OpenLockDoor: Sistema de Controle de Acesso de salas com base em Sensores RFID

Lucas F. A. Silva, Bruno S. Leite, Lucas S. Vaz, Talita L. Lima e Vinícius S. Amaral
Instituto de Ciência e Tecnologia - Universidade Federal de São Paulo – UNIFESP
12247-014, São José dos Campos, SP – Brazil
Email: {bruno.sampaio08, lucas.vaz, ludmila.lima, vsamaral, **e.lucas**}@unifesp.br

***Resumo*—Atualmente tem-se uma demanda muito grande por sistemas de segurança, como cercas elétricas, centrais de alarmes e fechaduras de portas eletrônicas, as quais atualmente contam com senhas e até mesmo leituras de impressões digitais. Porém esses sistemas dificilmente tem interfaces intuitivas e seu uso é limitado a configurações estáticas, as quais devem ser programadas no seu primeiro uso ou através de códigos secretos especiais de *hardware*. Nesse trabalho foi proposto e desenvolvido um sistema completo de internet das coisas para o controle de acesso a salas com uso de chaves RFID. Ele foi construído de um protótipo de hardware, que verifica a relação de agendamento de salas, e por um aplicativo para celulares *Android* para a administração de um banco de dados NoSQL com a relação de agendamentos de salas.**

I. Introdução

Atualmente, diversos tipos de sistemas estão se voltando para a plataforma de internet das coisas (IOT), devido a sua grande flexibilidade de comunicação entre diferentes dispositivos e tecnologias. Essa comunicação como o nome se refere, é feita, por exemplo, através de uma rede local entre dispositivos de uma casa, ou, através da internet. Esses sistemas podem ser constituídos de microcontroladores, com baixo poder de processamento e um baixo custo, ou sistemas microprocessados, com um poder de processamento maior com um alto custo.

Uma dentre as categoria de equipamentos que vem se voltando para a tecnologia IOT são, por exemplo, os sistemas de seguranças patrimoniais ou residenciais, os quais, a priori, eram verificados através de controladores humano e acionando apenas alarmes locais. O uso das tecnologias de IOT nestes sistemas garantem um nível maior de segurança e de completude, pois, como são conectados através da rede de internet, as notificações tem alcance global e não são mais restritos a um distância pequena.

Dentre os sistemas de segurança, as fechaduras eletrônicas, que são muito usados em salas comerciais e empresariais, estão cada vez mais agregando novas tecnologias ao seu funcionamento, como liberando acessos através da apresentação de senhas, de chaves RFID e até mesmo com a detecção de impressões digitais. No entanto, estes dispositivos normalmente não mantém históricos e também não realizam um controle dos acessos, de modo que qualquer pessoa com uma chave válida pode acessar a sala a qualquer momento. Além desses fatores, eles normalmente apresentam interfaces simples e de baixo nível, com a mudança de configurações acontecendo através da apresentação de chaves e/ou padrões secretos.

O trabalho apresentado nesse artigo tem por objetivo o desenvolvimento de um sistema de controle de acesso a salas, voltado, em um primeiro momento, as salas da universidade Federal de São Paulo - UNIFESP, campus São José dos Campos. Foi planejado que o acesso seria controlado através de uma chave RFID única para cada um dos professores e controladores de acesso da universidade. Os controladores de acesso estariam em posse de chaves classificadas como mestre, as quais abrem qualquer porta a qualquer momento do dia, e os professores com chaves classificadas como comuns, as quais só seriam liberados acessos em determinados horários e dias da semana. Ele foi dividido em duas partes, sendo elas: O desenvolvimento do hardware, composto por um sistema micro-controlado com acesso a internet, um sensor de RFID e um servomotor, simulando uma fechadura eletrônica; E uma aplicação para celulares Android, voltada para a manutenção do banco de dados, hospedado em servidor na internet.

As próximas seções estão organizadas da seguinte forma: a Seção II apresenta os materiais usados para o desenvolvimento do trabalho, assim como os métodos adotados. A Seção III contém as etapas de desenvolvimento do protótipo. A seção IV mostra os resultados finais e uma discussão acerca destes e, por fim, na Seção V é apresentado as principais contribuições e as direções futuras do trabalho.

II. Materiais e Métodos

Para o desenvolvimento deste trabalho foram usados os seguintes materiais:

1) Uma placa de desenvolvimento NodeMCU da LoLin, que conta com um processador ESP8266 da Espressif integrado com um módulo WiFi;
2) Um sensor de ondas eletromagnéticas RFID da MiFare RC522 para chaves (no inglês, *Tags*) de 13,56 MHz;
3) Duas tags RFID de 13,56 MHz, uma em formato de chaveiro e outra em formato de cartão;
4) Um servomotor da HobbyKing com rotação de 0 a 180 ° modelo HK15168 de 1,2 kg;
5) Uma fonte de tensão de 12 V de 2 A;
6) Um dispositivo regulador de tensão para placas de ensaios (do inglês, *protoboard*) de 12 V para 5 V da empresa YwRobot;
7) Uma placa de ensaios;
8) Diversos fios de conexão;
9) Celulares com sistema Android;

10) Serviços do Google Firebase:

Destes materiais, o núcleo do trabalho foi a placa de desenvolvimento NodeMCU, a qual foi confeccionada pela LoLin e consiste em um microcontrolador ESP8266 da Espressif com módulo de comunicação WiFi. Esse microcontrolador é constituído por um processador da Tensilica com tamanho de palavra de 32 bits do tipo RISC, podendo atingir até 160 MHz de frequência, com um baixo consumo de energia e diversos periféricos. Ele foi escolhido dentre outros modelos devido as seguintes características: *(i)* baixo custo, *(ii)* bom desempenho, *(iii)* fácil integração com a interface de desenvolvimento do Arduíno, *(iii)* módulo de comunicação WiFI integrado, e, também, devido a existência de *(iv)* diversas bibliotecas para a integração com diferentes dispositivos.

O microcontrolador tem por objetivo realizar a ponte entre a leitura das chaves RFID pelo sensor MiFare RC522, a comunicação com a rede WiFi para a requisição de informações presentes no banco de dados sobre as chaves RFID lidas pelo sensor, a interpretação da resposta retornada pela requisição e a atuação sobre o servomotor para a simulação da abertura de uma porta. Devido a disparidade entre o tensão do servomotor (5 V) e a placa de desenvolvimento (3,3 V), foi necessário o uso de uma fonte de tensão de 12 V, a qual teve sua tensão reduzida e regulada por um dispositivo regulador de tensão adaptado para placas de ensaios. Além disso, todos esses dispositivos foram interligados através de cabos de conexões, também conhecido no inglês como *jumpers*.

O software, por outro lado, só requer o uso de um celular com sistema operacional Android para execução e de um computador para o desenvolvimento e implementação da aplicação. Ele foi desenvolvido sobre a biblioteca *ReactNative*, voltada para o desenvolvimento da interface visual (*front-end*), e da biblioteca *Node*, para a integração com banco de dados. Ambas bibliotecas atuam sobre a linguagem de programação *javascript*. O banco de dados foi implementado sobre a plataforma de desenvolvimento de aplicativos *Firebase*, o qual contempla diversos serviços que provêm diferentes tipos de funcionalidades a aplicação, métodos de análise e visualização dos dados gerados e ferramentas que auxiliam o seu desenvolvimento. Destes serviços foram usados o serviço de autenticação (*Authentication*) de usuários, para o login usuários no aplicativo; O banco de dados NoSQL (*Cloud Firestore*), para manter a agenda de salas, a relação de usuários e suas chaves RFID únicas e a relação de salas e dispositivos; O serviço de funções em nuvem (*Cloud Functions*) para a implementação de funções chamáveis através do protocolo https.

Ambas partes, o desenvolvimento do hardware e do software, contou com uma adaptação da metodologia de projeto ágil SCRUM [1] para seu desenvolvimento. Por via de regra, na metodologia SCRUM são previstas reuniões diárias entre os integrantes da equipe, e pequenos entregáveis a cada período denominado *sprint*. Porém, para o desenvolvimento deste trabalho, essa metodologia foi adaptada para reuniões semanais, mas com as *sprints* e seus entregáveis mantidas a cada período de aproximadamente 2 semanas.

III. Desenvolvimento do Protótipo

O desenvolvimento do sistema de controle de acesso OpenLockDoor teve como principal motivação a observação da rotina de trabalho dos controladores de acesso das salas da Universidade Federal de São Paulo campus São José dos Campos. Estes, a cada início de aula, devem se deslocar até a respectiva sala para a abertura de suas portas, o que normalmente pode levar algum tempo. Logo, caso fosse possível os próprios professores abrirem as salas, esse deslocamento poderia ser evitado, e, portanto, ser usado para outras atividades. Esse projeto foi dividido em três etapas: *(i)* O planejamento dos requisitos, atividades e a divisão de trabalho; *(ii)* desenvolvimento do *Hardware*, e, em paralelo, *(iii)* o desenvolvimento do *Software*.

A. Planejamento

O desenvolvimento desse projeto primeiramente contou com um encontro síncrono de seus integrantes para a definição do escopo de atuação, das funcionalidades previstas, dos requisitos necessários e uma primeira divisão de tarefas.

Foi definido que o escopo de atuação do projeto não deveria apenas verificar o conhecimento de uma determinada chave RFID, assim como as fechaduras eletrônicas usuais, mas que, visto a característica dinâmica de agendamento de salas que a faculdade conta, isto é, a existência de atividades pontuais como monitorias, o sistema deveria suportar um certo nível de liberdade de agendamento de horários. Portanto, foi definido que nosso projeto deveria apresentar uma frente de hardware, para a leitura e efetivo controle de acesso, e uma frente de software, com um aplicativo para tratamento da relação de agendamentos de salas mantido em um banco de dados.

Dado o escopo de atuação definido, os seguintes requisitos puderam ser derivados:

- Cada pessoa deve ser representada através de um usuário único, assim como conter apenas um número de chave RFID;
  - Essa relação deve ser estabelecida por um administrador do sistema.
- O usuário deve ser capaz de realizar um agendamento de salas;
  - O agendamento pode acontecer a qualquer momento, mas deve especificar uma data, um intervalo de tempo e uma sala específica;
- O usuário deve ser capaz de ver seus agendamentos e excluir-los caso necessário;
- O banco de dados deve manter uma relação de usuários e chaves RFID, de fechaduras e salas, e agendamentos, os quais indicam uma data, um intervalo tempo e uma sala.
- A fechadura deve representar uma única sala;
  - Essa operação deve ser imutável e persistente;
- A fechadura deve ser capaz de ler chaves RFID;

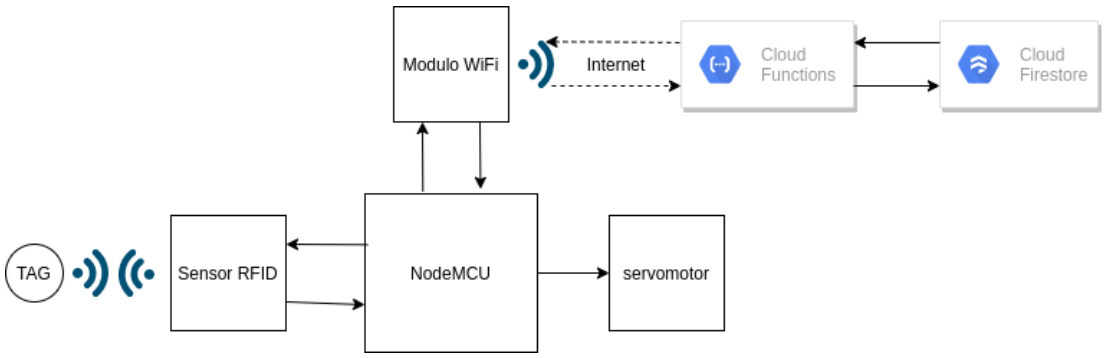

Figura 1. Diagrama de blocos do hardware e as direções das trocas de dados entre os dispositivos.

- A fechadura deve ser capaz de verificar informações no Banco de dados;
  - A verificação de informações no banco de dados deve ser feita através de um rede WiFi com requisições https;
- A fechadura deve ser capaz de liberar o acesso a porta;

Devido a existência do banco de dados, o desenvolvimento do *hardware* e do *software* pôde ser disjunto, e, portanto, acontecer de forma paralela. Por conta disso, a primeira etapa envolveu o time de desenvolvimento todo para a especificação e arquitetura do banco de dados. E, apenas em seguida, o time de desenvolvimento composto por cinco pessoas foi divido para as duas frente, de modo que, quatro dos integrantes foram responsáveis pelo desenvolvimento do *software* e um integrante ficou responsável pelo desenvolvimento do *hardware*.

B. Hardware

A primeiramente na frente de desenvolvimento de Hardware consistiu em uma pesquisa voltada para qual plataforma de desenvolvimento o projeto teria como base, das quais foi separado primeiramente o microcontrolador Arduíno UNO e o PIC18F4550. Porém, com base nos requisitos percebeu-se a necessidade de comunicações com a internet que deve ser estabelecida através de uma comunicação WiFI. Pesquisando tecnologias atuais de comunicação WiFi, encontrou-se o kit de desenvolvimento NodeMCU, que conta com um microcontrolador integrado com um módulo de comunicação WiFi, por um preço equivalente aos microcontroladores anteriores. Logo, verificou-se que a plataforma de desenvolvimento NodeMCU seria mais vantajosa dentre as vistas anteriormente.

O atuador servomotor tem seu controle através de PPM (Pulse Position Modulation), o qual apresenta uma frequência de oscilação fixa, com uma posição da descida de pulso referente a 0°, outra posição referente a angulação máxima e as posições intermediárias referentes as angulações intermediárias [2]. No servomotor usado neste projeto (HobbyKing HK15168) sua angulação varia entre 0° e 180°, com um período de 20 ms e a posição da sua descida de pulso deve estar entre 0,5 ms e 2 ms [2]. A implementação dele foi feita através da biblioteca Servo.h presente na IDE Arduíno, que contém métodos de conectar de servomotores em determinados pinos (*attach*) e de escrita da angulação passada por parâmetro (*write*) [3].

O sensor de RFID conta com um circuito integrado rc522 voltado para a interface de leitura e escrita de cartões RFID no padrão ISO/IEC 14443 A/MIFARE and NTAG de 13,56 MHz [4]. Ele conta com interfaces de comunicação I2C, SPI e USART, mas que, no dispositivo desenvolvido pela empresa keyStudio, apenas está disponível a interface SPI de comunicação [4]. É um circuito integrado completo, com diversas funcionalidades internas, como a implementação de filas, interrupções, temporizadores, pinos de entrada e saída de uso geral e um co-processador de código de verificação CRC. Nesse trabalho foi usado a biblioteca MFRC522.h disponível para a IDE Arduíno que tem portabilidade para o microcontrolador ESP8266 [5]. Seu uso é feito a partir da instanciação de um objeto da classe MFRC522, e os métodos usados foram: *PCD_Init* para inicialização da comunicação com o sensor; *PICC_IsNewCardPresent* para a verificação se existe um novo cartão presente; *PICC_ReadCardSerial* para a leitura do código do cartão presente. Além disso, foi necessário o acesso ao atributo *uid.uidByte*, um vetor que contém o valor da chave lida [5].

A comunicação com a rede WiFi é feita através do módulo de comunicação WiFI integrado a placa de desenvolvimento NodeMCU. O controlador desse módulo é integrado com o CI ESP8266 na forma de um periférico, e, portanto, o uso dele não requer uma porção muito grande do tempo de processamento. Neste trabalho foi usado a classe ESP8266WIFI.h e ESP8266WIFIMULTI.h para configurações gerais do módulo WiFi (conexão com o ponto de acesso, modo de operação, etc.), ESP8266HTTPCLIENT.h e WiFiClient.h para a instanciação de clientes de rede para acesso a servidores na conexão externa e WiFiClientSecureBearSSL.h para acesso aos padrões de criptografia usados nos clientes HTTPS [6].

O hardware foi configurado no modo estação, para a conexão em uma rede de WiFi externa provida por um ponto de acesso. Foram configurados o nome da rede, a senha e o código de impressão digital do servidor alvo, a qual foi necessária porque os serviços de acesso pela rede dos servidores do Firebase são configurados através de requisições HTTPS, que requerem o protocolo de segurança TLS 1.2 na camada de

transporte do protocolo HTTP.

O funcionamento da programação do microcontrolador está descrito no Algoritmo 1, de tal modo que a interconexão e o caminho dos dados entre os diferentes dispositivos pode ser visto no diagrama de blocos apresentado Figura 1.

Algorithm 1 Visão geral da execução do microcontrolador

```
1:  function SETUP
2:      Iniciar Comunicação USART
3:      Iniciar Módulo WiFi como Station
4:      Conectar na Rede Configurada
5:      Define a impressão digital do servidor
6:      Inicia comunicação SPI
7:      Inicia o módulo de leitura de RFID
8:      Inicia o servomotor fechado
9:      if Existe sala salva na EEPROM then
10:         Lê a numeração da sala
11:     else
12:         Lê a numeração do Banco de dados
13:         Salva a numeração na EEPROM
14:     end if
15: end function
16: function LOOP
17:     if Não tem nenhum cartão presente then return
18:     end if
19:     if Não tem cartão para ler then return
20:     end if
21:     Trava a leitura do mesmo cartão
22:     Lê o cartão
23:     Tratamento do cartão e composição da requisição
24:     Requisita ao banco de dados e recebe o retorno
25:     if Retorno é verdadeiro then
26:         Abre a porta
27:     else
28:         Não abre
29:     end if
30: end function
```

C. Software

O *software* foi implementado de tal forma que assegura-se uma interface amigável entre o banco de dados e o usuário, que, em geral, são os professores da universidade e a central de segurança do prédio. Além disso ele foi pensando para execução em plataformas mobile, em especial, para *smartphones* com sistema operacional Android.

A primeira etapa do desenvolvimento de software consistiu na decisão do modelo de banco de dados que seria desenvolvido, assim como sua arquitetura e modelagem, visto que ambas as frentes da aplicação (o hardware e o software) teriam por base chamadas ao banco de dados. O banco de dados usado foi o *cloud firestore* do conjunto de serviços do Firebase. Sua arquitetura pode ser vista na Figura 2, a qual prevê uma coleção de usuários que se relaciona com uma coleção de chaves RFID, uma coleção de salas que se relaciona com as identificações das fechaduras e uma coleção de agendamento que contém os meses, com cada um deles contendo coleções de dias, que, por sua vez, contém coleções de agendamentos com os campos: hora de início, hora de finalização, professor (se relaciona com a coleção usuários) e sala (se relaciona com a coleção salas).

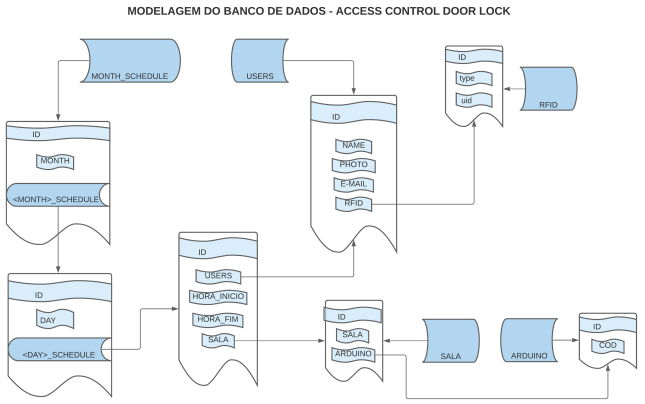

Figura 2. Arquitetura das coleções e documentos do banco de dados *cloud firestore*.

Em seguida, o desenvolvimento do *front-end* foi paralelo ao desenvolvimento do *back-end*, de modo que, quando novas funcionalidades se mostravam necessárias durante o planejamento do *front-end*, o *back-end* era implementado de tal forma que garantisse o funcionamento dessa funcionalidade. O *front-end* foi desenvolvido sobre a biblioteca ReactNative e o *back-end* sobre a biblioteca Node, ambos sobre a linguagem de programação javascript, caracterizando um *stack* completamente baseado em javascript.

A aplicação implementou as seguintes funcionalidades: *(i)* Um sistemas de autenticação de usuários mantido com uso da ferramenta *Firebase Auth*, assegurando segurança a nossa aplicação; *(ii)* Um sistema de visualização, alteração e exclusão de reservas de salas, o qual realiza chamadas ao banco de dados para o determinado usuário logado na aplicação; *(iii)* um sistema de reserva de salas, cujo usuário indica o dia e as horas pré-definidas.

IV. Resultados e Discussões

O banco de dados foi implementado sobre o serviço *cloud firestore* do Firebase acordando com a arquitetura proposta na seção anterior. Ele resultou nas coleções mostradas na Figura 3, podendo se observar que os documentos de usuários, por exemplo o mostrado mais a direita da figura, relaciona o usuário a um determinado documento de RFID, o qual será único para cada usuário. Esse mesmo padrão foi replicado para a relação de agendamentos com as salas e a relação de agendamentos com os usuários. Usando referências como relações é possível acessar o documento referenciado a partir do documento atual, diminuindo a replicação de documentos no banco de dados.

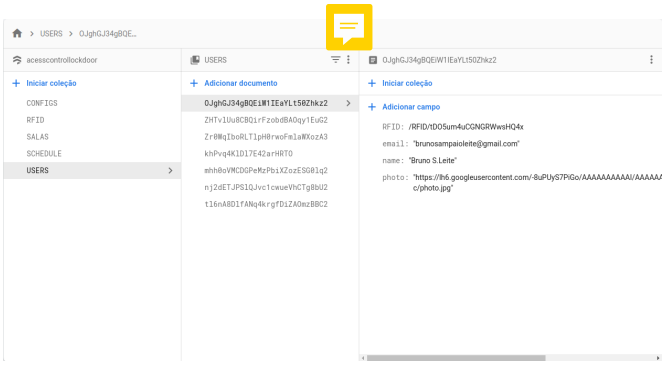

Figura 3. Resultado final do banco de dados NoSQL Cloud Firestore.

O desenvolvimento do hardware resultou em um protótipo montado sobre uma placa de ensaios, de tal forma que o sensor RFID foi fixado na placa de ensaios com uso de um [illegible] rígido e conectado com o microcontrolador através de cabos de conexão. Da mesma forma, o servomotor foi conectado ao microcontrolador através do uso de cabos de conexão, mas, agora, com uso das pistas condutoras da placa de ensaio. O regulador de tensão, usado para fornecer tensão suficiente para o servomotor, recebe a tensão da fonte de tensão 12 V e diminui para 5 V, conectando-a com as pistas laterais da placa de ensaios. Além disso o "Terra" do microcontrolador teve de ser interligado com o "Terra" do regulador de tensão, visto que suas alimentações eram provenientes de fontes distintas. Todo o protótipo esta apresentado na Figura 4.

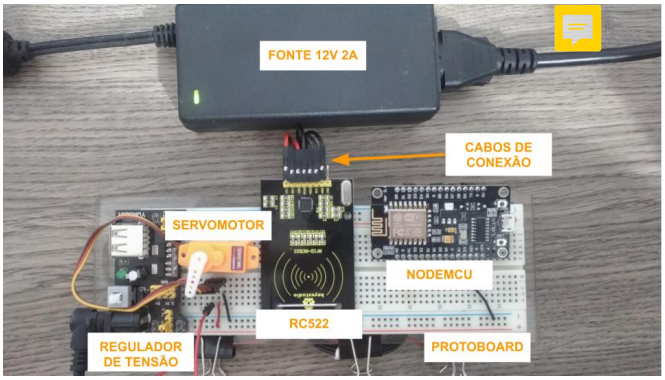

Figura 4. Montagem final do protótipo de hardware da fechadura eletrônica com controle de acesso.

A princípio, a comunicação serial do microcontrolador foi configurada para fornecer informações importantes sobre o status interno do dispositivo, como o IP e em qual etapa de execução ele se encontra no momento. Contudo, caso interligado seu pino V_{in} com o nível de tensão 5 V da pista lateral da placa de ensaios, seu funcionamento ainda acontece de acordo com o esperado sem o uso de um computador auxiliar para a comunicação serial, desde que, o protótipo esteja sobre o alcance da rede WiFi configurada inicialmente.

A comunicação entre o hardware e o banco de dados foi implementada sobre o serviço do Firebase *functions*, no qual criou-se um código em javascript que é chamado e executado em uma instância de computador virtual quando chega uma requisição via HTTPS sobre um endereço determinado. Na Figura 5 é possível ver as duas funções desenvolvidas, a função denominada *verifyTAG* e a função denominada *getID*, alem de seus respectivos endereços de requisição. A função *verifyTAG*, prevê uma sequência de caracteres (do inglês, *String*) referente a chave de um cartão e de uma numeração de sala, onde, em seguida, acontece o acesso ao banco de dados *cloud firestore* para a verificação das seguintes condições: A chave é mestre ?; A chave é conhecida e referente a um usuário ?; a sala existe ?; existe alguma agendamento para o horário atual, para aquele usuário e naquela sala ?. Ao fim, caso a primeira verificação ou todas as demais forem verdade, é retornado uma resposta ao hardware permitindo a abertura da sala, e, caso negativo, é retornado falso para o hardware. A função getID é chamada pelo hardware apenas quando não existe uma sala gravada em sua memória não volátil EEPROM, e, nesse caso, é verificado um documento específico do banco de dados que retorna ao hardware sua numeração de sala para ser salvo na memória EEPROM.

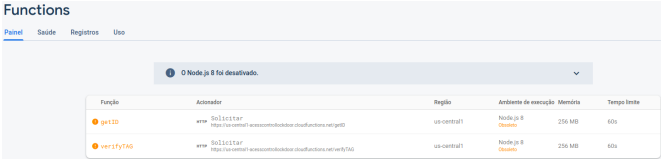

Figura 5. Funções chamáveis via https definidas no serviço *functions* do Firebase.

A aplicação de Software consistiu no desenvolvimento de uma interface gráfica referente ao *front-end* e uma parte de acesso ao banco de dados referente ao *back-end*. Na Figura 6 pode ser observado todas as telas do *front-end* desenvolvidas para a aplicação Android, de modo que: As telas (a,b) referem-se a etapa de login do usuário, assegurando que cada usuário tenha apenas uma chave RFID; a tela (c) mostra uma lista de reservas do respectivo usuário; a tela (d) mostra o menu lateral da aplicação; as telas (e, f, g, h) referem-se ao sistema de agendamento de salas, que, respectivamente, permitem a escolha do dia para o agendamento, a escolha do andar da sala, a escolha da sala, e, ao fim, a escolha dos horários desejados.

Na lista de salas apresentada na Figura 6(c), também foi implementado um modo de exclusão de uma determinada reserva caso necessário. Além disso, na sub-figura (h), caso existam salas que já foram reservadas para um determinado horário, estes horários aparecem riscados, não permitindo o agendamento nestes horários. Vale ressaltar que, algumas etapas importantes devem ser feitas por um administrador do sistema, que, atualmente, devem ser inseridas diretamente no banco de dados, como a designação de um chave RFID para um determinado usuário e o controle da numeração da sala para a gravação da *EEPROM* do hardware.

Portanto, com os desenvolvimentos supracitados pôde-se constatar o correto funcionamento individualmente da aplicação desenvolvida pela frente de *software* e do protótipo desenvolvido pela frente de *hardware*. Como ambas frentes

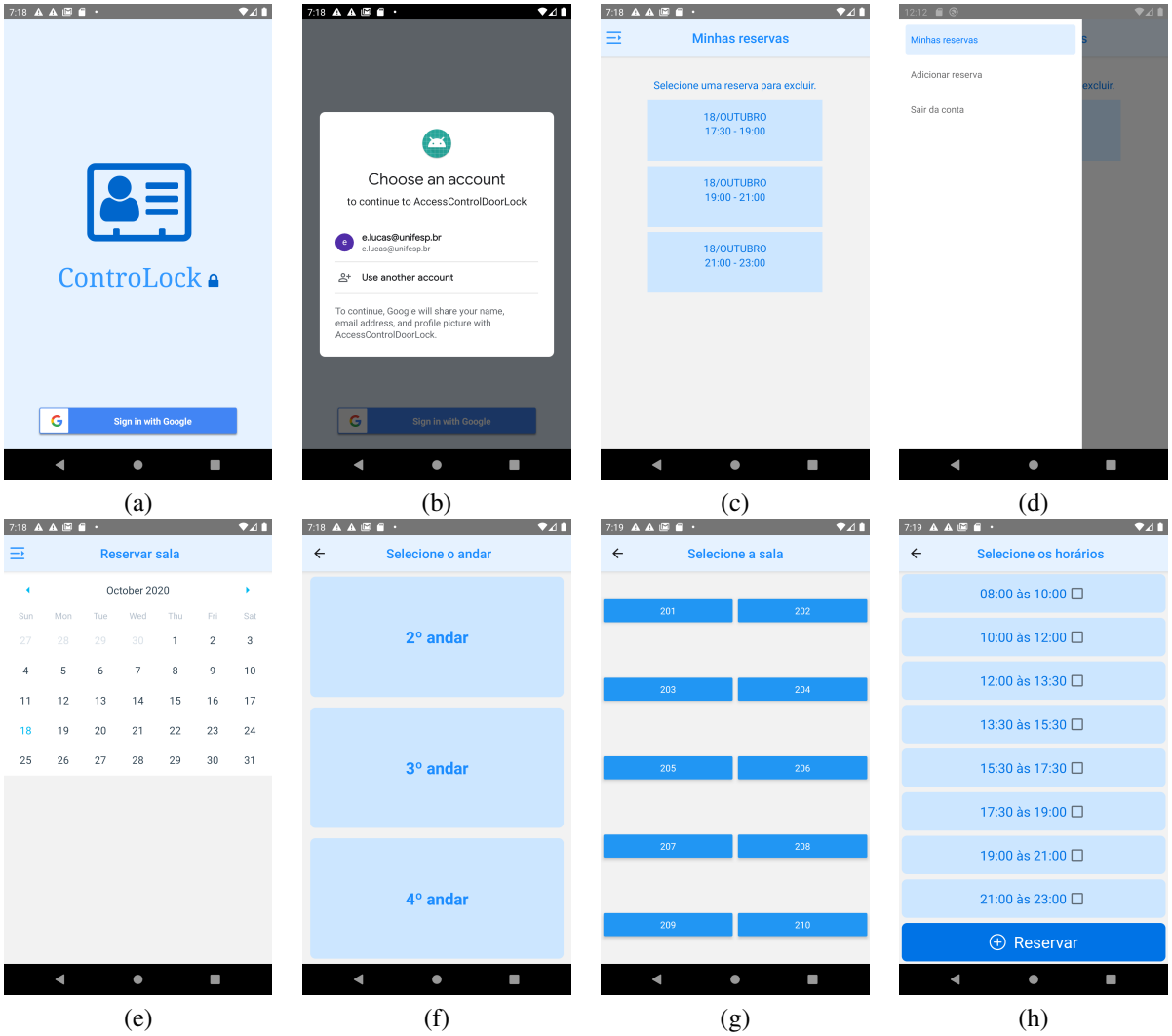

Figura 6. Telas da aplicação Android. As telas (a,b) referem-se a etapa de login da aplicação; (c) mostra as reservas atuais; (d,e,f,g) contém as paginas necessárias para o agendamentos de salas.

não devem ser fisicamente integradas, e, essa integração é feita através da camada do banco de dados, também foi possível constatar o correto funcionamento da aplicação como um todo. Todas as implementações foram mantidas pelo versionador de arquivos GIT pelos servidores do GitHub, podendo ser acessado em [7].

V. Conclusão

O trabalho contribuiu com o desenvolvimento de uma aplicação completa, hardware e software, de uma fechadura eletrônica com controle de acesso por cartões RFID. Seu controle de acesso é mantido por um banco de dados hospedado por servidores na internet e controlado por uma aplicação mobile para Android. O *hardware* tem por objetivo a leitura de chaves RFID através de um sensor, a requisição de informações através de uma comunicação com a internet e o controle de um um servomotor. Enquanto que o *software* tem por objetivo a manutenção dos agendamentos do banco de dados.

Devido ao curto período de desenvolvimento, a aplicação mobile teve um ritmo de desenvolvimento acelerado e, portanto, sua interface é simples e direta. Da mesma forma, o hardware foi desenvolvido sobre um protótipo conectado com cabos de prototipagem sobre uma placa de ensaios, não possuindo um formato final que possa ser aplicado diretamente a fechaduras. Mas que, apesar disso, pôde ser verificado e validado seu correto funcionamento em acordo com os requisitos impostos anteriormente.

Como perspectivas futuras, podemos citar melhorias da interface da aplicação, permitir diferentes tipos de usuários que tenham acessos a diferentes conjuntos de funcionalidades respectivos ao seu tipo de usuário, criação e o desenvolvimento de um envólucro compatível com portas e um sistema para configuração de parâmetros internos do hardware.

Agradecimentos

O autor L. F. A. Silva agradece ao apoio financeiro da FAPESP cedido sobre o número de processo: 2020/08770-3.

REFERÊNCIAS

[1] L. Williams and A. Cockburn, "Guest editors' introduction: Agile software development: It's about feedback and change," *Computer*, vol. 36, pp. 39– 43, 07 2003.

[2] "Datasheet servomotor sg90," (acessado em 19 de outubro de 2020). [Online]. Available: https://datasheetspdf.com/pdf-file/791970/TowerPro/SG90/1

[3] Arduíno, "Documentação biblioteca servo," 2020 (acessado em 19 de outubro de 2020). [Online]. Available: https://www.arduino.cc/reference/en/libraries/servo/

[4] NXP, "Mfrc522," 2016 (acessado em 19 de outubro de 2020). [Online]. Available: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf

[5] miguelbalboa and Rotzbua, "Repositório mfrc522," 2020 (acessado em 19 de outubro de 2020). [Online]. Available: https://github.com/miguelbalboa/rfid

[6] esp8266, "Repositório esp8266," 2020 (acessado em 19 de outubro de 2020). [Online]. Available: https://arduino-esp8266.readthedocs.io/en/latest/index.html

[7] B. S. Leite, L. F. A. e Silva, L. S. Vaz, T. L. Lima, and V. S. do Amaral, "Repositorio do projeto access control lock door," 2020 (acessado em 19 de outubro de 2020). [Online]. Available: https://github.com/brunosampaio08/AccessControlDoorLock