IF-KEY: Sistema de Controle de Chaves

Gessica S. S. Lehmann, Daniela Marques, Leandro C. Ledel

[1]Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Câmpus Hortolândia
Av. Thereza Ana Cecon Breda, S/N - Vila São Pedro, Hortolândia - SP - Brasil, 13183-250

furquim.gessica@aluno.ifsp.edu.br, {marquesdaniela, ledel}@ifsp.edu.br

Abstract. *The* IF-KEY *key control system emerged from the difficulties noted in locating the keys for the IFSP-HTO laboratories and classrooms. An iterative and incremental model was used for developing the system requirements. The architecture was organized into logical layers, which separated presentation, management, and data storage. Through the use of QR-Code inserted in the keys of the Institute's rooms and laboratories, it was possible to access the link to the web system, in which the user was be able to log in or register, so that he could schedule the use of a room or laboratory and/or search for the current user, thus facilitating the location of keys. The final web application was found to have successfully achieved its objectives from an implementation standpoint.*

Resumo. *O sistema de controle de chaves* IF-KEY *surgiu a partir das dificuldades notadas na localização das chaves dos laboratórios e salas de aula do IFSP-HTO. Utilizou-se um modelo iterativo e incremental para o desenvolvimento dos requisitos do sistema. A arquitetura se deu em camadas lógicas, as quais separaram a apresentação, o gerenciamento e o armazenamento de dados. Através do uso de QR-Code inseridos nas chaves das salas e laboratórios do Instituto, foi possível acessar o link para o sistema web, no qual o usuário foi capaz de efetuar login ou cadastro, para que fosse capaz de agendar o uso de uma sala ou laboratório e/ou realizar a busca pelo usuário atual, facilitando assim a localização das chaves. A aplicação web final demonstrou ter atingido os seus objetivos do ponto de vista da implementação.*

1. Introdução

O *IF-KEY* foi proposto como tema de trabalho de conclusão do curso com o objetivo de melhorar o controle de usuários aos laboratórios e salas de aula do IFSP-HTO. A motivação para o desenvolvimento do projeto de pesquisa veio da necessidade de organização e responsabilidade pelas chaves dos laboratórios e salas de aula.

O claviculário do IFSP - Campus Hortolândia se encontra na parede próxima a cozinha, para que seja fácil aos servidores retirarem as chaves de salas de aula e laboratórios. Porém levando isso em consideração percebeu-se que havia uma dificuldade em localizar os usuários das chaves, por vezes professores, outras alunos. Sendo que, em certas ocorrências o utilizador não as retorna em seu devido lugar após o uso, causando impacto para outros usufrutuários, como o atraso nas aulas.

O objetivo do projeto é a criação de uma aplicação para plataforma web. O usuário será redirecionado a aplicação através de um *link* em formato QR-Code que será impresso e fixado nas chaves do Instituto. O utilizador do sistema será capaz de criar sua conta e

em seguida poderá este, acessar o sistema e consultar chaves ou agendar a utilização de uma sala ou laboratório do Instituto.

Desta forma, espera-se contribuir com a entidade de ensino, através do tema acima abordado, obtendo melhor controle de usuários, contando -se com uma ferramenta extra na detecção de eventual responsabilidade pelo uso dos laboratórios e/ou salas de aulas. No decorrer deste artigo serão apresentados os referenciais teóricos, os trabalhos correlatos, escopo do produto, os requisitos funcionais e não funcionais, a arquitetura da aplicação, o desenvolvimento da aplicação web.

2. Referencial Teórico

Apresentaremos agora os conceitos fundamentais para o desenvolvimento deste projeto, bem como seus embasamentos teóricos.

2.1. Sistema de Informação

O autor [SOMMERVILLE 2011] define sistemas de informação como todos os sistemas que envolvem a interação com bancos de dados compartilhados. Um sistema de informação consiste de uma camada de controle de acesso aos dados de uma determinada base, de forma que estes estejam protegidos de acesso não autorizado e da escrita não coerente de novos dados.

2.2. Desenvolvimento Incremental

O desenvolvimento incremental é uma abordagem que, de acordo com [SOMMERVILLE 2011], consiste em decompor as funcionalidades e requisitos de um produto em partes menores e sucessivas que possam ser desenvolvidas como uma série de incrementos, de maneira que cada destes adiciona funcionalidade ao anterior. Vide Figura 1, para melhor entendimento do modelo incremental.

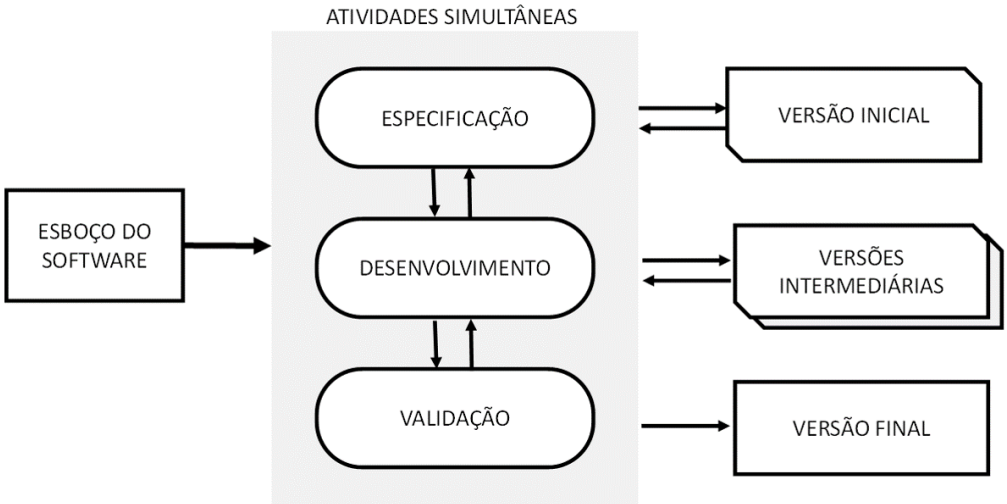

Figura 1. Modelo de processo Incremental (Fonte: SOMMERVILLE, 2011)

2.3. Ferramentas para o desenvolvimento

Nesta Seção são abordados as tecnologias que serão utilizadas no desenvolvimento do projeto.

2.3.1. Bootstrap

O Bootstrap, de acordo com sua documentação oficial, é um *framework* de código aberto, que inclui CSS, HTML e JavaScript para o desenvolvimento de aplicações web responsivas, isto é, aplicações cujas interfaces adaptam-se a diferentes dimensões de telas e dispositivos, aumentando a usabilidade destas.

2.3.2. Contêiner

Neste projeto, faremos uso do Docker, que é um conjunto de ferramentas multiplataforma para facilitar a criação, execução e instalação de aplicações, dentro de um ambiente virtual composto de contêineres. Contêineres são criados à partir de imagens que consistem de pacotes autônomos que compõem todas as dependências da aplicação em questão. A especificação de contêineres, seu funcionamento e gerenciamento são padronizados pela *[Open Container Initiative]* (OCI). Também será utilizado uma ferramenta adicional chamada *Docker Compose* que é utilizada para orquestrar a execução de vários contêineres que interagem entre si.

2.3.3. Easy QR

Nas chaves serão utilizados códigos de resposta rápida, também conhecidos por QR codes. Estes serão gerados através do aplicativo de código aberto *Easy QR* [Sunny Attic SOFTWARE], disponível na loja de aplicativos. O aplicativo é apresentado na Figura 2. Após gerar estes códigos, eles serão plastificados e aderidos aos chaveiros das chaves do instituto para que o usuário possa fazer a leitura com seu dispositivo móvel, sendo assim redirecionado para tela de login do IF-KEY. Desta forma, o usuário poderá realizar o login ou se cadastrar no sistema.

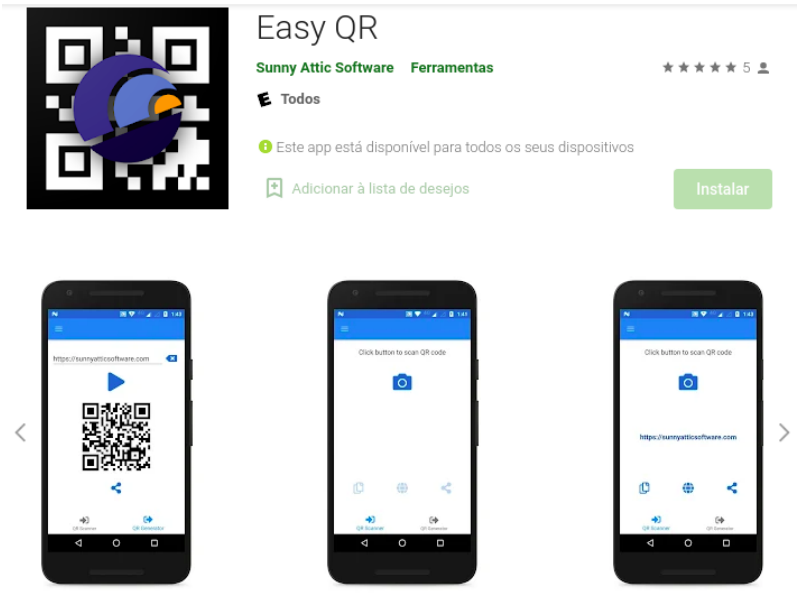

Figura 2. Aplicativo Easy QR

2.3.4. HTML e CSS

Segundo [FLATSCHART 2011], o HTML (*Hyper Text Markup Language* – Linguagem de Marcação de Hipertexto) é a principal linguagem utilizada na web, para criação de documentos estruturados. Sendo uma aplicação web, o *IF-KEY* fará uso do HTML para a estruturação das páginas que comporão a interface do usuário. O CSS (*Cascading Style Sheets*) é uma linguagem voltada para a criação de folhas de estilo, e como [SEGURADO 2017] nos indica, é um conjunto de regras responsáveis pela aparência de um site. Portanto, o CSS será utilizado para lapidar a interface de usuário, com intuito de que seja mais amigável e intuitiva do ponto de vista do usuário.

2.3.5. JavaScript

JavaScript é uma linguagem de programação, que de acordo com [MANZANO and TOLEDO 2009], pode ser utilizada dentro do ambiente de um navegador ou em um servidor. No caso do navegador, a linguagem JavaScript serve para dar dinamismo às páginas HTML, aspecto este que será utilizado no *frontend* do *IF-KEY*.

2.3.6. JSON

JSON ou *JavaScript Object Notation*, em tradução literal, notação de objeto JavaScript, é um formato de serialização de dados, de fácil leitura e entendimento humanos e que serve para troca de dados entre diferentes sistemas, comumente utilizado em APIs REST [BASSETT 2015]. Este formato será abordado de maneira mais aprofundada nas próximas seções.

2.3.7. Node.js

Segundo [KIESSLING 2012], Node.js é um ambiente de tempo de execução que junto de sua biblioteca padrão, permite executar JavaScript fora do navegador. Devido a isto, no *IF-KEY*, o Node.js será a plataforma sobre a qual será desenvolvida a API (*application programming interface*, do inglês, interface de programação de aplicação) que representará as regras de negócio, eventualmente acessadas através do *frontend*.

2.3.8. MySQL

O funcionamento correto do *IF-KEY* requer uma base de dados robusta, confiável e de alto desempenho, características estas que estão presentes no MySQL. O autor [NIEDERAUER and PRATES 2005] denomina o MySQL como um Sistema de Gerenciamento de Banco de Dados relacional (SGBD). Esse SGBD, faz uso da linguagem SQL, sigla para *Structured Query Language*, do inglês, Linguagem Estruturada para Consulta que é a linguagem padrão utilizada em bancos de dados relacionais, conforme [PUGA et al. 2013]. O MySQL é utilizado para realizar as operações *CRUD* (*Create, Read, Update, Delete*) no *IF-KEY*, permitindo a manipulação dos dados do sistema.

2.3.9. React

React, uma biblioteca JavaScript, é uma ferramenta utilizada para o desenvolvimento de aplicações web. Segundo a documentação oficial do React, [Equipe de Desenvolvimento do React], a biblioteca tem por objetivo auxiliar os desenvolvedores na construção de aplicações de forma componentizada. Os componentes no React são reutilizáveis e autônomos, permitindo que os desenvolvedores os escrevam de maneira declarativa.

3. Trabalhos Correlatos

Através de uma busca no site do Instituto Federal de São Paulo - Campus Hortolândia, foi possível localizar o trabalho de conclusão de curso do aluno Justh Franklin M. Leal, *Finder: Aplicativo de Achados e Perdidos* [Leal, 2017]. Este aplicativo realiza o cadastro de itens perdidos e encontrados e compartilha estas informações para que quem perdeu seus objetos possa os encontrá-lo com facilidade.

Consideramos o artigo mencionado relevante para este contexto, uma vez que segue a mesma abordagem de assistência ao usuário e também adota a arquitetura REST em seu desenvolvimento. No entanto, os sistemas apresentam abordagens distintas: enquanto o Finder propõe a busca de itens perdidos através de GPS, o IF-KEY propõe o controle do uso das chaves do Instituto, inicialmente por meio de QR-Code. No entanto, em trabalhos futuros, essa funcionalidade pode ser aprimorada com a utilização de tecnologia RFID. Na Figura 3, podemos notar a tela de Login e de Registro da aplicação *Finder*.

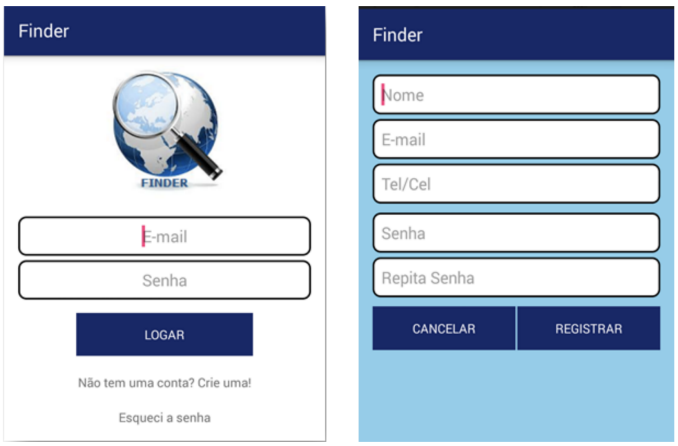

Figura 3. Finder [Leal, 2017]

4. Metodologia

Com o objetivo de melhorar o controle de usuários aos laboratórios e salas de aula do IFSP-HTO, sucederá preliminarmente a realização da pesquisa bibliográfica. Com embasamento em trabalhos correlatos, decorrerá a investigação de linguagens de programação cabíveis ao contexto do desenvolvimento do *software* de controle de acesso às chaves e inserção de QR-Code impresso nas chaves do instituto, com *link* que redirecionará o usuário ao *IF-KEY*.

O gerenciamento do projeto foi organizado através do desenvolvimento iterativo e incremental dos requisitos funcionais e não funcionais do sistema.

A arquitetura do projeto se dará em camadas lógicas, para que as informações sejam apresentadas separadamente do seu gerenciamento, desagregadas da implementação da lógica da aplicação e ainda isolado do armazenamento destes dados. A totalidade do processo será validado em conformidade com o documento de requisitos de *software*.

5. Desenvolvimento

Para o desenvolvimento do projeto, foi planejada um arquitetura de *software* que permita com que, no futuro, seja criado um dispositivo físico, como um armário inteligente, que seja responsável por gerenciar o acesso físico às chaves das salas do IFSP.

Enquanto o *software* desenvolvido neste trabalho será responsável por coordenar as reservas de chaves feitas pelos usuários, seus respectivos períodos de validade e a atribuição de códigos QR para a identificação destas tanto no momento de retirada, quanto na devolução, tal armário seria responsável por ler o código QR, comunicar-se com o *backend* do *IF-KEY* para que este realize as devidas validações e então tome a decisão de permitir que o usuário receba, ou não, a chave relativa à reserva em questão.

Os detalhes da implementação desse armário, como hardware, sensores e afins, estão fora do escopo deste trabalho. Nas seções seguintes serão descritos, entre outras coisas, os protocolos envolvidos para que seja possível que um *software* comunique-se como um cliente do *IF-KEY*, supondo que um armário como o descrito acima já exista.

5.1. Atividade 1: Definição de trabalhos correlatos

Esta atividade pode ser vista com maiores detalhes na seção 3. Trabalhos Correlatos.

5.2. Atividade 2: Escopo do Produto

O escopo do produto inclui o gerenciamento de usuários, a administração do acesso às chaves e a capacidade dos usuários de agendar salas e laboratórios do IFSP por meio da aplicação web *IF-KEY*.

5.3. Atividade 3: Definição dos requisitos funcionais

Na Figura 4, é possível visualizar o diagrama caso de uso do *IF-KEY*. O usuário do *IF-KEY* deve ser capaz de realizar as atividades a seguir:

1. Realizar login.
2. Realizar cadastro do seu usuário.
3. Consultar agenda das salas e laboratórios.
4. Realizar agendamentos.
5. Alterar agendamentos.
6. Excluir agendamentos.

O usuário administrador deve ser capaz das mesmas ações que o usuário comum. Ademais, as atividade mencionadas abaixo.

1. Adicionar chaves.
2. Consultar chaves.
3. Alterar chaves.
4. Excluir chaves.
5. Excluir usuários.

Figura 4. Diagrama de caso de uso do *IF-KEY*

5.4. Atividade 4: Definição dos requisitos não-funcionais

A aplicação web *IF-KEY* terá os seguintes requisitos não-funcionais:

1. O sistema deve ser de fácil uso.
2. O sistema deve ser organizado de forma a minimizar erros dos usuários.
3. As operações devem ser executadas rapidamente.
4. As operações devem ser implementadas de maneira eficiente.
5. As operações não devem impedir o usuário de atingir seu objetivo em tempo hábil.
6. As operações não devem onerar o desempenho dos serviços da aplicação, evitando gargalos.

5.5. Atividade 5: Criação da arquitetura do *software*

Nesta seção será abordado como foi feita a arquitetura da aplicação.

5.5.1. Representational State Transfer (REST)

REST (*Representational State Transfer*) em tradução literal, transferência de estado representacional, é um tipo de arquitetura de *software* que estabelece uma coleção de regras para criação de serviços web. Na Figura 5, podemos ver a arquitetura em alto nível do *IF-KEY*, que utiliza API REST em seu *backend*.

No *IF-KEY* serão criados *endpoints* que representarão seus casos de uso. Estes *endpoints* consistem de uma combinação de diferentes verbos do protocolo HTTP (ex:

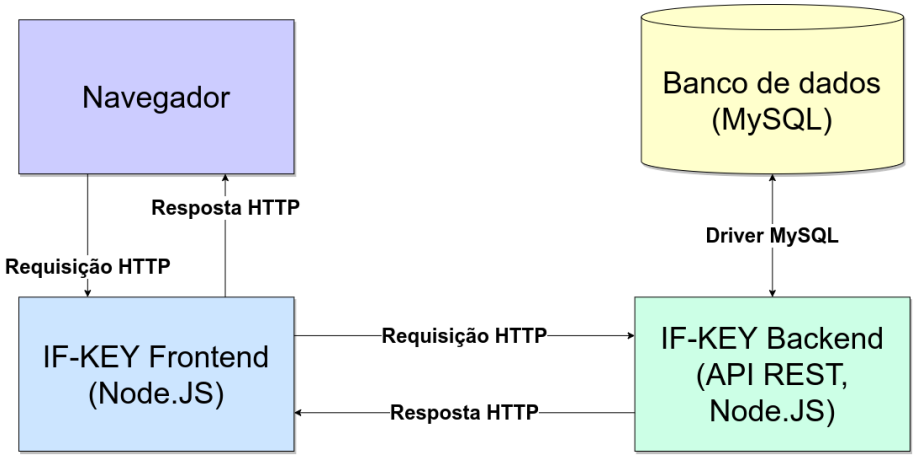

Figura 5. Arquitetura do *IF-KEY*: Sistema de Controle de Chaves

GET, POST, PATCH, DELETE), com caminhos de acordo com os padrões de URI (*Uniform Resource Identifier*) da web (ex: */usuario/3*).

5.6. Atividade 6: Elaboração do ambiente de desenvolvimento

Nesta seção será abordada como foi feita a elaboração do ambiente de desenvolvimento.

5.6.1. Dockerfile

Foram criados arquivos denominados *Dockerfiles* que contém descrições de como preparar os diferentes ambientes de execução do *IF-KEY*, como ambiente de desenvolvimento e produção, utilizados respectivamente, durante o desenvolvimento da aplicação e na execução da aplicação em seu ambiente final.

Na Figura 6, podemos ver o *Dockerfile* do *IF-KEY*. O arquivo *Dockerfile* relativo ao ambiente de desenvolvimento, utiliza como base uma imagem contendo o NodeJS na versão 16. Em seguida, é feita a cópia do código-fonte e outros arquivos necessários para o gerenciamento e instalação de dependências. Por fim, é feita a execução do script de entrada do *backend*, o main.js. A execução é feita utilizando a ferramenta *nodemon* que detecta mudanças nos arquivos de código e reinicia a aplicação com a versão atualizada, para facilitar durante o desenvolvimento.

A imagem de contêiner para o ambiente de produção é feita de maneira semelhante, porém não utilizamos do *nodemon* para executar o arquivo de entrada.

```
FROM node: 16
WORKDIR / opt / app
COPY package * .json. /
    RUN npm install\ &&
    npm install - g nodemon
COPY..
EXPOSE 8080
CMD["nodemon", "src/main.js"]
```

Figura 6. Dockerfile do *IF-KEY*: Sistema de Controle de Chaves

5.6.2. Docker compose

O trecho de código apresentado subsequentemente na Figura 7, descreve os serviços necessários para obtermos o ambiente de desenvolvimento completo.

Temos dois serviços, sendo um deles um contêiner baseado na imagem da Figura 6 e o outro utilizando a imagem docker oficial do MySQL, que é o SGBD escolhido para este projeto.

```yaml
version: "3"

services:
  app:
    build: .
    [...]
    ports:
      - "127.0.0.1:8080:8080"
    volumes:
      - "./:/opt/app"

  db:
    image: mysql:8.0.27
    restart: always
    [...]
    environment:
      MYSQL_DATABASE: "if-key"
      MYSQL_ROOT_PASSWORD: ""
      MYSQL_ALLOW_EMPTY_PASSWORD: "yes"
```

Figura 7. Docker Compose do *IF-KEY*: Sistema de Controle de Chaves

5.6.3. Package.json

Neste projeto é utilizado o Node.js e usaremos o NPM (Node Package Manager) para gerenciar as dependências e outros recursos do projeto. No código da Figura 8, podemos ver o arquivo package.json que entre outras coisas, descreve a lista de dependências do projeto e suas respectivas versões.

```json
{
    "name": "if-key-backend",
    "version": "1.0.0",
    "description": "",
    "author": "Gessica Lehmann <furquim.gessica@aluno.ifsp.edu.br>",
    "main": "main.js",
    "scripts": {
        "start": "node main.js"
    },
    "dependencies": {
        "@mysql/xdevapi": "8.0.27",
        "express": "^4.16.1"
    }
}
```

Figura 8. Package.json do *IF-KEY*: Sistema de Controle de Chaves

5.7. Atividade 7: Desenvolvimento do banco de dados

Conforme relatado por PUGA, S., FRANÇA, E., e GOYA, M. (2013), no livro "Banco de Dados: Implementação em SQL, PL/SQL e Oracle 11g", a Modelagem de Dados desempenha um papel importante na concepção e implementação de sistemas de banco de dados. O DER ou Diagrama Entidade-Relacionamento é uma ferramenta que permite a obtenção dos requisitos do sistema e a projetação de uma estrutura de dados que atenda às necessidades da aplicação.

O modelo de dados desenvolvido na Figura 9, captura os cenários planejados para o *IF-KEY* com um conjunto relativamente pequeno de tabelas (ou entidades). Além disso, todo a implementação do esquema foi feita de forma a suportar caracteres do idioma português, utilizando codificação UTF−8 para campos de texto. Em particular, foram utilizando o conjunto de caracteres (*charset*) utf8mb4 e método de ordenação de texto (*collation*) utf8mb4_0900_ai_ci.

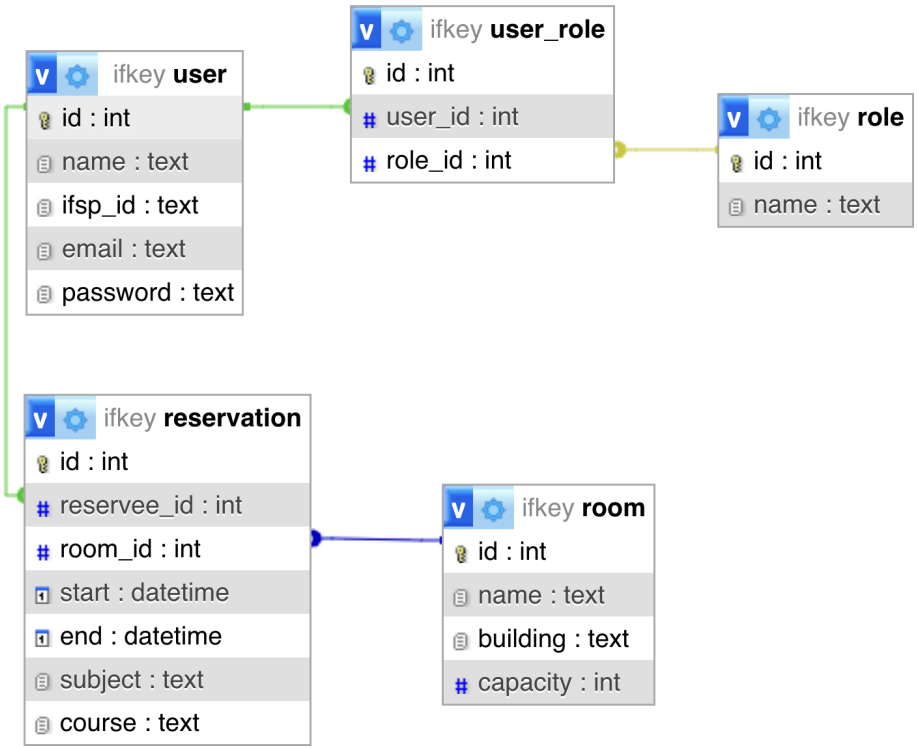

Figura 9. Diagrama Entidade-Relacionamento do *IF-KEY*

A Figura 10 apresenta a tabela room, que é utilizada para identificação interna das salas que terão acesso gerenciado através das reservas feitas pelo *IF-KEY*, contendo a sua chave primária id, a e o campo name que contém a identificação da sala, como é feita no IFSP Hortolândia. Nesta tabela também temos a referência ao prédio em que a sala está e a capacidade de alunos que ela comporta.

```
CREATE TABLE IF NOT EXISTS `room` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` text NOT NULL,
  `building` text NOT NULL,
  `capacity` text NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Figura 10. Tabela *room* do *IF-KEY*

A tabela role (do inglês, papel ou função), vista na Figura 10, representa os diferentes papéis que um usuário pode assumir nos casos de uso, descritos na Figura 4 e possui o intuito de abstrair e parametrizar os papéis da aplicação, para que haja o mínimo de alterações em código na eventualidade de surgir necessidade de novos papéis no futuro. Temos nesta tabela, a chave primária id e o campo name que contém uma identificação em texto do papel.

```
CREATE TABLE IF NOT EXISTS `role` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` text NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Figura 11. Tabela *role* do *IF-KEY*

A tabela user (do inglês, usuário), exposta na Figura 12, contém as informações dos usuários que são pertinentes ao funcionamento do *IF-KEY*, como a identificação dos usuários interna e externamente, por exemplo, em outros sistemas que o usuário possa utilizar no Instituto. Sendo assim a tabela contém a chave primária id, nome, a matrícula no IFSP, representada pelo campo ifsp_id e e-mail.

```
CREATE TABLE IF NOT EXISTS `user` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` text NOT NULL,
  `ifsp_id` text NOT NULL,
  `email` text NOT NULL,
  `password` text NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Figura 12. Tabela *user* do *IF-KEY*

A senha será salva no formato Bcrypt [Hcode Treinamentos], que é uma técnica de adicionar ao hash de senha, um código aleatório, reforçando assim sua segurança.

Para associar usuários aos seus devidos papéis, temos a tabela associativa user_role , apresentada na Figura 13. Ela representa o relacionamento muitos-para-muitos entre as respectivas tabelas.

```sql
CREATE TABLE IF NOT EXISTS `user_role` (
  `id` int NOT NULL AUTO_INCREMENT,
  `user_id` int NOT NULL,
  `role_id` int NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_user_user_role` (`user_id`),
  KEY `fk_role_user_role` (`role_id`),
  CONSTRAINT `fk_role_user_role`
  FOREIGN KEY (`role_id`) REFERENCES `role` (`id`),
  CONSTRAINT `fk_user_user_role`
  FOREIGN KEY (`user_id`) REFERENCES `user` (`id`)
) ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Figura 13. Tabela associativa *user-role* do *IF-KEY*

A tabela reservation , explicitada na Figura 14, contendo as informações pertinentes as reservas realizadas pelos usuários. Mais especificamente, contendo a chave primária id, as chaves estrangeiras reservee_id , representando o usuário que realizou a reserva, e room_id representando a sala reservada, bem como os dias e horas de início e fim da reserva. Além disso, a coluna reservation_code deve conter um UUID que será utilizado na identificação da reserva através de QR Code.

```sql
CREATE TABLE `reservation` (
    `id` int NOT NULL AUTO_INCREMENT,
    `reservee_id` int NOT NULL,
    `room_id` int NOT NULL,
    `start` datetime NOT NULL,
    `end` datetime NOT NULL,
    `subject` text NOT NULL,
    `course` text NOT NULL,
    PRIMARY KEY (`id`),
    KEY `fk_reservee_id` (`reservee_id`),
    KEY `fk_room_id` (`room_id`),
    CONSTRAINT `fk_reservee_id`
    FOREIGN KEY (`reservee_id`) REFERENCES `user` (`id`),
    CONSTRAINT `fk_room_id`
    FOREIGN KEY (`room_id`) REFERENCES `room` (`id`)
) ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Figura 14. Tabela *reservation* do *IF-KEY*

5.8. Atividade 8: Desenvolvimento do *backend*

O código relativo ao *backend*, tem por objetivo implementar uma API REST que forneça as funcionalidades listadas na definição dos requisitos funcionais. Como por exemplo, a realização do *login*.

Essas funcionalidades são disponibilizadas através de *endpoints*, cujos parâmetros estarão contidos de acordo com os padrões do protocolo HTTP, isto é, em URIs (*Uniform Resource Identifier* - Identificador Uniforme de Recurso) ou no corpo das requisições ou respostas, que devem conter, como dito anteriormente, um documento JSON válido.

Tais considerações relativas ao formato JSON serão tratadas pelo *framework Express* e sua infraestrutura de *middlewares*. Em particular, utilizaremos o *middleware* que realiza análise sintática de documentos JSON.

Além disso, o código-fonte foi projetado considerando-se módulos ou camadas que realizam tarefas distintas. As camadas foram delimitadas em

- Repositórios
  - A camada de repositórios, consiste de componentes que realizam a comunicação com a base de dados através da biblioteca mysql/xdevapi do Node.JS e transforma os dados em um formato de fácil manipulação para o restante da aplicação.
- Serviços
  - A camada de serviços, por sua vez, realiza as validações necessárias para garantir o funcionamento esperado das funcionalidades do *IF-KEY*, desde os dados dos usuários durante seus cadastros até a garantia de que as salas estejam disponíveis para reserva em dias e horas requisitados, delegando para componentes da camada de repositórios a persistência das informações após todas as validações serem concluídas. Na Figura 15, podemos verificar a tela de Login e na Figura 16, a página de cadastro da aplicação *IF-KEY*.
- *Endpoints*
  - A camada de *endpoints*, efetua a serialização e deserialização das requisições HTTP, convertendo também documentos JSON recebidos, em objetos esperados pelos componentes da camada de serviços, e vice-versa, isto é, convertendo objetos gerados pela camada de serviços em documentos JSON para envio nas respostas HTTP.

5.8.1. *Endpoints*

Nesta seção descreveremos os onze endpoints disponíveis no IF-KEY. Estes utilizaram os verbos do protocolo HTTP (ex: GET, POST, PUT, DELETE).

GET

O *endpoint* abaixo utilizará o verbo GET, que retornará do servidor os dados solicitados.

- Consultar agenda das salas e laboratórios;
- Consultar chaves.

POST

A seguir temos os *endpoints* que utilizaram o verbo POST, este verbo enviará os novos dados no corpo da requisição, para o endpoint, que criará e armazenará os dados no servidor.

- Realizar login;
- Realizar cadastro do usuário;
- Realizar agendamentos;
- Adicionar chaves.

PUT

Em seguida temos os *endpoints* que utilizaram o verbo PUT, este verbo adereçará as informações à serem atualizados, para que sejam alteradas no servidor.

- Alterar agendamentos;
- Alterar chaves.

DELETE

Ademais, temos os *endpoints* que utilizaram o verbo DELETE, este verbo encaminhara o dado identificador para o endpoint, que excluirá os dados no servidor.

- Excluir agendamentos;
- Excluir chaves;
- Excluir usuários.

Abaixo temos um exemplo de *endpoint* disponível no *IF-KEY*. Desde seu URI, formato de uma requisição até o formato de uma resposta.

- **GET /user/{id}**
  Descrição: retorna o usuário cujo campo *id* corresponde ao requisitado.
  Requisição
  Formato: Query string
  Corpo: vazio
  Resposta
  Formato:
  - Caso exista um usuário com o *id* correspondente:
    Código 200
    Corpo:

```
{
    "id": <id>,
    "name": <nome do usuário>,
    "ifsp_id": <matrícula do usuário>,
    "email": <email do usuário>,
    "role": <papel do usuário>
}
```

  - Caso o *id* não corresponda a nenhum usuário registrado: Código 404
    Corpo: null

5.9. Atividade 9: Desenvolvimento do *frontend*

De maneira análoga ao *backend*, o *frontend* foi desenvolvido em camadas, sendo estas:

- Serviços
  - A camada de serviços é responsável pelo acesso aos endpoints HTTP definidos pelo *backend*. O acesso é feito através da biblioteca Axios que possui várias funcionalidades convenientes para a geração de requisições HTTP, bem como o consumo de respostas HTTP.
- *Views* (Componentes visuais)

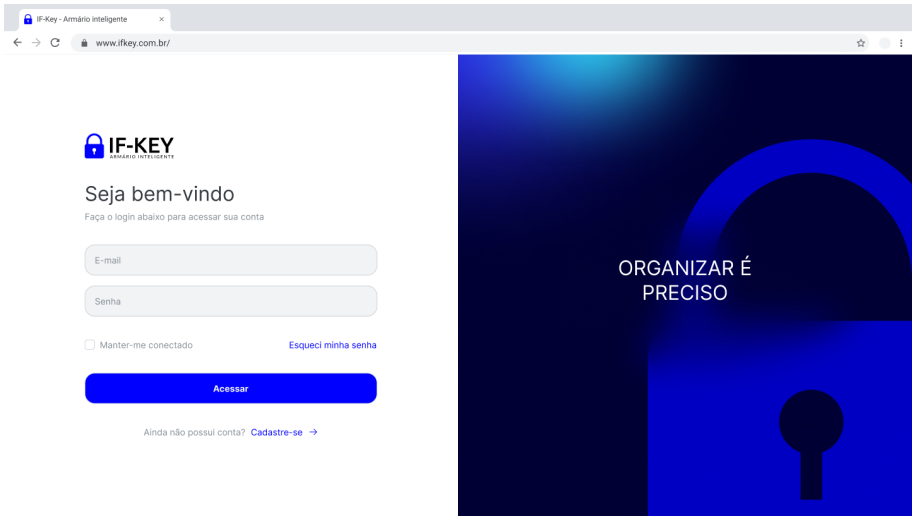

Figura 15. Página de login do sistema *IF-KEY*

– A camada de views foi desenvolvida utilizando componentes criados de acordo com os padrões da biblioteca *React*. Foram criados componentes para cada página disponível no *IF-KEY*, representados nas figuras a seguir.

Na Figura 15, apresenta-se a interface de *Login*. Caso o usuário não esteja cadastrado, ele pode ser redirecionado para a página de cadastro do usuário, como ilustrado na Figura 16, ao clicar no botão de 'Cadastre-se'. O usuário administrador tem acesso a uma página semelhante para cadastrar chaves/salas.

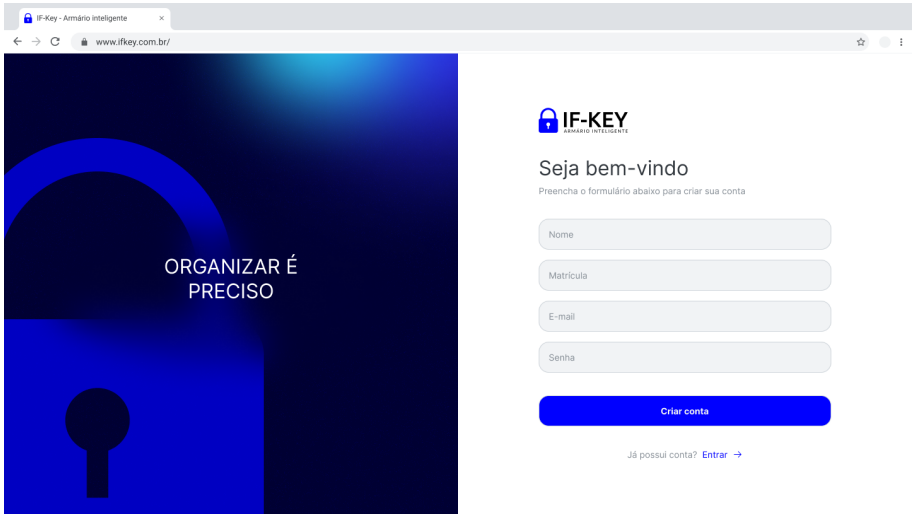

Figura 16. Página de cadastro de usuário *IF-KEY*

Nas Figuras 17 e 18, vemos a página inicial de reserva de salas/chaves sem filtragem e com filtragem aplicada. E por fim, na Figura 19, vemos a página de sucesso ao realizar uma reserva na aplicação *IF-KEY*.

Cada uma destas páginas possui subcomponentes *React* que realizam tarefas específicas como formulário para coletar informações do usuário sendo cadastrado, tabela

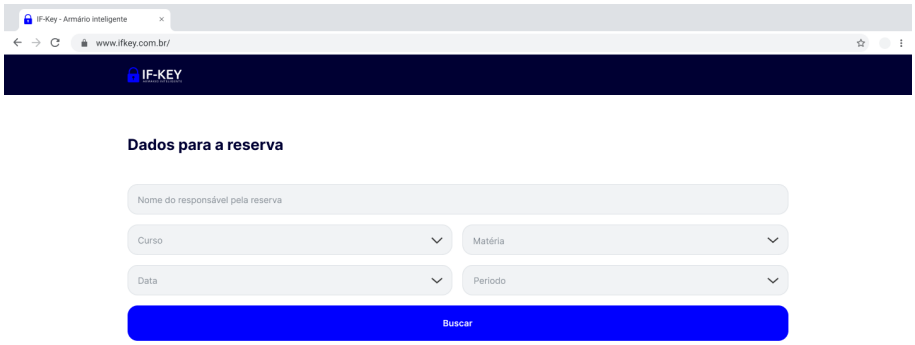

Figura 17. Página de reserva de chaves/salas sem filtros *IF-KEY*

de usuários já cadastrados. Esses subcomponentes fazem uso da camada de serviços para obter informações contidas no *backend* bem como no sentido contrário, isto é, preparando dados, provenientes do usuário através componentes visuais, para envio e gravação no *backend*.

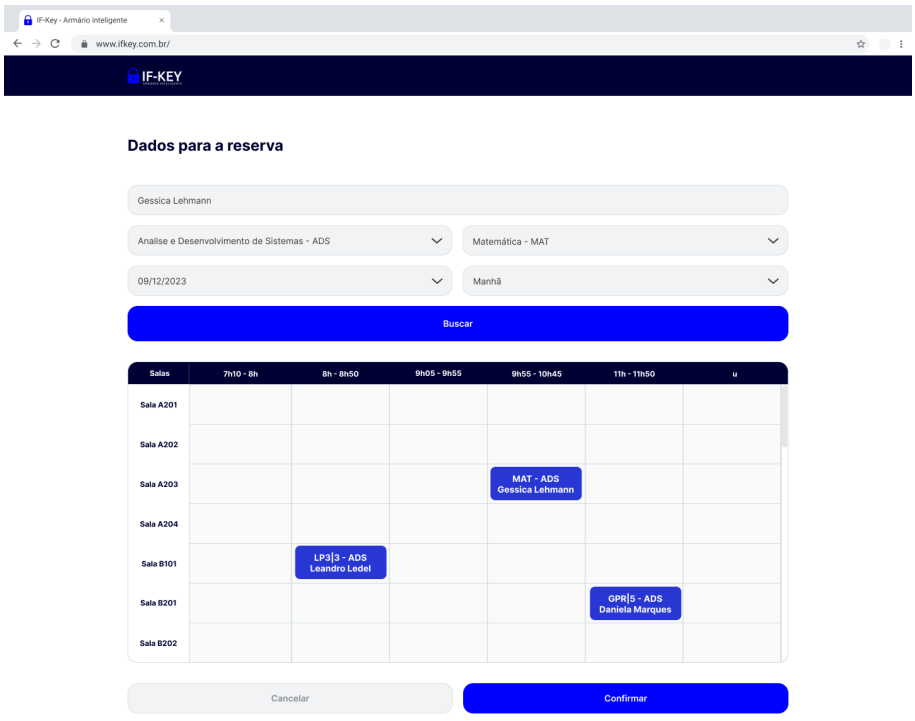

Figura 18. Página de reserva de chaves/salas com filtros *IF-KEY*

O *IF-KEY* é uma aplicação web com interface responsiva, ou seja, foi planejado para ser adaptável para tablets, computadores e *smartphones*. O sistema foi projetado para ser de fácil usabilidade e entendimento, sendo assim foi utilizada uma interface simples, com cores que refletissem serenidade e profissionalismo ao usuário.

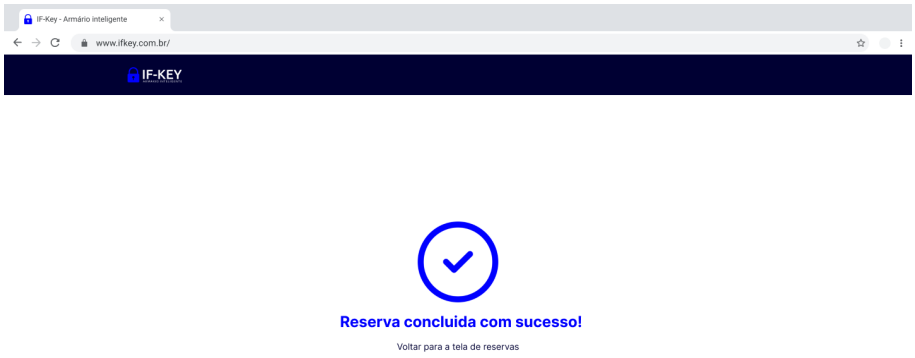

Figura 19. Página com mensagem de sucesso *IF-KEY*

6. Conclusão e trabalhos futuros

O objetivo deste sistema foi de melhorar o controle de usuários aos laboratórios e salas de aula do IFSP-HTO. Através da implementação do *software*, da inserção de QR-Code impresso nas chaves do instituto e com *link* que redireciona o usuário ao *IF-KEY*, o usuário foi capaz de criar sua conta e logar no sistema para consultar chaves ou agendar a utilização de uma sala ou laboratório do Instituto.

O sistema demonstrou ter atingido os seus objetivos do ponto de vista da implementação. Como trabalho futuro, seria interessante a implementação do controle das chaves através de *tag (RFID)* para que seja feito o rastreio das chaves.

Este projeto possibilitou a expansão de conhecimentos sobre o que foi ensinado ao decorrer do curso em diversas matérias e também possibilitou a aprendizagem de novas tecnologias, disponíveis no mercado. Além do mais, foram facultados neste artigo, conhecimentos adquiridos durante o período de estágio da autora.

Referências

BASSETT, L. (2015). *Introdução ao JSON: Um guia para JSON que vai direto ao ponto*. Novatec Editora.

Docker, Inc. Docker. `https://docs.docker.com/`. Acessado em: 04 dez. 2023.

Equipe de Desenvolvimento do React. React.a javascript library for building user interfaces. `https://legacy.reactjs.org/`. Acessado em: 10 feb. 2024.

FIELDING, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, Universidade da Califórnia, Irvine.

FLATSCHART, F. (2011). *HTML 5: embarque imediato*. BRASPORT.

Hcode Treinamentos. Criptografia em node js com bcrypt. `https://hcode.com.br/blog/criptografia-em-node-js-com-a-lib-bcrypt`. Acessado em: 04 dez. 2023.

KIESSLING, M. (2012). *The Node Beginner Book: A Comprehensive Node.js Tutorial*. Leanpub.

LAUDON, K. and LAUDON, J. (2014). *Sistema de Informação*. Pearson Education do Brasil, 11ª edition.

LEAL, J. and GUEDES, G. (2017). *Finder: Aplicativo de Achados e Perdidos*. Instituto Federal de São Paulo (IFSP) Campus Hortolândia, Caixa Postal 15.064 – 91.501-970 – Hortolândia – SP – Brasil.

MANZANO, J. and TOLEDO, S. (2009). *Guia de Orientação e Desenvolvimento de Sites - HTML, XHTML, CSS e JavaScript/Jscript*. Editora Érica, 4ª edition.

NIEDERAUER, J. and PRATES, R. (2005). *MySQL 5 - Guia de Consulta Rápida*. Novatec.

Open Container Initiative. Open container initiative. `https://opencontainers.org/`. Acessado em: 04 dez. 2023.

PINHEIRO, J. (2006). *Identificação por Radiofrequência: Aplicações e Vulnerabilidades da Tecnologia RFID*. Cadernos UniFOA, 2ª edition.

PUGA, S., FRANÇA, E., and GOYA, M. (2013). *Banco de Dados: Implementação em SQL, PL/SQL e Oracle 11g*. Editora Pearson, 1ª edition.

Red Hat, Inc. O que é orquestração de containers? `https://www.redhat.com/pt-br/topics/containers/what-is-container-orchestration`. Acessado em: 04 dez. 2023.

SEGURADO, V. (2017). *Projeto de Interface com o Usuário*. Editora Pearson.

SOMMERVILLE, I. (2011). *Engenharia de Software*. Editora Pearson Education do Brasil, 9ª edition.

Sunny Attic SOFTWARE. Easy qr. `https://play.google.com/store/apps/details?id=com.sunnyatticsoftware.easyqr&hl=pt_BR&gl=US`. Acessado em: 29 jul. 2023.

Documento Digitalizado Público

Artigo final de TCC da aluna Gessica S. S. Lehmann

Assunto: Artigo final de TCC da aluna Gessica S. S. Lehmann
Assinado por: Leandro Ledel
Tipo do Documento: Relatório
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Cópia Simples

Documento assinado eletronicamente por:

- **Leandro Camara Ledel**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 13/03/2024 18:25:14.

Este documento foi armazenado no SUAP em 13/03/2024. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1610925
Código de Autenticação: eda187d0aa