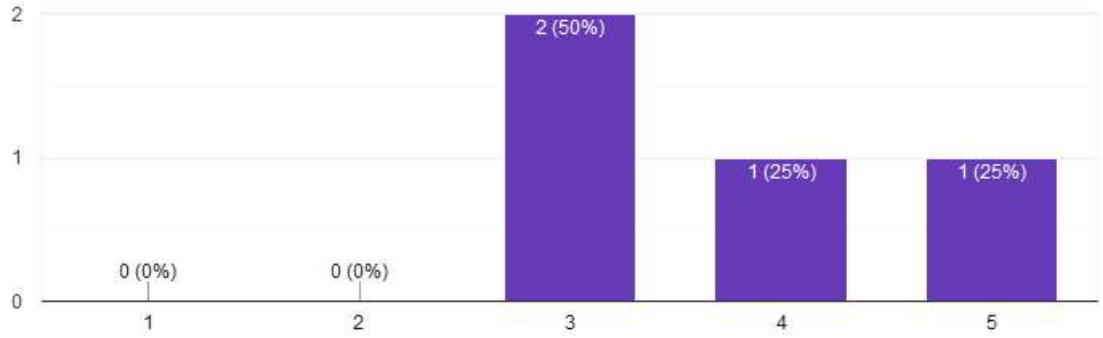
Achei o sistema muito complicado de usar

4 respostas



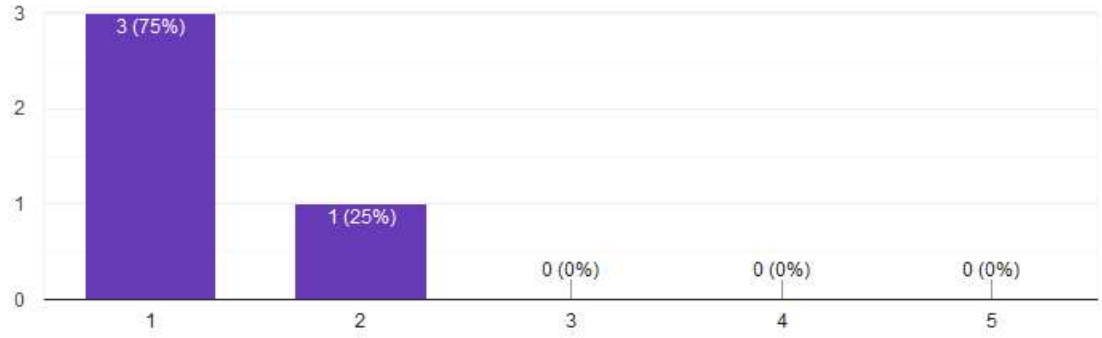
Senti-me muito confiante ao utilizar o sistema

4 respostas



Precisava de aprender muitas coisas antes de poder avançar com este sistema

4 respostas



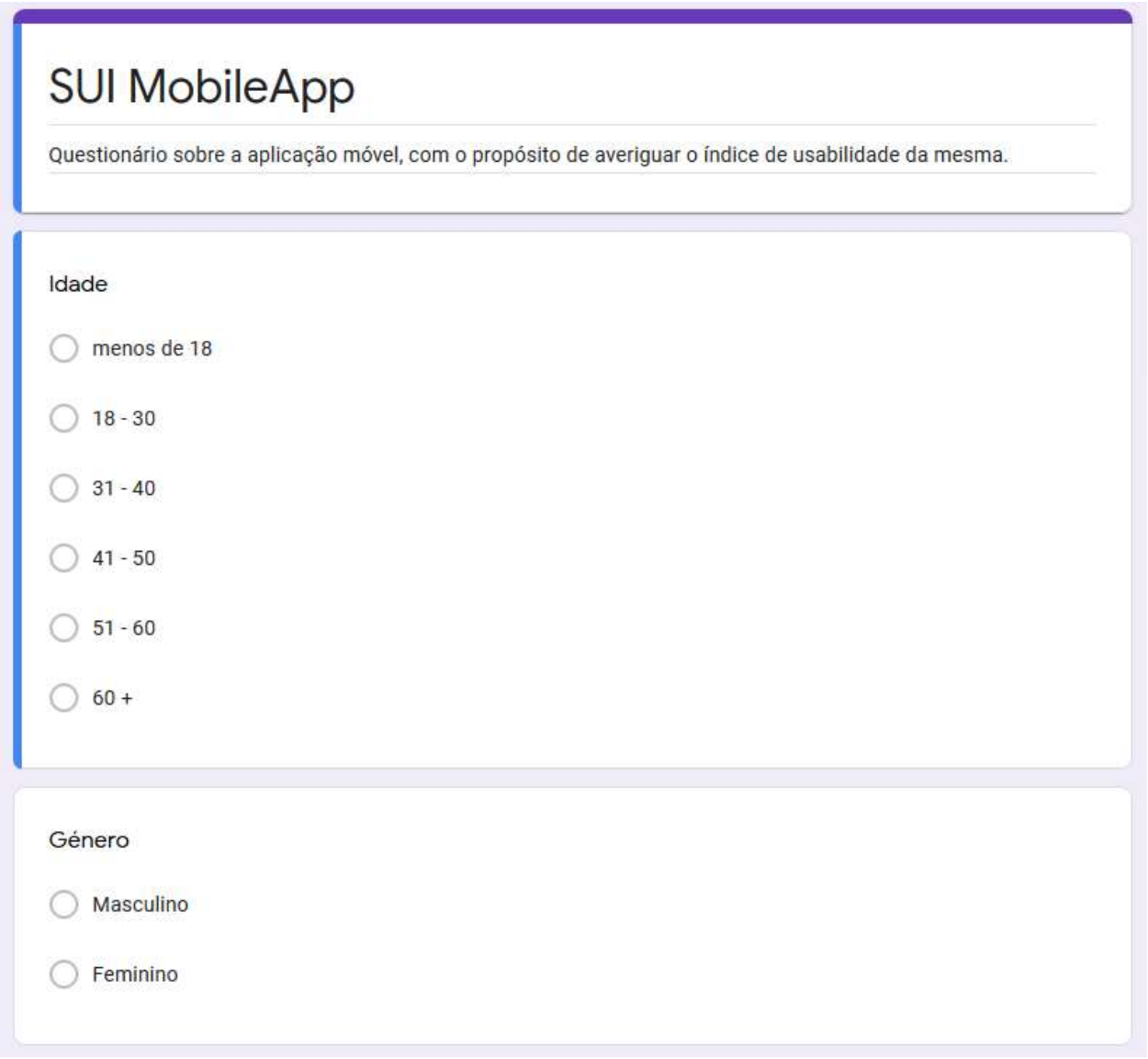
Sugestões, alterações, melhorias

2 respostas

Alterar as cores das setas relacionadas com os valores máximos e mínimos da temperatura para azul e vermelho. As outras cores, da humidade e da luminosidade podem manter-se. Na parte dos calendários deveria ter uma legenda a dizer o seu propósito.

Icons da máxima temperatura, humidade, luz com cores pouco intuitivas. Barra lateral devia ter um botão para voltar para atrás. O botão "Go back to main page" encontra-se no lado oposto de todas as outras funcionalidades disponiveis.
Ao adicionar um novo nodo ao sistema o botao "list" a direita não é descritivo o suficiente de qual é a funcionalidade, o botão encontra-se a direita e nao à esquerda. A pagina dos "logs" não apresenta logs nenhums ao carregar a pagina. Podia pelo menos aparecer os logs mais recentes. Header bar para fazer logout e visualizar o nome do utilizador devia estar presente nas outras páginas. O botão execute nos nodos parece que se refere ao grafico de cima.

Anexo F – Questionário SUI aplicação móvel



SUI MobileApp

Questionário sobre a aplicação móvel, com o propósito de averiguar o índice de usabilidade da mesma.

Idade

- menos de 18
- 18 - 30
- 31 - 40
- 41 - 50
- 51 - 60
- 60 +

Género

- Masculino
- Feminino

Habilitações académicas

- ○ 1º Ciclo
- ○ 2º Ciclo
- ○ 3º Ciclo
- ○ Secundário
- ○ Licenciatura
- ○ Mestrado
- ○ Doutoramento

Penso que gostaria de utilizar este sistema frequentemente

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Achei o sistema desnecessariamente complexo

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Pensei que o sistema era fácil de usar.

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Penso que precisaria do apoio de uma pessoa técnica para poder utilizar este sistema

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Achei que as várias funções deste sistema estavam bem integradas.

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Pensei que havia demasiada incoerência neste sistema

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Imagino que a maioria das pessoas aprenderia a utilizar este sistema muito rapidamente

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Achei o sistema muito complicado de usar

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Senti-me muito confiante ao utilizar o sistema

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Precisava de aprender muitas coisas antes de poder avançar com este sistema

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

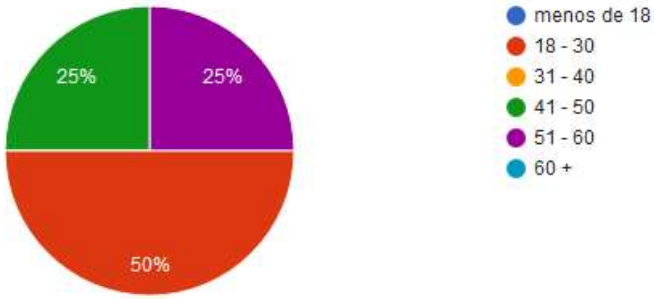
Sugestões, alterações, melhorias

Texto de resposta longa

Anexo G – Respostas Questionário SUI aplicação móvel

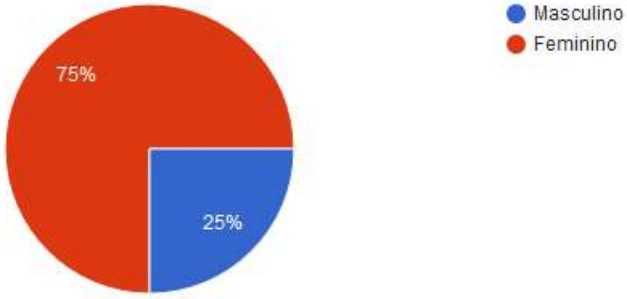
Idade

4 respostas



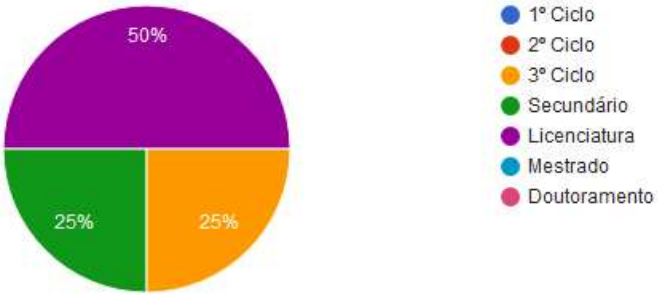
Género

4 respostas



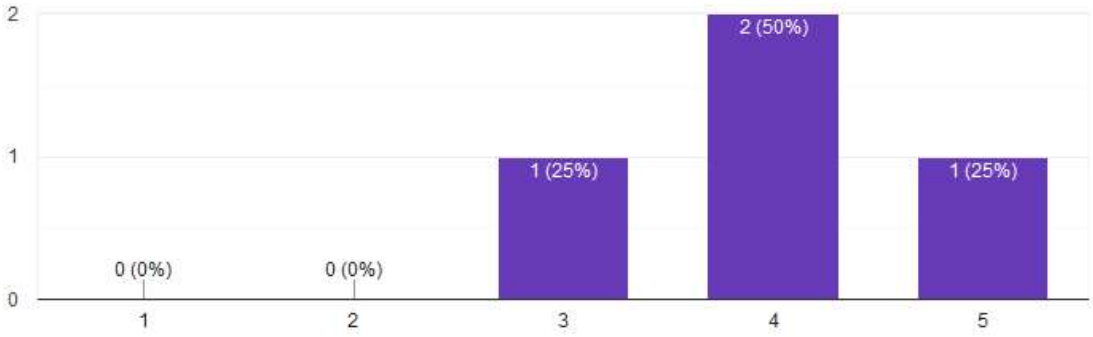
Habilitações académicas

4 respostas



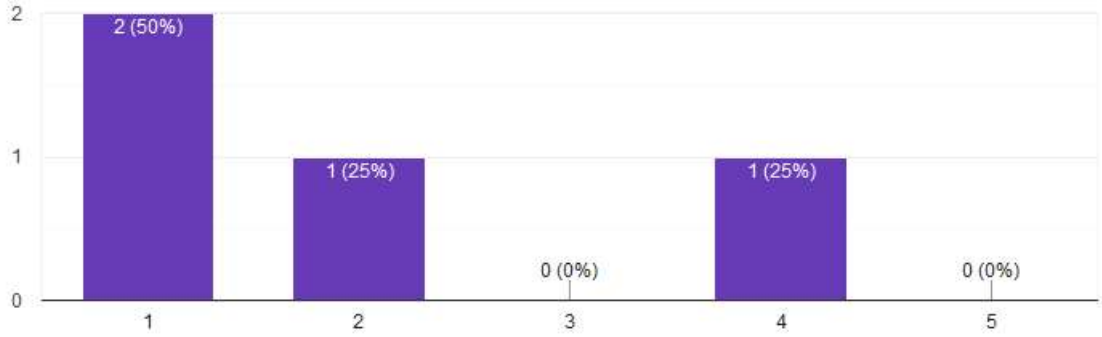
Penso que gostaria de utilizar este sistema frequentemente

4 respostas



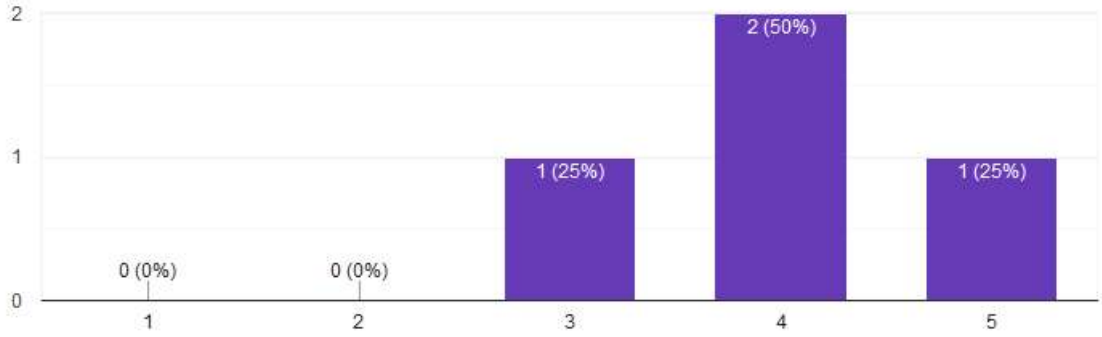
Achei o sistema desnecessariamente complexo

4 respostas



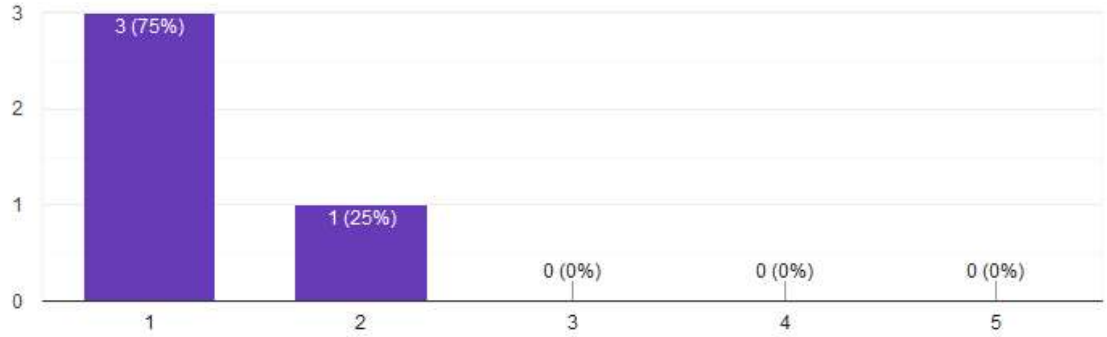
Pensei que o sistema era fácil de usar.

4 respostas



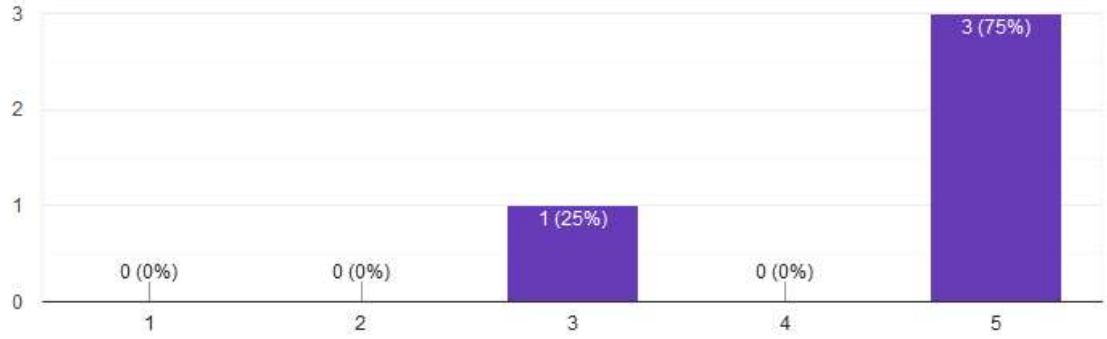
Penso que precisaria do apoio de uma pessoa técnica para poder utilizar este sistema

4 respostas



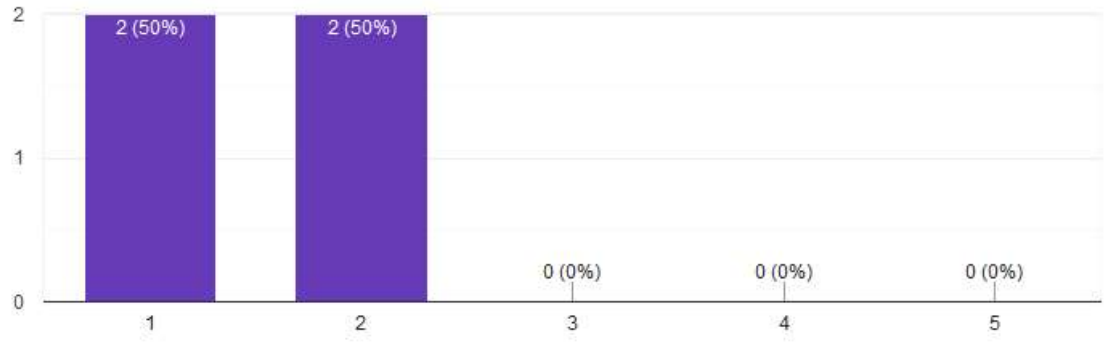
Achei que as várias funções deste sistema estavam bem integradas.

4 respostas



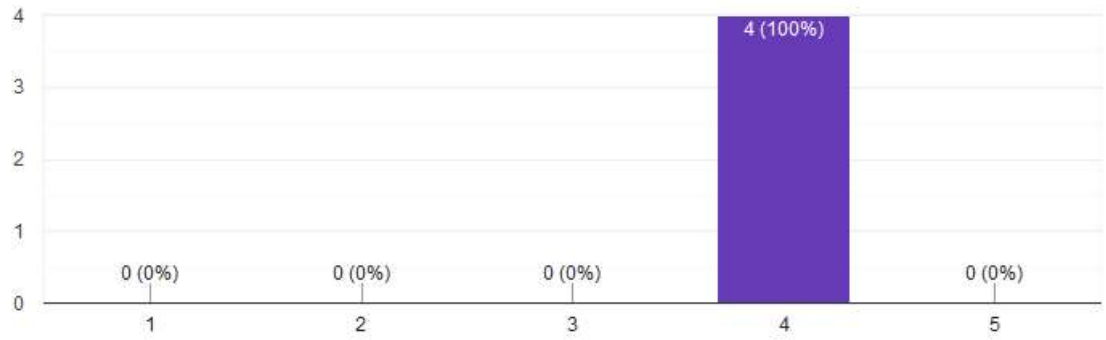
Pensei que havia demasiada incoerência neste sistema

4 respostas



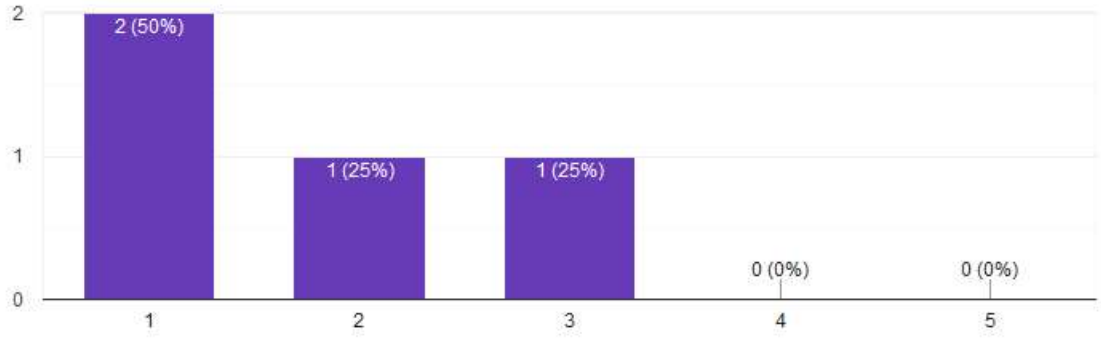
Imagino que a maioria das pessoas aprenderia a utilizar este sistema muito rapidamente

4 respostas



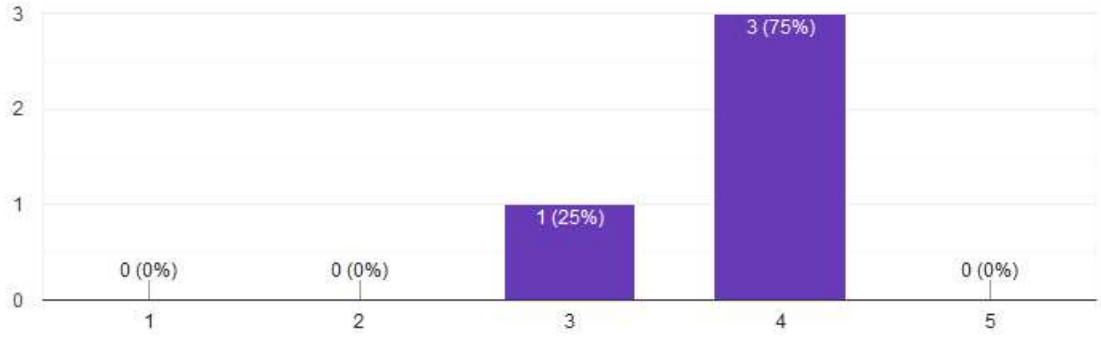
Achei o sistema muito complicado de usar

4 respostas



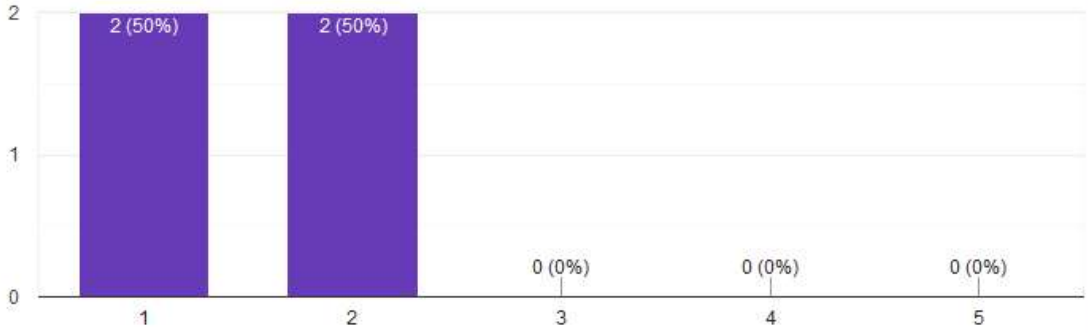
Senti-me muito confiante ao utilizar o sistema

4 respostas



Precisava de aprender muitas coisas antes de poder avançar com este sistema

4 respostas



Sugestões, alterações, melhorias

1 resposta

Login ok, No criar um novo utilizador a dialog box não se ve. Na pagina principal não é intuitivo onde posso visualizar os nodos inativos. Os logs mais recentes deviam aparecer primeiro. Acho que devia ser mais facil de deletar um nó e mudar o seu nome.

Anexo H – Questionário SUI da skill da Alexa

SUI Alexa Skill

Questionário sobre a skill da Alexa, com o propósito de averiguar o índice de usabilidade da mesma.

Idade

- menos de 18
- 18 - 30
- 31 - 40
- 41 - 50
- 51 - 60
- 60 +

Género

- Masculino
- Feminino

Habilitações académicas

- ○ 1º Ciclo
- ○ 2º Ciclo
- ○ 3º Ciclo
- ○ Secundário
- ○ Licenciatura
- ○ Mestrado
- ○ Doutoramento

Usa assistentes virtuais com frequência

	1	2	3	4	5	
Nunca	○	○	○	○	○	Sempre

Se respondeu 1 na pergunta anterior salte para a próxima questão. Que assistentes virtuais já usou?

- [ ] Google Assistant
- [ ] Alexa
- [ ] Siri
- [ ] Bixby
- [ ] Cortana
- [ ] Outra

Penso que gostaria de utilizar este sistema frequentemente

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Achei o sistema desnecessariamente complexo

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Pensei que o sistema era fácil de usar.

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Penso que precisaria do apoio de uma pessoa técnica para poder utilizar este sistema

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Achei que as várias funções deste sistema estavam bem integradas.

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Pensei que havia demasiada incoerência neste sistema

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Imagino que a maioria das pessoas aprenderia a utilizar este sistema muito rapidamente

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Achei o sistema muito complicado de usar

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Senti-me muito confiante ao utilizar o sistema

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

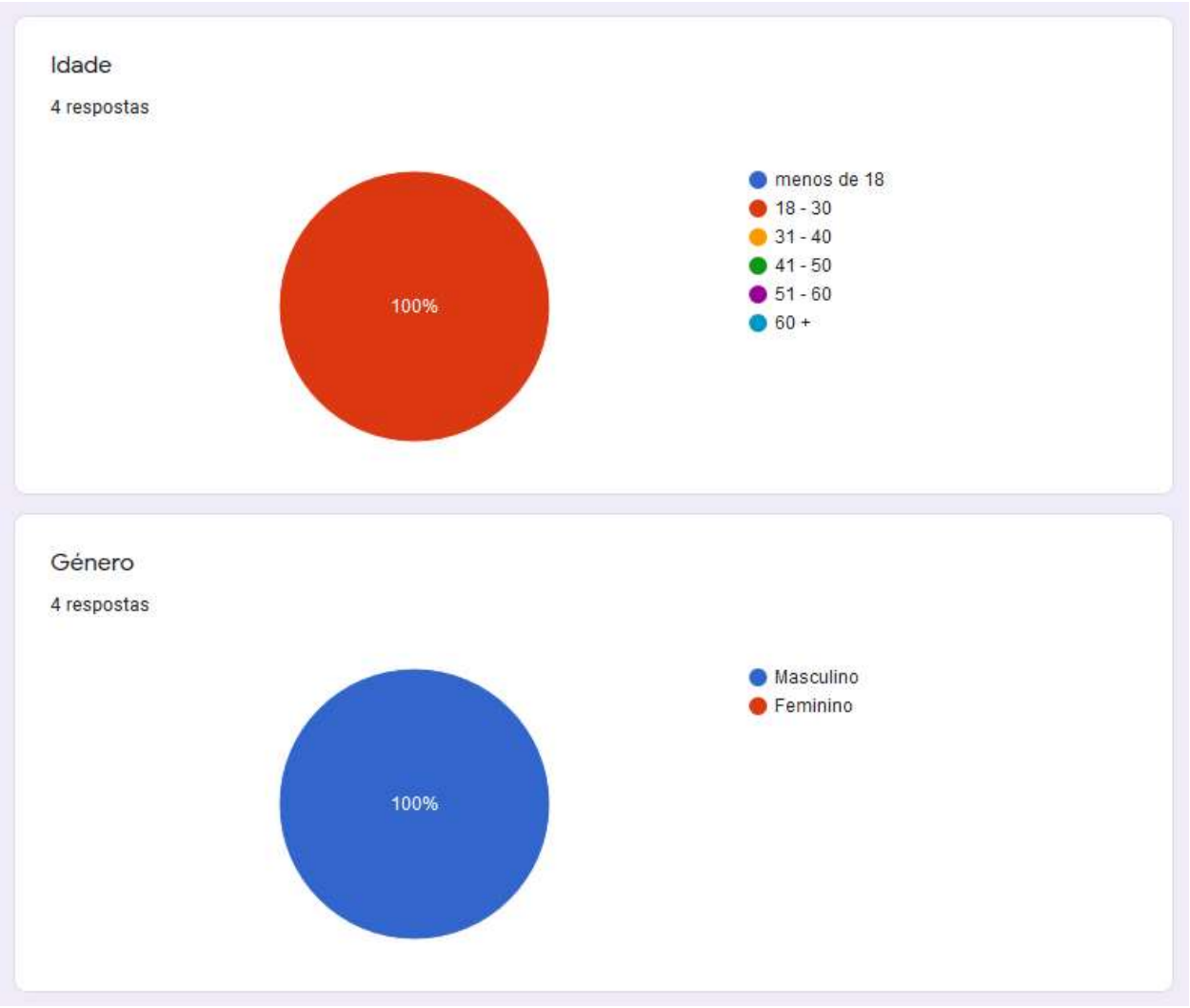
Precisava de aprender muitas coisas antes de poder avançar com este sistema

	1	2	3	4	5	
discordo fortemente	○	○	○	○	○	concordo fortemente

Sugestões, alterações, melhorias

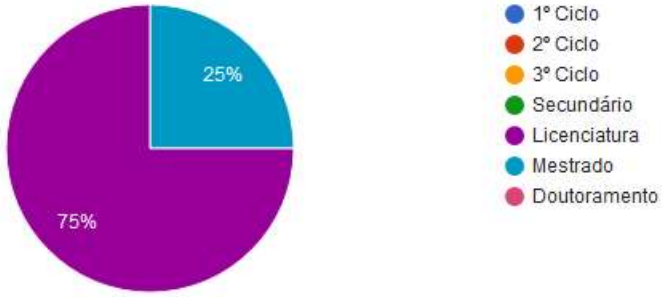
Texto de resposta longa

Anexo I – Respostas Questionário SUI da skill da Alexa



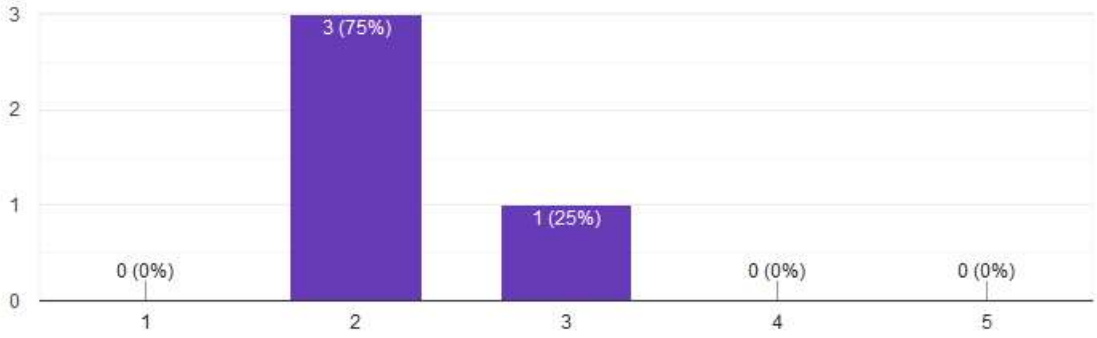
Habilitações académicas

4 respostas



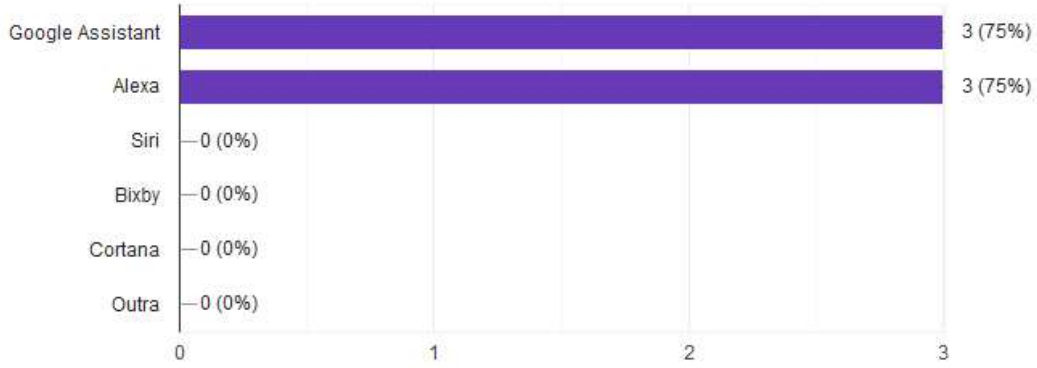
Usa assistentes virtuais com frequência

4 respostas



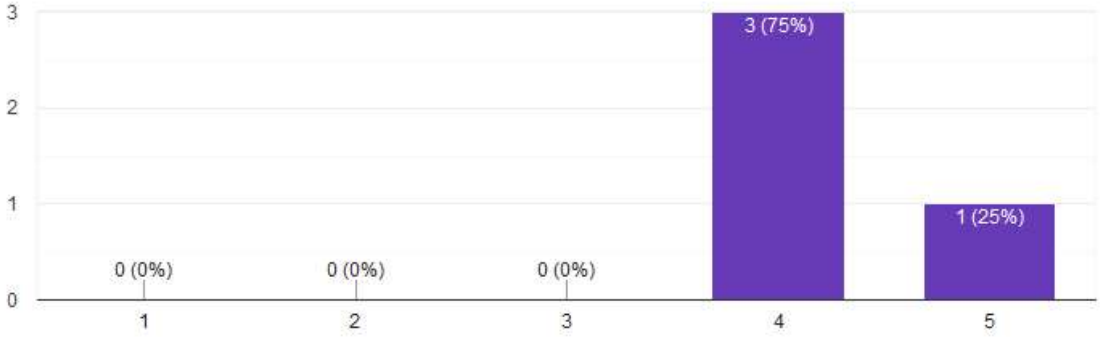
Se respondeu 1 na pergunta anterior salte para a próxima questão. Que assistentes virtuais já usou?

4 respostas



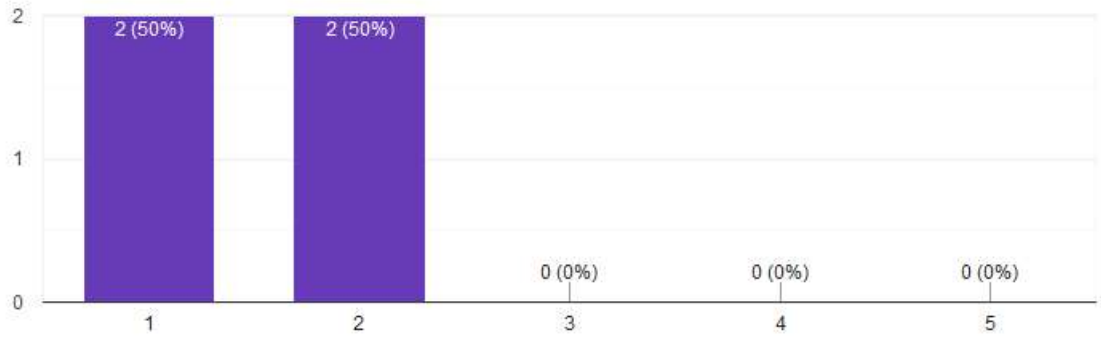
Penso que gostaria de utilizar este sistema frequentemente

4 respostas



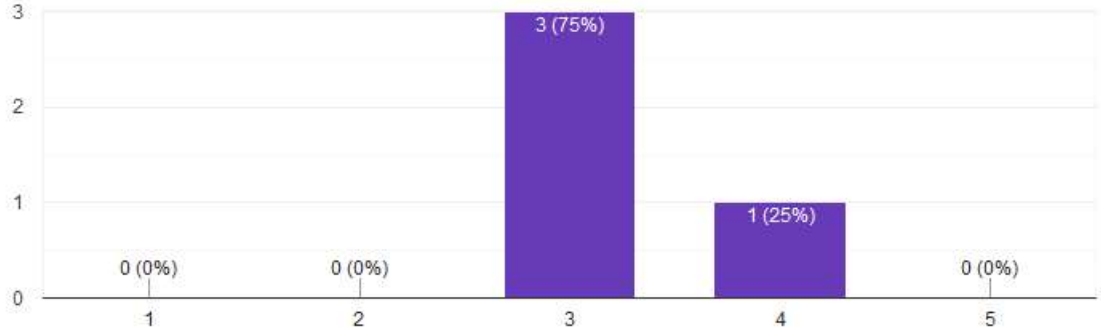
Achei o sistema desnecessariamente complexo

4 respostas



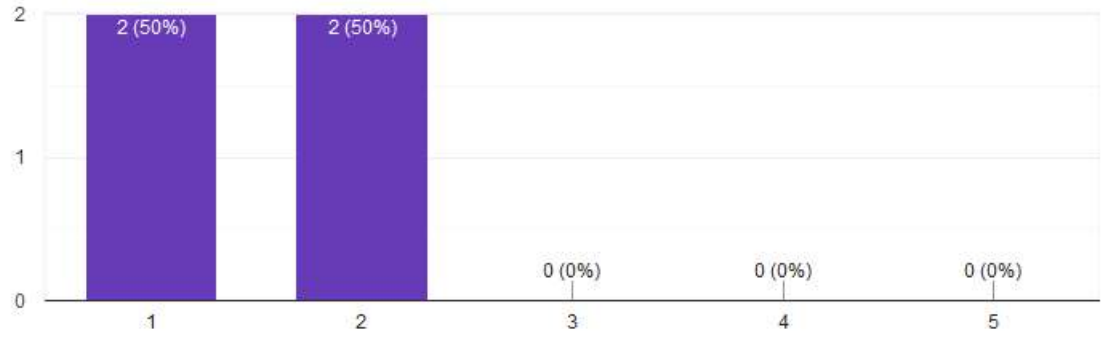
Pensei que o sistema era fácil de usar.

4 respostas



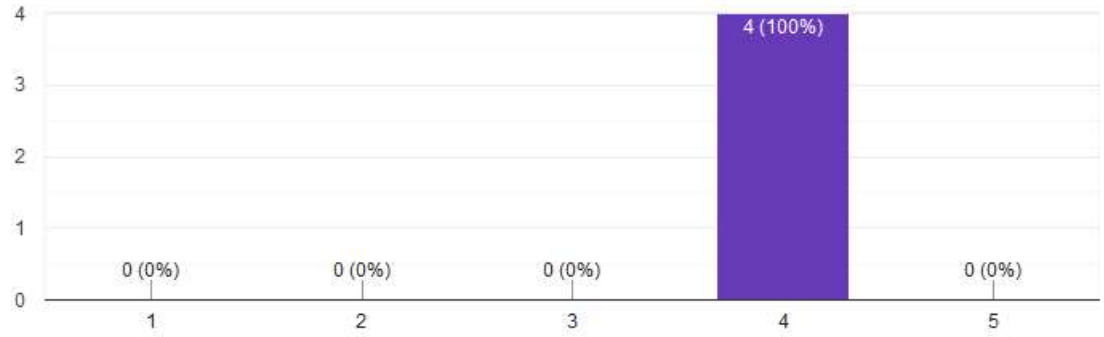
Penso que precisaria do apoio de uma pessoa técnica para poder utilizar este sistema

4 respostas



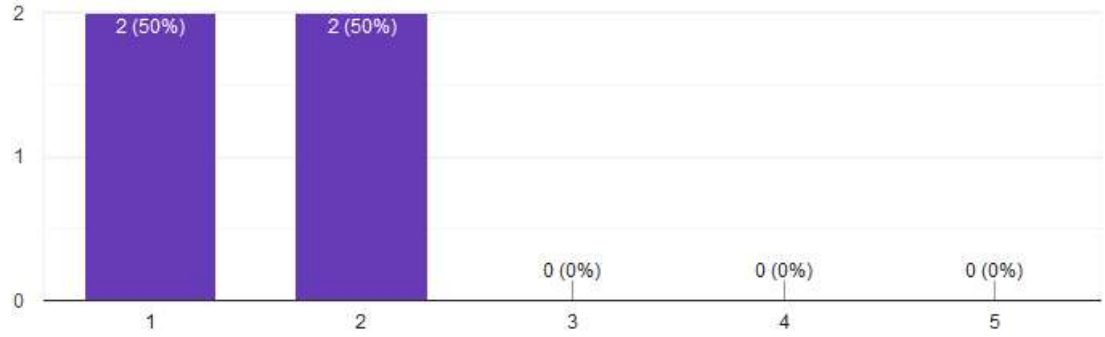
Achei que as várias funções deste sistema estavam bem integradas.

4 respostas



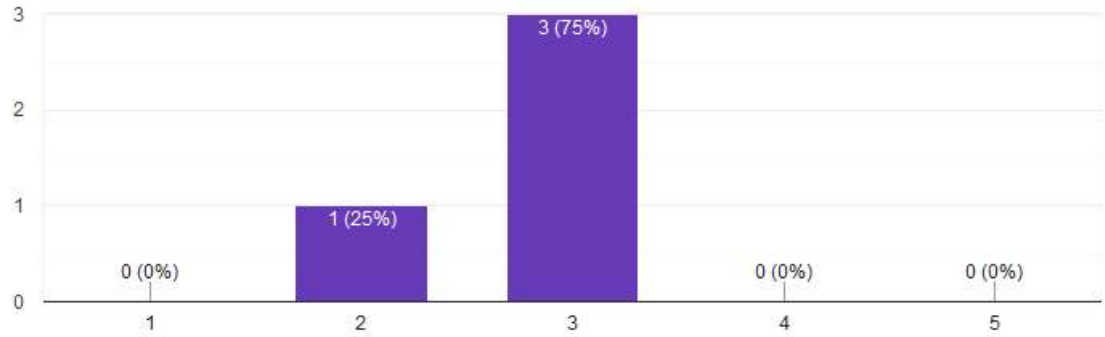
Pensei que havia demasiada incoerência neste sistema

4 respostas



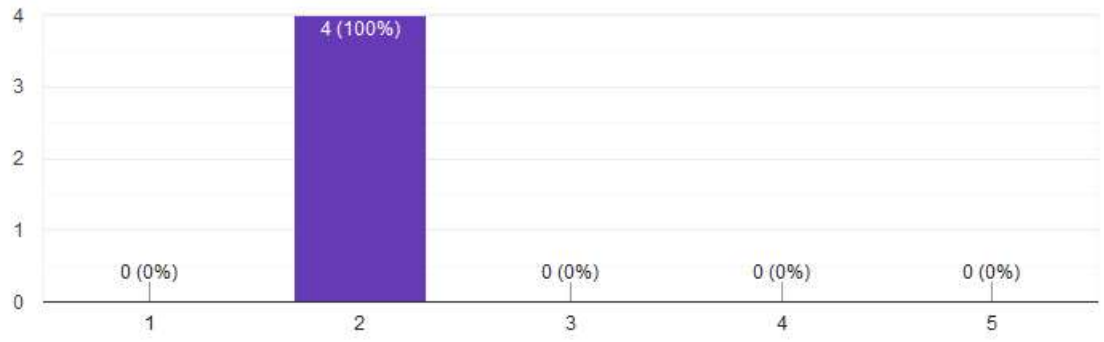
Imagino que a maioria das pessoas aprenderia a utilizar este sistema muito rapidamente

4 respostas



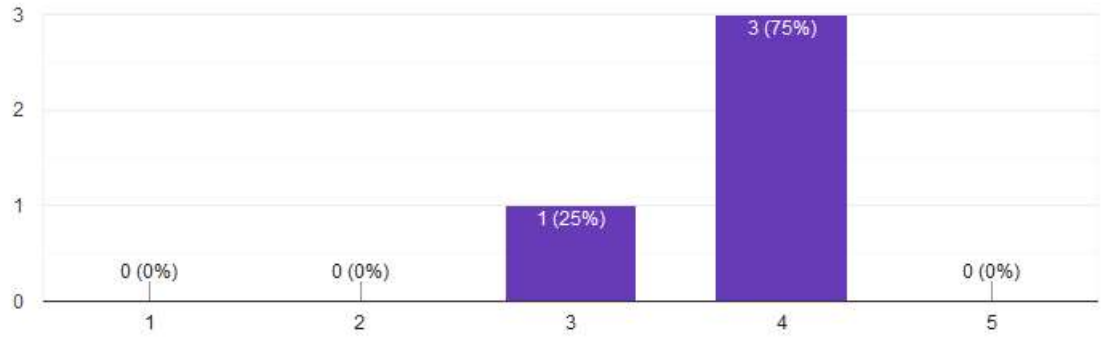
Achei o sistema muito complicado de usar

4 respostas



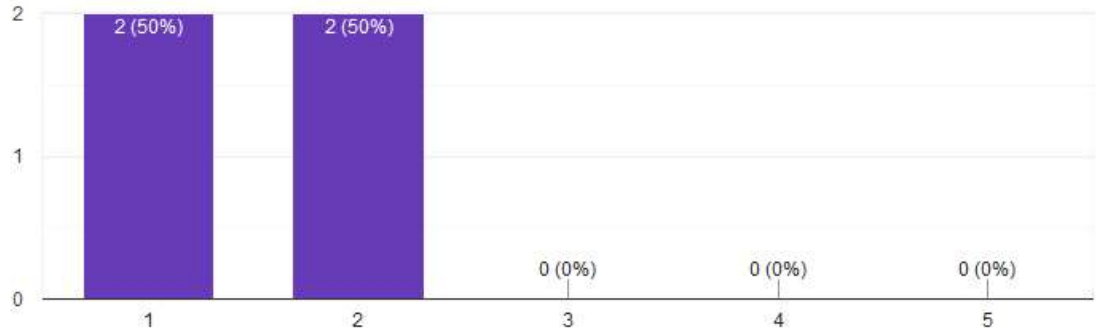
Senti-me muito confiante ao utilizar o sistema

4 respostas



Precisava de aprender muitas coisas antes de poder avançar com este sistema

4 respostas



Sugestões, alterações, melhorias

1 resposta

Era bom se tivesse uma versão em português. A ajuda fornecida devia ser mais clara.