UNIVERSIDADE FEDERAL DO CEARÁ
INSTITUTO UFC VIRTUAL
SISTEMAS E MÍDIAS DIGITAIS

IAGO GOMES BARRETO

APLICATIVO PAAP: AUTOMATIZAÇÃO DA GESTÃO DOS PROCESSOS DO PROGRAMA DE FORMAÇÃO À DOCÊNCIA SUPERIOR DA UFC

FORTALEZA
2022

IAGO GOMES BARRETO

APLICATIVO PAAP: AUTOMATIZAÇÃO DA GESTÃO DOS PROCESSOS DO PROGRAMA DE FORMAÇÃO À DOCÊNCIA SUPERIOR DA UFC

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas e Mídias Digitais do Instituto Universidade Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de Bacharel em Sistemas e Mídias Digitais.

Orientadora: Prof. Dra. Maria de Fátima Costa de Souza

FORTALEZA
2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

B262a Barreto, Iago Gomes.
Aplicativo PAAP: Automatização da Gestão dos Processos do Programa de Formação à Docência Superior da UFC / Iago Gomes Barreto. – 2022.
59 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Instituto UFC Virtual, Curso de Sistemas e Mídias Digitais, Fortaleza, 2022.
Orientação: Profa. Dra. Maria de Fátima Costa de Souza.

1. Aplicativo PAAP. 2. Automatização da gestão. 3. Formação continuada. 4. Educação superior. I. Título.
CDD 302.23

IAGO GOMES BARRETO

APLICATIVO PAAP: AUTOMATIZAÇÃO DA GESTÃO DOS PROCESSOS DO PROGRAMA DE FORMAÇÃO À DOCÊNCIA SUPERIOR DA UFC

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas e Mídias Digitais do Instituto Universidade Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de Bacharel em Sistemas e Mídias Digitais.

Aprovada em: 17 de Fevereiro de 2022

BANCA EXAMINADORA

_____

Profa. Dra. Maria de Fátima Costa de Souza (Orientadora)
Universidade Federal do Ceará (UFC)

_____

Profa. Dra. Ticiana Linhares Coelho da Silva
Universidade Federal do Ceará (UFC)

_____

Ma. Adriana Madja dos Santos Feitosa
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

À minha família, minha mãe, Francisca Cicera Gomes Barreto, meu pai, Antônio José Rocha Barreto , pelo apoio, dedicação e confiança.

Aos meus colegas e amigos que fiz durante esta caminhada no curso.

Aos professores, coordenadores e servidores técnico-administrativos do curso de Sistemas e Mídias Digitais, pela experiência de convívio, tornando o nosso curso um lugar prazeroso de se estar, nossa segunda casa.

A minha orientadora, professora Maria de Fátima de Costa Souza, pela parceria e paciência que teve comigo ao longo deste trabalho.

A Universidade Federal do Ceará, em especial a unidade da EIDEIA e todos seus servidores, onde tive minha primeira oportunidade profissional, e pude desempenhar meus conhecimentos de forma prática.

RESUMO

Muitas instituições sofrem com o excesso de trabalho repetitivo e acabam precisando lidar com grandes volumes de dados de maneira manual. Com o apoio da tecnologia nos últimos anos, muitos processos tradicionalmente manuais entraram na esteira da automatização por programas de computador ou dispositivos móveis, exigindo uma nova abordagem na execução de diferentes processos. Este trabalho tem por objetivo descrever e documentar o processo de desenvolvimento de uma solução para apoiar a gestão nas atividades de formação colaborativa para os docentes oferecidas pelo Programa de Formação para a Docência no Ensino Superior da Universidade Federal do Ceará, denominado atualmente Programa de Apoio e Acompanhamento Pedagógico (PAAP). Trata-se de uma pesquisa aplicada com o uso da observação e das entrevistas informais. Utiliza-se também de estudos bibliográficos sobre os elementos fundamentais da arquitetura de software. O projeto reúne detalhes sobre as etapas de programação do aplicativo, estimulo para a escolha das tecnologias, arquitetura do projeto, diagramas de casos de uso, metodologia aplicada e aplicação das principais funcionalidades, além de projetos futuros para o aplicativo.

Palavras-chave: Aplicativo PAAP; Automatização da gestão; Formação continuada. Educação superior.

ABSTRACT

Many repetitive institutions suffer from excessive repetitive work and end up having to deal with volumes of manual data. With technology support of manual processing processes or processes, automated device processing processes and manual device processing processes, manual device processing processes, tool processing processes, different processing processes. The objective of this work is to define and document the management process for the management of collaborative training solutions of the training program for Teaching in Higher Education designated by the Training Program for Teaching in Higher Education on Federal University of Ceara, currently called Programa de Apoio e Acompanhamento Pedagógico (PAAP). This is an applied research using observation and informal interviews. It also uses bibliographic studies on the fundamental elements of software architecture. The project gathers details about the application's programming steps, and encourages the choice of technologies, project architecture, use case diagrams, applied methodology and application of the main functionalities, in addition to future projects for the application.

Keywords: PAAP application. Management automation; Continuing training. College education.

LISTA DE FIGURAS

Figura 1 — Exemplo de arquitetura cliente servidor 16
Figura 2 — Modelo MVC 17
Figura 3 — Exemplos de verbos HTTP utilizando REST 18
Figura 4 — Modelagem do banco de dados 28
Figura 5 — Endpoints 29
Figura 6 — Services 29
Figura 7 — Repositories 30
Figura 8 — Models 30
Figura 9 — DTOs 31
Figura 10 — Utils 31
Figura 11 — Authentication 31
Figura 12 — Módulo de Evento 38
Figura 13 — Providers 39
Figura 14 — Routes 39
Figura 15 — Themes 39
Figura 16 — Tela de login e respectivos módulos de administrador e docente 40
Figura 17 — Telas de usuário e categoria 41
Figura 18 — Telas de cadastro 42
Figura 19 — Telas de mais detalhes 43
Figura 20 — Tela de eventos para docente 44
Figura 21 — Perfil de usuário 44
Figura 22 — Tela de configurações 45
Figura 23 — Diagrama de Estados Tela de Inscrição 46
Figura 24 — Estado não inscrito 47
Figura 25 — Estado inscrito 47
Figura 26 — Estado presente 48

LISTA DE TABELAS

Tabela 1 — Relação Atividade x Tempo 23
Tabela 2 — Métodos de Usuário 32
Tabela 3 — Métodos de Evento 32

LISTA DE CÓDIGOS FONTE

Código 1 — Método GenerateQrCode 33
Código 2 — Método de inscrição 34
Código 3 — Método de remoção de inscrição 34
Código 4 — Método de registro de frequência e carga horária 36
Código 5 — Método de Download de certificado 37

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CASa	Programa de Cooperação e Aprendizagem significativa
CSS	Cascading Style Sheets
DTO	Data Transfer Object
EIDEIA	Escola Integrada de Desenvolvimento e Inovação Acadêmica
HTML	Hypertext Markup Language
JRXML	Jasper Reports Extensible Markup Language
JSON	JavaScript Object Notation
MVC	Model View Controller
PAAP	Programa de apoio e Acompanhamento Pedagógico
REST	Representational State Transfer
SGBD	Sistema de gerenciamento de banco de dados
SIAPE	Sistema Integrado de Administração de Recursos Humanos
XML	Extensible Markup Language

SUMÁRIO

1 **INTRODUÇÃO** 13
1.1 **Objetivo** 14
1.1.1 **Geral** 14
1.1.2 **Específicos** 14
2 **REFERENCIAL TEÓRICO** 15
2.1 **Desenvolvimento de Software** 15
2.1.1 **Arquitetura de Software** 15
2.1.2 **Padrão arquitetural** 15
2.1.3 **Arquitetura REST** 17
2.1.4 **Desenvolvimento multiplataforma** 19
2.1.5 **Flutter** 20
2.2 **Prototipação** 20
2.2.1 **Figma** 21
2.3 **Dispositivos móveis com sistema Android** 21
2.4 **Trabalhos relacionados** 22
2.4.1 **SisEventos** 22
2.4.2 **IPresence** 22
3 **METODOLOGIA** 23
3.1 **Cenário** 23
3.2 **Participantes** 23
3.3 **Instrumentos de coleta de dados** 23
3.4 **Procedimentos** 23
3.5 **Análise dos dados** 24
3.5.1 **Requisitos Funcionais** 24
3.5.2 **Requisitos Não-Funcionais** 25
3.5.3 **Especificação de requisitos do sistema** 25
4 **ESTRUTURA DO PROJETO** 27
4.1 **API** 27
4.1.1 **Modelagem dos dados** 27
4.1.2 **Estrutura organizacional da API** 29
4.1.2.1 **Endpoints** 29
4.1.2.2 **Services** 29
4.1.2.3 **Repositories** 30
4.1.2.4 **Models** 30
4.1.2.5 **DTOs** 30
4.1.2.6 **Utils** 31

4.1.2.7 **Authentication** 31
4.1.3 **Métodos da API** 32
4.1.4 **Codificação dos Métodos** 33
4.1.4.1 **Criação de evento** 33
4.1.4.2 **Registro e remoção de inscrição** 33
4.1.4.3 **Registro de frequência e carga horária** 35
4.1.4.4 **Download de certificado** 36
4.2 **Aplicativo** 37
4.2.1 **Estrutura Organizacional do Aplicativo** 38
4.2.1.1 **Providers** 38
4.2.1.2 **Routes** 39
4.2.1.3 **Themes** 39
5 **DESENVOLVIMENTO** 40
5.1 **Aspectos Funcionais** 40
5.1.1 **Login e telas iniciais** 40
5.1.2 **Módulo Administrador** 41
5.1.3 **Telas de Cadastro** 41
5.1.4 **Telas de mais detalhes** 42
5.1.5 **Módulo Usuário** 43
5.1.6 **Inscrição e presença no evento** 45
6 **CONCLUSÃO** 49
REFERÊNCIAS 50
ANEXO A — DIAGRAMAS DE CASOS DE USO 52
ANEXO B — ESPECIFICAÇÃO DOS CASOS DE USO 55

1 **INTRODUÇÃO**

A Universidade Federal do Ceará (UFC) lançou, a partir de 2009, o programa de cooperação e aprendizagem significativa (CASa) com o intuito de formar professores, recém ingressos na UFC, através de diversos eventos disponibilizados pelo programa, como seminários pedagógicos, laboratório de práticas docentes, aulas sobre didática, acessibilidade, entre outros. A proposta era auxiliar o docente a desenvolver uma base formativa, reflexiva e humana, durante seu estágio probatório nos campis da Capital e do interior da Universidade Federal do Ceará: Fortaleza, Sobral, Itapagé, Crateús, Russas e Quixadá. A participação do programa passou a ser obrigatório para os docentes ingressos com a lei LEI Nº 12.772, de 28 de Dezembro de 2012, sendo dever do participante cumprir sessenta e quatro horas no decorrer de três anos[1]. Em 2016, o Programa de Formação para a Docência no Ensino Superior passou a fazer parte da Coordenadoria de Inovação e Desenvolvimento Acadêmico (COIDEA) da Escola Integrada de Desenvolvimento e Inovação Acadêmica (EIDEIA). No início do ano de 2020, passou novamente por formulações e atualmente se chama Programa de Apoio e Acompanhamento Pedagógico (PAAP), mas ainda continua com o mesmo intuito de promoção de ações formativas para os professores, visando o aperfeiçoamento das práticas didático-pedagógicas na UFC, em especial, os que estão em estágio probatório. Só no ano de 2019 o programa contabilizou 290 docentes participantes nos seus eventos, e 126 professores concluintes do programa no mesmo ano, de acordo com relatório da Coordenadoria do Programa.

Passados mais de uma década do programa desde o seu início em 2009, o processo para o docente acompanhar sua carga horária de participação nos eventos da formação, bem como a solicitação de certificados dos eventos que participou, permanece manual. O processo consiste das seguintes etapas: o docente entra em contato com a secretaria do programa via e-mail, ou diretamente na unidade. Os servidores estipulam um prazo para localização dos arquivos, que geralmente estão distribuídos entre diversas pastas, planilhas e por fim, enviam as informações solicitadas para o e-mail do docente. Ao final do 3º ano, o docente necessita fazer um relatório informando as ações do programa que participou e por isso era necessário ter o registro de todas as atividades. Todo esse controle feito de forma manual acarreta uma sobrecarga tanto para o professor recém ingresso, quanto para o programa que realiza o gerenciamento das participações desses docentes de forma mecânica e morosa.

Sendo assim, a questão problema é definida em: como automatizar o processo de gerência das frequências e histórico de participação dos docentes nos eventos que participam do projeto PAAP?

Neste contexto, este projeto tem como objetivo geral o desenvolvimento de um sistema de informação que automatize, tanto o controle de frequência quanto o processo de

1 Resolução N° 04/CEPE. Disponível em: https://www.ufc.br/images/_files/a_universidade/cepe/resolucao_cepe_2016/resolucao04_cepe_2016.pdf. Acesso em: 23 de Fevereiro 2022.

emissão de certificados reduzindo a carga de trabalho manual e otimizando as informações, a partir do smartphone.

A escolha por um sistema para dispositivos móveis, especificamente *smartphone*, pode ser justificada porque a internet móvel vem fazendo cada vez mais parte do dia a dia do cidadão e de acordo com pesquisa feita pela TIC Domicílios(2018), cerca de três em cada quatro brasileiros possuem acesso a internet, e o celular é o principal meio de acesso no Brasil (IBGE, 2018). Levando em consideração essas informações para as áreas que prestam serviços, torna-se essencial a introdução do maior número de processos ou ferramentas informatizados, que alavancam a produtividade, gerando a fluidez dos dados acessados e automatizando processos, removendo a sobrecarga de trabalho manual do servidor.

Intenciona-se com esse sistema desenvolver um software de apoio a gestão, definido em módulos de administrador e usuário, em uma versão mobile, onde os próprios participantes registram sua presença, através da interação com QRCode no local do evento, e as informações são atualizadas no momento da interação. O sistema permite ao docente acompanhar as horas restantes para a conclusão do programa, ver o histórico de eventos que já participou, sem a necessidade de entrar em contato com os servidores, removendo a sobrecarga de trabalho repetitivo dos mesmos.

1.1 **Objetivo**

1.1.1 **Geral**

Desenvolver um software de apoio à gestão para os docentes recém ingressos na UFC, definido em módulos de administrador e usuário, para automatização das frequências dos participantes e acesso ao histórico dos eventos do Programa PAAP.

1.1.2 **Específicos**

- Identificar as necessidades do PAAP no que concerne ao controle da gestão dos docentes nos eventos de formação.
- Descrever os requisitos funcionais e não funcionais para o desenvolvimento de um aplicativo.
- Estruturar uma aplicação móvel de gerenciamento de frequência e controle de eventos.
- Desenvolver um aplicativo para automatizar o controle de frequência e eventos.

2 **REFERENCIAL TEÓRICO**

Nesta seção, serão apresentadas as justificativas teóricas para a escolha no desenvolvimento de uma aplicação móvel bem como os conceitos necessários para a implementação do mesmo . É relatado o processo de desenvolvimento de sistemas, seu histórico e os elementos fundamentais que os compõem.

2.1 **Desenvolvimento de Software**

2.1.1 **Arquitetura de Software**

Para desenvolver um software, primeiramente é necessário um levantamento das necessidades com base em análise e diretrizes. A arquitetura de software existe para conduzir o desenvolvimento com base em experimentos e regras já consolidadas. Segundo Shaw e Garlan (1994), a arquitetura define o que é o sistema em termos de elementos computacionais, juntamente com os relacionamentos entre estes componentes, os modelos que guiam a sua composição e as restrições. Segundo Sommerville (2011, p.18), o processo de arquitetura de software é realizado através dos seguintes pontos:

- **Especificação:** Corresponde a parte mais gerencial. Realiza a tradução da necessidade ou requisito para uma descrição da funcionalidade a ser executada.
- **Implementação de software:** Relacionada a transcrição dos requisitos para os componentes de software, que atendam as necessidades especificadas. Adequa-se às regras de negócio a um padrão arquitetural existente que atinja os objetivos especificados. Deve ser produzido com base no projeto de software.
- **Validação de software:** Corresponde a realização de testes, com a finalidade de validar se os requisitos atendem as especificações.
- **Evolução do software:** A importância da manutenção e constante evolução para facilitar possíveis alterações solicitadas.

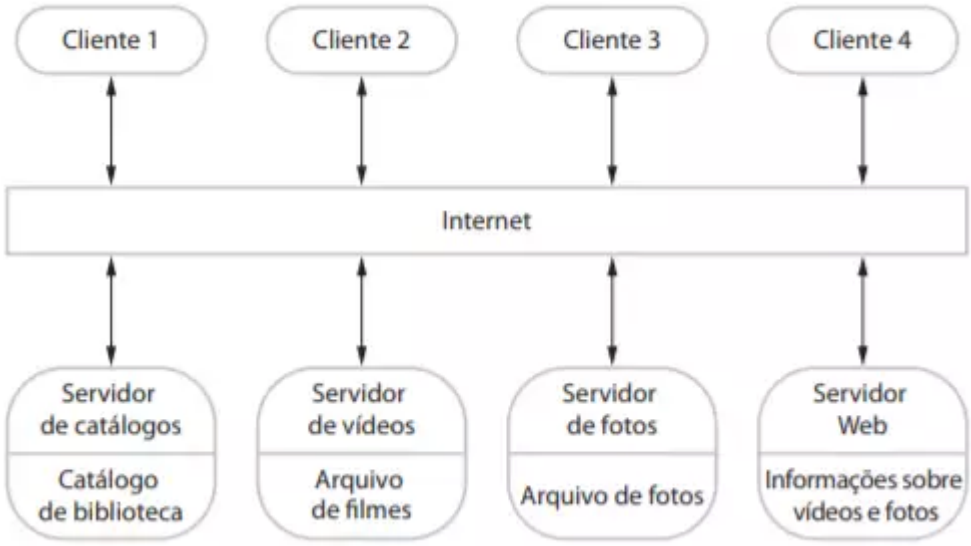
2.1.2 **Padrão arquitetural**

> Os padrões de software começaram a ser mapeados na década de 90 e consolidaram-se como uma maneira popular e complementar de se descrever e expandir projetos de software, capturando e nomeando técnicas que provaram funcionar (Azevedo, 2014, p.23)

Os padrões arquiteturais definem regras e diretrizes para organizar a relação entre os componentes do sistema, proporcionam uma estrutura pré-definida com base nas necessidades do software. São consideradas soluções para problemas recorrentes ao desenvolver software. Além disso, com um padrão de arquitetura definido, o entendimento acerca do projeto torna-se mais compreensível para desenvolvedores e gerentes. Os dois padrões arquiteturais mais conhecidos são citados por Sommerville (2011):

- **Arquitetura cliente-servidor**: Estrutura organizacional baseada em serviços combinando dados do cliente e do servidor. É necessário da parte do cliente acesso a redes de computadores. Funciona através de solicitações e respostas. Muitos aplicativos operam de acordo com o padrão cliente/servidor. Os programas (clientes) solicitam um serviço ao servidor, e este concentra toda a lógica necessária para o processamento dos dados. É um dos padrões mais difundidos e que está habitualmente na rotina das pessoas, como em aplicativos bancários e serviços de e-mail.

Figura 1 — Exemplo de arquitetura cliente servidor

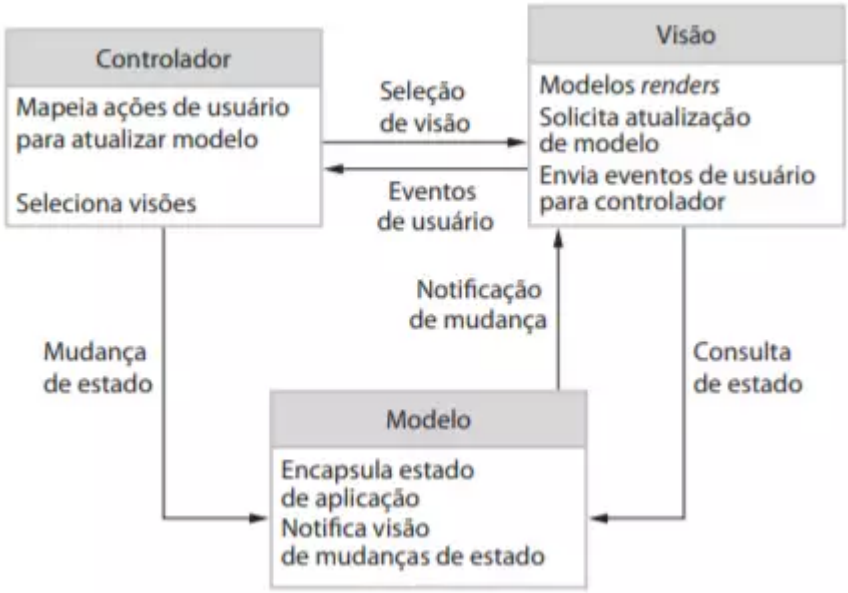


Fonte: SOMMERVILLE (2011, p. 114)

Arquitetura MVC: Distribuído em três camadas (Modelo, Visão e Controle), este padrão é um dos mais comuns padrões para web, provendo um modelo interativo de sistema. O Modelo lida com a manipulação dos dados, seja leitura ou escrita. A Visão é responsável pela exibição dos dados processados, onde acontece a interação com o usuário, enquanto o

Controle lida com as requisições realizadas pelos usuários, sendo um mediador, responsável por resgatar aquilo que foi solicitado e enviando aquilo que foi processado como resposta.

Figura 2 — Modelo MVC



Fonte: SOMMERVILLE (2011, p. 109)

Existem também arquiteturas que são oriundas da junção de outras, como no caso da arquitetura REST, que veremos na seção a seguir.

2.1.3 **Arquitetura REST**

Transferência de Estado Representacional, REST, é um estilo arquitetural híbrido para sistemas hipermídia distribuídos derivado dos estilos arquiteturais cliente-servidor e sem estado (TOBALDINI, 2008). Utiliza-se do protocolo HTTP que é baseado em requisições e respostas, para criar, atualizar, pesquisar, executar e remover recursos. Brito(2020), descreve cinco restrições para o serviço REST:

- **1º restrição: Interface uniforme:** Onde cada método é associado a uma função. Utilizando-se dos mesmos métodos HTTP.

Figura 3 — Exemplos de verbos HTTP utilizando REST

Verbo HTTP	Coleção de recursos. Ex: cliente
GET	Lê recuso(s). Exemplo: ***/cliente - Recupera uma lista de clientes.*** ***/cliente/33 - Recupera um cliente específico.***
POST	Cria recurso(s). Exemplo: ***/cliente - Cria um novo cliente.***
PUT	Atualiza todos os dados do(s) recurso(s). Exemplo: ***/cliente/33 - Atualiza todos os atributos do cliente número 33.***
PATCH	Atualiza atributos específicos(s) do recurso(s). Exemplo: ***/cliente/33 - Atualiza parcialmente dos dados do cliente número 33.***
DELETE	Remove recurso(s). Exemplo: ***/cliente/33 - Remove o cliente número 33.***

Fonte: Matera

- **2º restrição: Sem Estado:** As comunicações devem acontecer independente do estado do servidor, sem armazenamento de sessão, por exemplo, e independentes entre as próprias requisições. Ou seja, uma requisição não deve utilizar-se dos recursos de outra. (COUTO, 2009)

- **3º restrição: Cacheavel:** É possível utilizar-se dos benefícios do cache em REST, visando a melhoria na performance. Lima(2018), afirma que "o propósito é acelerar a busca de dados que são muito utilizados e poupar a utilização de recursos de um servidor." Mas, cabe a cada software analisar a adoção ou não de cache.

- **4º restrição: Cliente-servidor:** Onde os conceitos são separados, a aplicação REST, não precisa saber como a interface está implementada, ela tem apenas o dever de prover os dados de maneira representacional dos recursos. O cliente abstém-se da implementação e lógica dos dados, e lida apenas com as informações representacionais (SAMPAIO, 2015).

- **5º restrição: Código sob demanda:** Tobaldini(2008), explica que "A utilização de código sob demanda também pode caracterizar uma aplicação REST com o objetivo de simplificar e ampliar as capacidades dos clientes.". Isso estende características do servidor para o cliente, porém é um atributo opcional.

Com um serviço REST estamos aptos a trabalhar de maneira multiplataforma, uma vez que a camada de visão não estará acoplada no próprio servidor, como acontece na arquitetura MVC. Também possui comunicação através de objetos representacionais, como XML ou

JSON, que a maioria dos sites utilizam. Souza(2020), ressalta que esta tecnologia permite uma maior escalabilidade, provendo uma estrutura que se adequa ao desenvolvimento multiplataforma, permitindo a integração com outros sistemas, como redes sociais e sistemas de pagamento.

Essas características permitem uma melhor interoperabilidade entre outras aplicações, caso necessário. Pode auxiliar, por exemplo, na produção de uma versão web e mobile utilizando os mesmos dados sem a necessidade de codificar novamente a lógica de negócios. Com a utilização da arquitetura REST será possível desenvolver a API da aplicação, que é definida como a abstração da arquitetura web que oferece funções, recursos ou dados composta por regras de negócio que padronizam a criação de projetos com interface bem definidas.

2.1.4 **Desenvolvimento multiplataforma**

No contexto dos dispositivos móveis, as plataformas dominantes são os sistemas Android, da *Google* e o IOS da *Apple*. O mercado mobile está dividido entre os dois sistemas, onde cada um possui arquiteturas e características distintas, e estimulam conhecimento para desenvolver aplicações de forma nativa, que são específicas para cada sistema operacional. Uma aplicação nativa é um software que é desenvolvido pela linguagem exclusiva de cada sistema operacional, possuindo seu próprio ferramental e características de interface de usuário (JUNIOR e MERCADO, 2018). Porém, manter diferentes equipes de desenvolvimento para cada sistema operacional, acarretam em maior custo, mais tempo para a compreensão de múltiplas linguagens e maior complexidade no gerenciamento do projeto.

O desenvolvimento multiplataforma surgiu como uma alternativa ao desenvolvimento nativo. A produção é feita utilizando conhecimentos tecnológicos comuns da web, como por exemplo, HTML, CSS e Javascript. Segundo Muller e Soares (2019), as ferramentas multiplataforma ou *cross-plataform* são utilizadas para desenvolver software em múltiplas arquiteturas de computadores ou sistemas operacionais. O desenvolvimento de software móvel torna-se mais eficiente, fazendo o uso da mesma base de código, que pode ser viabilizado para os diversos sistemas operacionais, através das ferramentas disponibilizadas pelos frameworks multiplataformas, como F*lutter, React ou Ionic*.

2.1.5 **Flutter**

O *Flutter* é um kit de ferramentas de interface de usuário de código aberto para a construção de aplicativos móveis, web e desktop de alta qualidade. Foi criado pela *Google* e sua linguagem base é o *Dart*, que é fácil de aprender e possui muitas semelhanças com a linguagem mais popular para web, o javascript. Ao produzir um aplicativo com o framework, o código é compilado para a linguagem base do dispositivo alvo, sendo assim, não precisa de recursos de terceiros, como *plugins*, para acessar funcionalidades nativas do aparelho, gerando um melhor desempenho da aplicação. Suas principais características são:

- *Write Once, Run Everywhere*: O termo significa "escreva uma vez, execute em qualquer lugar". Trata-se justamente do objeto do desenvolvimento multiplataforma, com a mesma base de código fonte é possível produzir a aplicação para diversos sistemas.
- Possui maior desempenho em comparação com outros frameworks, pois seu código é compilado de forma nativa.
- Acesso direto a recursos nativos: Uma aplicação criada com *Flutter* possui acesso nativo a todos os recursos do dispositivo, como GPS, galeria, câmera e *WI-FI*. Dispensando o uso de *plugins* de terceiros.
- Foi encabeçado pelo *Google*, está em constante atualização e tem uma grande comunidade ativa.

Com base no exposto, o presente trabalho tem como objetivo utilizar o framework Flutter para gerar uma versão Android do sistema projetado, pois sendo um sistema livre, não tem restrições na geração do aplicativo. Outra vantagem quanto ao uso do framework mencionado é que utilizando a mesma base de código é possível produzir uma versão para *smartphones* IOS, desde que o desenvolvedor tenha um computador com sistema proprietário *MAC OS*, e a ferramenta de desenvolvimento *XCode*, para gerar a versão de produção.

2.2 **Prototipação**

Antes de desenvolver qualquer site ou aplicação móvel, é importante desenhar todo o seu funcionamento. Quando é iniciado um projeto, o primeiro passo deve ser estabelecer uma hierarquia na informação no conteúdo para poder transportá-lo visualmente. Desta forma, o cliente consegue compreender e visualizar como será estruturado na fase final (PAREDES, 2019). Para isso, existem diversas ferramentas como Axure, Sketch, Balsamiq, XD, Figma,

dentre outros que podem nos auxiliar. Nos softwares de prototipagem, são definidos pontos importantes como a navegação, posição de elementos e etc. Essas etapas facilitam a identificação dos aspectos que precisam ser corrigidos antes da fase de desenvolvimento e codificação da interface. Nesta seção, abordaremos uma ferramenta gratuita.

2.2.1 **Figma**

O Figma é uma ferramenta de design de interface e prototipação gratuita onde é possível fazer todo o trabalho através do navegador, sendo compatível com *Windows, Linux e Mac* (Figma, 2020). Na parte de prototipação, é possível selecionar entre alguns dispositivos e criar interações entre as páginas, deixando o teste ainda mais parecido com uma aplicação real, facilitando a visualização do produto final e tornando a apresentação do projeto mais espontânea. Seus benefícios são:

- Versionamento automático.
- Biblioteca compartilhada de componentes.
- Possui uma grande número de *plugins*, para adição de novas funcionalidades, dependendo do projeto.

2.3 **Dispositivos móveis com sistema Android**

Com a expansão da internet, entre os anos de 1970 e 1990, grande parte do acesso se dava pelo computador, graças ao sucesso dos computadores pessoais. Com a passagem dos anos, o poder de processamento em dispositivos menores cresceu, e passaram de um simples aparelho que efetuava ligações, para um hardware com sistema operacional. E aliando melhor desempenho, mobilidade e praticidade, o uso de smartphones evoluiu de forma exponencial. No ano de 2014, no Brasil, os smartphones superaram pela primeira vez a preferência dos usuários para o acesso a internet em relação ao computador (IBGE, 2018). Em 2016 a quantidade de aparelhos era em torno de 2,5 bilhões, e há indicativo de que em 2021, o uso de smartphones alcancem a marca de 4 bilhões em todo o mundo (SEBRAE, 2020). Levando em consideração a perspectiva futura, mais usuários estarão conectados a smartphones, gerando uma demanda maior por aplicativos.

Atualmente, o sistema operacional mais utilizado no Brasil é o Android, cobrindo a marca de 95% dos *smartphones* do Brasil (G1, 2019). Foi criado pela startup homônima Android Inc. em outubro de 2003. Em agosto de 2005 foi adquirida pela empresa *Google*, que

lançou em novembro de 2007, juntamente com a OHA, o sistema Android, *open-source* e baseado no *kernel* do *Linux* (Matos e Silva, 2016). Sua versão mais atual é a 12, e seu sucesso é impulsionado pela grande aceitação das empresas do ramo da tecnologia, que comercializam seus hardwares com o sistema operacional gratuito. A escolha por gerar uma aplicação na versão Android é feita em razão dos dados apresentados anteriormente.

2.4 **Trabalhos relacionados**

Esta seção tem como objetivo mostrar alguns produtos com proposta similares, que servem como exemplo para embasar funcionalidades para a aplicação.

2.4.1 **SisEventos**

Disponibilizado pela Universidade Regional do Cariri, o SisEventos[2] é um sistema web voltado para o gerenciamento de eventos científicos. Tem como ponto positivo diversas funcionalidades, onde é possível cadastrar eventos, palestras, oficinas, realizar inscrições online, emissão de certificados e registro de frequência de forma automatizada. Possui uma abordagem *all-in-one*, além de ser disponibilizado também em versão móvel (SisEventos, 2020). Apesar de ser um plataforma provida por uma universidade, o usuário tem total controle sobre os eventos que cadastra, e pode realizar os acompanhamento dos inscritos e resultados.

2.4.2 **IPresence**

Desenvolvido com objetivo de gerenciar as frequências de alunos de forma online, IPresence [3] é um aplicativo móvel que possui integração com um sistema de salas virtuais (SAV) da Universidade Federal do Rio Grande Do Sul, plataforma facilitadora da interação aluno-professor. Sua proposta é tornar mais transparente a contagem de frequência dos participantes em aula e reduzir o trabalho manual para efetuar o registro das mesmas no SAV (Heck,2013).

2 SisEventos. Disponível em http://siseventos.urca.br/ Acesso em 30 de Janeiro de 2022.
3 IPresence. Disponível em https://www.lume.ufrgs.br/bitstream/handle/10183/100288/000931702.pdf. Acesso em 30 de Janeiro de 2022.

3 **METODOLOGIA**

A metodologia será baseada na pesquisa aplicada, método cientifico que pretende produzir conhecimento através da prática, gerando soluções para problemas específicos, tendo como objetivo o desenvolvimento de um sistema para gerenciamento de frequências e acompanhamento da formação dos professores em estágio probatório na Universidade Federal do Ceará. Ela está organizada através das seguintes etapas:

3.1 **Cenário**

O local da análise foi na unidade do programa EIDEIA da Universidade Federal do Ceará entre os anos de 2019 a 2022.

3.2 **Participantes**

Coordenadora, secretária, auxiliar administrativo e um docente participante do programa CASa UFC.

3.3 **Instrumentos de coleta de dados**

Entrevista não estruturada (aberta e informal) e método de observação.

3.4 **Procedimentos**

Através do método de observação, foi acompanhado as atividades diárias dos participantes. A atividade da coordenadora é contatar palestrantes, professores para possíveis eventos, e o trabalho manual fica concentrado na secretária e o auxiliar administrativo. Além disso, foi necessário o comparecimeto em alguns eventos, para analisar como era feito o registro da presença dos participantes no evento.

Por meio de uma entrevista aberta e informal, a secretária relatava as atividades que realizava durante o dia na unidade do programa CASa, o tempo para concluí-las, quais problemas eram recorrentes ao atender o público, e como o programa funciona. Foi indagado a participante de que modo essas atividades poderiam ser otimizadas. Na tabela 1, são listadas as atividades e o tempo médio para cada tarefa:

Tabela 1 — Relação Atividade x Tempo (continua)

Atividade	**Tempo Médio (Em minutos)**
Repassar informações sobre evento via telefone	3

Tabela 1 — Relação Atividade x Tempo (conclusão)

Atividade	Tempo Médio (Em minutos)
Disponibilizar um novo evento	5
Cadastrar participante interessado no evento	2
Produzir lista de frequência de um evento	5
Coletar frequências no evento	9
Recuperar histórico de eventos de um participante	15
Contabilizar carga horária de um participante	15
Emitir certificado de participação em um evento	3
Emitir certificado de conclusão de um docente	5

Fonte: O Autor (2019)

3.5 **Análise dos dados**

A partir dos dados levantados das observações e entrevistas sobre o funcionamento das atividades do programa e apresentados na Tabela 1, foi identificado a necessidade da criação de um sistema de gerenciamento das atividades que ainda eram feitas manualmente. Dessa forma, é proposto neste projeto o desenvolvimento de um sistema móvel, com vista a otimizar as atividades da secretária da EIDEA, além de favorecer ao usuário docente uma maior praticidade no acompanhamento de sua formação no programa, que possui uma período de até três anos. Como forma de viabilizar a referida aplicação, foram listados abaixo um conjunto de requisitos (funcionais e não funcionais) que irão compor a mesma. Um maior detalhamento dos requisitos pode ser encontrado na seção 3.6.3.

3.5.1 **Requisitos Funcionais**

- **[RF001]** Acessar diferentes módulos. O sistema deve conter módulo administrador e usuário.
- **[RF002]** Armazenar dados dos usuários. O sistema deve conter os dados relevantes dos usuários.
- **[RF003]** Cadastrar eventos. O sistema deve ser capaz de cadastrar novos eventos.
- **[RF004]** Cadastrar usuários. O sistema deve ser capaz de cadastrar novos usuários.
- **[RF005]** Atualizar dados do usuário docente. O usuário deve ser capaz de atualizar seus dados.
- **[RF006]** Registrar presença. O sistema deve ser capaz de registrar a presença do

docente.

- **[RF007]** Contabilizar carga horária. O sistema deve adicionar a carga horária de um evento para o usuário.
- **[RF008]** Consultar inscritos. O sistema deve permitir ao usuário administrador visualizar os docentes inscritos em um evento.
- **[RF009]** Consultar presentes. O sistema deve permitir ao usuário administrador visualizar os usuários docentes que registraram presença em um evento.
- **[RF010]** Consultar a situação atual do docente. O sistema deve ser capaz de gerar status do usuário docente em relação ao programa.
- **[RF011]** Consultar histórico de eventos. O sistema deve permitir ao usuário docente visualizar os eventos ao qual ele esteve presente, através de um histórico.
- **[RF012]** Realizar *Download* de certificado. O sistema deve permitir ao usuário docente baixar o certificado de participação no evento ao qual ele esteve presente.

3.5.2 **Requisitos Não-Funcionais**

[RNF001] A regra de negócio deve ser programada em Java. A escolha da linguagem é justificada pela grande comunidade de desenvolvimento que a compõe, assim, caso ocorra algum problema no decorrer da produção, torna-se mais fácil encontrar uma solução, visto a dimensão da comunidade. Também é a linguagem ao qual o desenvolvedor da aplicação já possui familiaridade.

[RNF002] O aplicativo deve ser desenvolvido com ferramenta multiplataforma. Com ela, é possível reutilizar o mesmo código desenvolvido para o Android em outras plataformas. O desenvolvedor também possui mais conhecimento em linguagens voltadas para web, aos quais os frameworks híbridos possuem similaridades.

[RNF003] O sistema deve ter acesso à banco de dados. Necessário para armazenar os dados identificados nos requisitos funcionais.

3.5.3 **Especificação de requisitos do sistema**

[RF001] A administração do aplicativo fica a cargo da secretaria (adição de eventos e usuários) e os docentes podem acessar os dados (sua carga horária, fazer inscrição, consulta de eventos e certificados).

[RF002] Os dados de um usuário docente: nome, e-mail, CPF, matrícula do SIAPE, data de ingresso no programa, status do professor, carga horária e departamento.

[RF003] Os eventos contém data, hora, localização, palestrante e carga horária a ser adicionada. Ao criar um novo evento automaticamente é gerado um QRCode que será disponibilizado no dia do evento para registrar a presença dos participantes.

[RF004] Apenas o usuário administrador pode cadastrar novos usuários no sistema. Ao finalizar o cadastro, a conta de usuário é gerada com uma senha padrão.

[RF005] O usuário docente pode modificar apenas seu e-mail e telefone. Enquanto o administrador pode fazer alteração de todos os dados.

[RF006] Ao interagir com o QR Code disponibilizado pelo evento, automaticamente é registrada a presença do usuário.

[RF007] Ao interagir com o QR Code disponibilizado pelo evento, automaticamente é acrescida a carga horária do evento para o usuário.

[RF008] O sistema deve gerar uma listagem com o nome de todos os usuários que se inscreveram para o evento.

[RF009] O sistema deve gerar uma listagem com o nome de todos os usuários que registraram presença no evento.

[RF010] Os possíveis status são:

1. Concluído: O usuário já atingiu a carga horária necessária para formação;
2. Pendente: O usuário já atingiu o tempo limite para formação (3 anos);
3. Em atividade: Usuário ainda está em processo formativo dentro do tempo limite.

4 **ESTRUTURA DO PROJETO**

Neste capítulo serão levantados os pontos mais relevantes do projeto da API da aplicação, desde a modelagem dos dados, listagem das funcionalidades fornecidas pela API, além da estrutura organizacional do aplicativo PAAP Móvel.

4.1 **API**

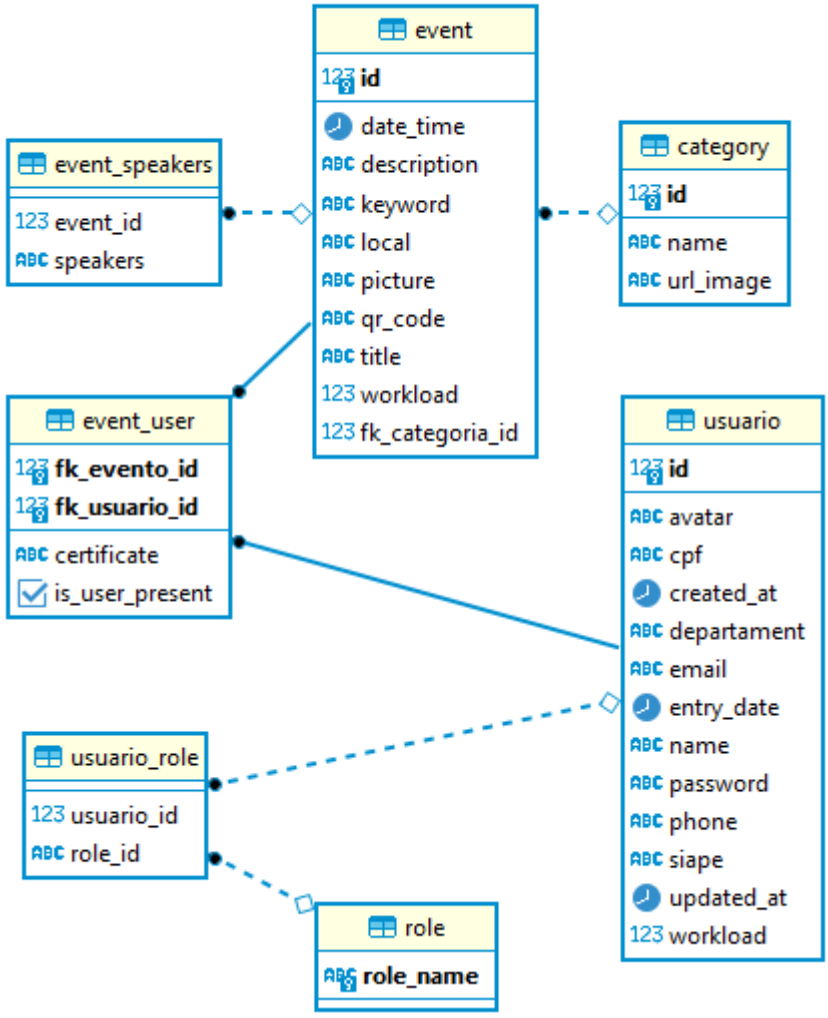
Para a codificação, a linguagem JAVA foi utilizada. Bastante difundida pelas universidades, tem uma grande comunidade ativa desde sua criação. Atualmente é a terceira linguagem mais utilizada no mundo em 2022, de acordo com a pesquisa de popularidade do Tiobe [4]. Permite o uso do paradigma de orientação a objetos e seus benefícios, como encapsulamento, interfaces, herança e polimorfismo. Na API, também é utilizado no JAVA o framework Spring Boot [5], cujo objetivo é prover uma estrutura arquitetural como os repositórios, serviços e controladores, além de abstrair toda a configuração necessária para trabalhar com aplicações JAVA, tornando assim o desenvolvimento mais ágil. O servidor está atualmente hospedado na nuvem através da plataforma Heroku [6], utilizando versão gratuita. E para salvar as imagens é utilizado um repositório na Amazon S3 Service [7] , usufruindo do período gratuito.

4.1.1 **Modelagem dos dados**

No aplicativo, as informações são persistidas em um banco de dados. O SGBD escolhido foi o PostgreSQL [8] . É disponibilizado gratuitamente e com ele é possível modelar um esquema de estruturas que se relacionam entre si. Na figura 4 é mostrado a representação da modelagem de dados que compõe a aplicação.

4 Tiobe Index. Disponível em: https://www.tiobe.com/tiobe-index. Acesso em 03 de Fevereiro de 2022.
5 Spring Boot. Disponível em: https://spring.io/projects/spring-boot. Acesso em 03 de Fevereiro de 2022.
6 Heroku. Disponível em: https://devcenter.heroku.com. Acesso em 04 de Fevereiro de 2022.
7 Amazon S3 Service. Disponível em: https://aws.amazon.com/pt/s3. Acesso em 04 de Fevereiro de 2022
8 PostgreSQL. Disponível em: https://www.postgresql.org . Acesso em 03 de Fevereiro de 2022.

Figura 4 — Modelagem do banco de dados



Fonte: O autor (2022)

A tabela event representa um evento que o usuário queira disponibilizar, e seus respectivos atributos, como data e hora do evento, descrição, localização, foto, titulo, carga horária que será acrescida para o participante, link e palavra chave do QR Code. O evento também possui relacionamento com a tabela de palestrante, onde são guardados os palestrantes referentes a determinado evento. Um evento também se relaciona com uma categoria, na qual ele deve pertencer. O usuário tem os seus devidos atributos, como data de entrada, CPF, matrícula do SIAPE, entre outros. Todo usuário do sistema deve ter uma função, representada pela tabela role. Na aplicação existem duas funções específicas que são as de usuário comum e de administrador. A tabela de relação event_user representa o relacionamento entre o evento e o usuário. Quando um dado entre evento e usuário é inserido nessa tabela, significa que o usuário se inscreveu para o evento, caso contrário, o dado seja removido da tabela, o usuário removeu sua inscrição no evento. Os atributos de presença e link do certificado são modificados à medida que o usuário interage, registrando a frequência

e solicitando download do certificado.

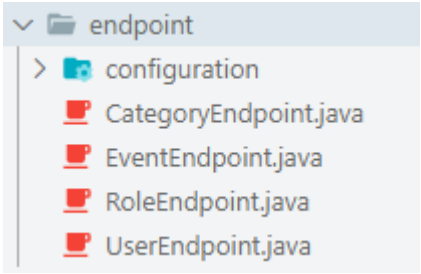
4.1.2 **Estrutura organizacional da API**

Neste tópico é apresentado a arquitetura organizacional dos arquivos que compõem a API da aplicação:

4.1.2.1 **Endpoints**

Essa parte do projeto contém todas as classes que fazem a ponte entre o cliente (aplicativo) e a regra de negócio. São os pontos de entrada para as requisições e onde saem os dados para o consumo. No framework Spring Boot são chamados de controladores.

Figura 5 — Endpoints

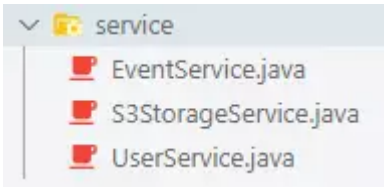


Fonte: O autor (2022)

4.1.2.2 **Services**

São as classes que lidam com as regras de negócio, nestes arquivos estarão concentrados a maior parte da lógica da aplicação.

Figura 6 — Services

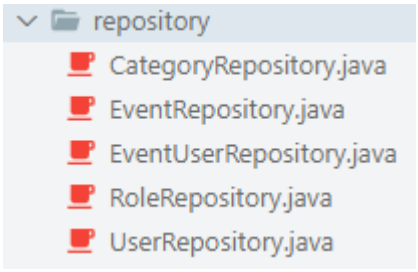


Fonte: O autor (2022)

4.1.2.3 **Repositories**

São as classes responsáveis pelo acesso ao banco de dados, persistência e busca de informações referente a cada entidade.

Figura 7 — Repositories

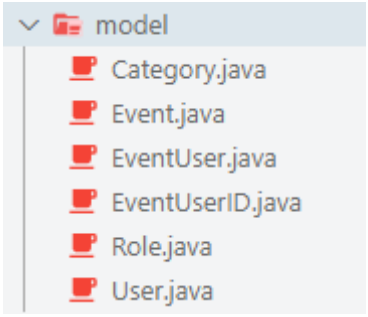


Fonte: O autor (2022)

4.1.2.4 **Models**

São as classes que representam as entidades da aplicação.

Figura 8 — Models

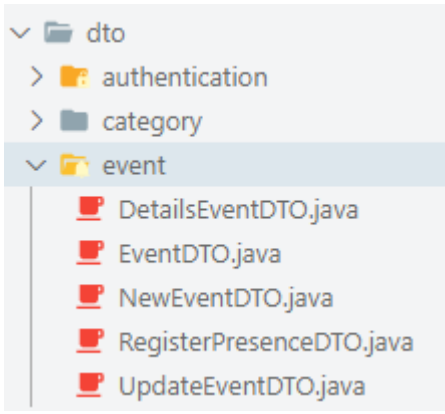


Fonte: O autor (2022)

4.1.2.5 **DTOs**

São as classes utilizadas para transferência de dados entre cliente e servidor. É um padrão de projeto bastante usado em Java para o transporte de dados entre diferentes componentes do sistema, tem como função agrupar um conjunto de atributos em uma classe simples de forma a otimizar a comunicação.

Figura 9 — DTOs

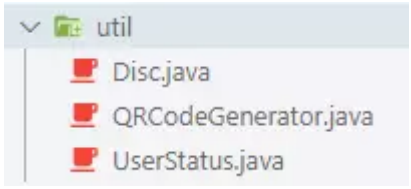


Fonte: O autor (2022)

4.1.2.6 **Utils**

São classes que possuem funções auxiliares.

Figura 10 — Utils

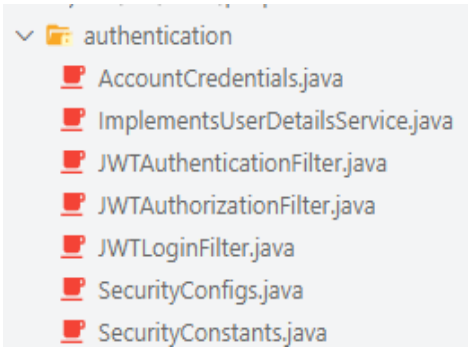


Fonte: O autor (2022)

4.1.2.7 **Authentication**

São classes responsáveis pelo processo de login e segurança da aplicação.

Figura 11 — Authentication



Fonte: O autor (2022)

4.1.3 **Métodos da API**

Nesta sessão serão exibidas as funcionalidades mais relevantes expostas para acesso do cliente realizadas através dos protocolos HTTPS.

Tabela 2 — Métodos de Usuário

Verbo HTTP	Endereço	Descrição
POST	api/user	Insere um novo usuário
GET	api/user	Busca todos os usuários
GET	api/user/10	Busca usuário com id 10
PUT	api/user/10	Atualiza usuário com id 10
PUT	api/user/10/changePassword	Atualiza senha do usuário com id 10
DELETE	api/user/10	Remove usuário com id 10

Fonte: O autor (2022)

Tabela 3 — Métodos de Evento

Verbo HTTP	Endereço	Descrição
GET	api/event	Busca todos os eventos
GET	api/event/5	Recupera o evento com id 5
GET	api/event/open	Recupera todos os eventos abertos
GET	api/event/enrolled?userID=5	Recupera todos os eventos que o usuário com id 5 esteja inscrito
GET	api/event/historic?userID=5	Recupera todos os eventos que o usuário com id 5 esteve presente
POST	api/event	Insere um novo evento
POST	api/event/5/subscribe?userID=10	Inscreve no evento com id 5 o usuário com id 10
POST	api/event/certification? eventID=5&userID=10	Gera certificado do evento com id 5 para o usuário com id 10
PUT	api/event	Atualiza dados do evento
PUT	api/event/5/register-presence	Atualiza presença do usuário no evento com id 5
DELETE	api/event/5	Deleta o evento com id 5
DELETE	api/event/5/remove-subscribe? userID=10	Remove inscrição no evento com id 5 do usuário com id 10

Fonte: O autor (2022)

4.1.4 **Codificação dos Métodos**

Nesta seção são explicadas algumas regras de negócio essenciais para o funcionamento da aplicação.

4.1.4.1 **Criação de evento**

Ao enviar os dados para o criação de eventos automaticamente é gerado uma palavra passe que representará o evento em específico. O método de salvar é responsável por criar uma palavra, que é composta de um texto aleatório e o id do evento, e logo em sequência executa o método GenerateQRCode, guardando a palavra passe dentro da imagem, retornando o endereço URL de onde será salvo.

Código 1 — Método GenerateQrCode

```
    private String generateQRCode(Event newEvent) {
        String resourceURL = null;
        try {
            var file = File.createTempFile(String.format("qrcode_%d",
newEvent.getId()), ".png");
            file = qrCodeGenerator.create(newEvent.getKeyword(), 200, 200,
file);
            resourceURL = s3StorageService.saveImage(file, newEvent.getId(),
QRCODES_FOLDER);
        } catch (Exception e) {
            throw new RuntimeException("Erro ao gerar QRCode: " +
e.getMessage());
        }
        return resourceURL;
    }
```

Fonte: O autor (2022)

4.1.4.2 **Registro e remoção de inscrição**

O método de inscrição recebe o identificador do usuário e do evento, antes de tudo é verificado se ambos ids realmente existem na base dados através do findEvent e findUser. Se passar por essas verificações, é checado se já existe essa relação na tabela event_user. Se for verídico, é lançada uma exceção com a mensagem que o usuário já está inscrito. Se não houver esta relação, significa que o usuário não está inscrito realizando de fato a inscrição e retornando o status ok para o cliente.

Código 2 — Método de inscrição

```
/**
     * Inscreve um usuário em um evento.
     *
     * @param eventoID Id do Evento ao qual o usuário deseja participar.
     * @param userID   Id do usuário interessado.
     */
    public ResponseEntity<?> subscribe(Long eventoID, Long userID) {
        var event = findEvent(eventoID);
        var user = findUser(userID);
        var findRelation = findRelation(event.getId(), user.getId());
        if (findRelation.isPresent())
            throw new RuntimeException(String.format("Usuário(a) %s já inscrito
no evento", user.getName()));
        var relation = EventUser.builder()
                .event(event)
                .user(user)
                .build();
        eventUserRepository.save(relation);
        return ResponseEntity.ok().build();
    }
```

Fonte: O autor (2022)

Na remoção, também é checado se o evento e usuário existem, caso positivo é feito a busca pela relação entre os dois na tabela event_user. Se verdadeiro, a conexão entre os dois é deletada da base de dados, retornando ok para o cliente, caso negativo é lançada uma exceção ao qual a mensagem informa que o usuário não possui inscrição no evento.

Código 3 — Método de remoção de inscrição

```
public ResponseEntity<?> removeSubscribe(Long eventoID, Long userID) {
        var event = findEvent(eventoID);
        var user = findUser(userID);
        var relationID = new EventUserID(event.getId(), user.getId());
        return eventUserRepository.findById(relationID)
                .map(relation -> {
                    eventUserRepository.deleteById(relationID);
                    return ResponseEntity.ok().build();
                }).orElseThrow(() -> new RuntimeException("Usuário não possui
inscrição no evento"));
    }
```

Fonte: O autor (2022)

4.1.4.3 **Registro de frequência e carga horária**

A classe DTO de registro de presença possui o id do usuário e a palavra chave que ele escaneou no QRCode. Primeiramente é checado se o usuário está inscrito no evento, caso negativo é lançada uma exceção, se o código passar dessa validação, então existem as seguintes validações:

- Checa se a palavra “passe” escaneada é a mesma salva na entidade do evento, caso negativo, retorna um erro para o usuário com a mensagem de código inválido. Caso positivo, avança para a próxima validação.
- Checa se o evento já está acontecendo. Se negativo, retorna um erro para o usuário com a mensagem que o evento ainda não iniciou. Caso positivo, avança para a próxima validação.
- Checa se o usuário já registrou a presença anteriormente. Caso positivo, é retornada uma mensagem de erro informando que a presença já foi registrada anteriormente. Caso negativo, a presença é registrada e a carga horária adicionada para o usuário, retornando ok para o cliente.

Código 4 — Método de registro de frequência e carga horária

```
public ResponseEntity<?> registerPresence(Long eventID, RegisterPresenceDTO
data) {

        return findRelation(eventID, data.getUserID())
                .map(relation -> {
                    this.validateEventKeyCode(relation, data.getKeyword());
                    this.checkIfEventIsHappening(eventID);
                    if (relation.isUserPresent())
                        throw new IllegalArgumentException("Sua presença já foi
registrada anteriormente");
                    relation.setUserPresent(true);
                    var eventWorkload = relation.getEvent().getWorkload();
                    var user = relation.getUser();
                    user.setWorkload(user.getWorkload() + eventWorkload);
                    return ResponseEntity.ok().body(UserDTO.parse(user));

                }).orElseThrow(() -> new RuntimeException("Relação entre evento
e usuário não existe"));
    }
```

Fonte: O autor (2022)

4.1.4.4 **Download de certificado**

Este método também possui a verificação entre a relação evento e usuário, validado esta parte, o código irá verificar se o usuário esteve presente no evento. Se o usuário não participou do evento é lançada uma exceção com a mensagem de que o usuário não registrou presença no evento. Caso o atributo, presença do usuário, esteja como verdadeiro, o código passa para próxima validação.

Neste momento é checado se o atributo certificado está setado na entidade event_user. Caso seja nulo, significa que o usuário está solicitando pela primeira vez o download do certificado, então o método de geração de certificado é chamado. Para codificar a função de produção de certificados, foi utilizado a ferramenta JasperSoft e JasperSoft Studio[9] disponível para a linguagem Java. Com esse programa, geramos um template visual em JRXML que possui campos que serão preenchidos em tempo de execução. O template recebe o nome, data, e carga horária do evento e nome do participante, gerando assim de forma automática a certificação. No final, o método retorna a url onde estará salvo o certificado. Se o atributo certificado não for nulo, significa que o usuário já gerou o certificado anteriormente, então é só retornar o link que está no atributo. Ao final do método, é gerado um pdf com o nome do usuário e do evento e enviado para o cliente.

9 Jaspersoft Studio. Disponível em https://community.jaspersoft.com/project/jaspersoft-studio. Acesso em 03 de Fevereiro de 2022.

Código 5 — Método de Download de certificado

```
public ResponseEntity<?> getCertification(Long eventID, Long userID) {
        return this.findRelation(eventID, userID)
                .map((relation) -> {
                    var participated = relation.isUserPresent();
                    if (!participated) {
                        throw new RuntimeException(String.format("%s não
possui registro de presença no evento %s",
                                relation.getUser().getName(),
relation.getEvent().getTitle()));
                    }
                    if (relation.getCertificate() == null) {

relation.setCertificate(this.generateCertificate(relation));
                    }
                    return ResponseEntity.ok()
                            .contentType(MediaType.APPLICATION_PDF) //
"application/octet-stream"
                            .header(HttpHeaders.CONTENT_DISPOSITION,
                                    String.format("attachment;
filename=\"%s_%s.pdf\"", relation.getEvent().getTitle(),
                                            relation.getUser().getName()))

.body(getDocumentStream(relation.getCertificate()));
                })
                .orElseThrow(() -> new RuntimeException("Cadastro em evento
inexistente"));
    }
```

Fonte: O autor (2022)

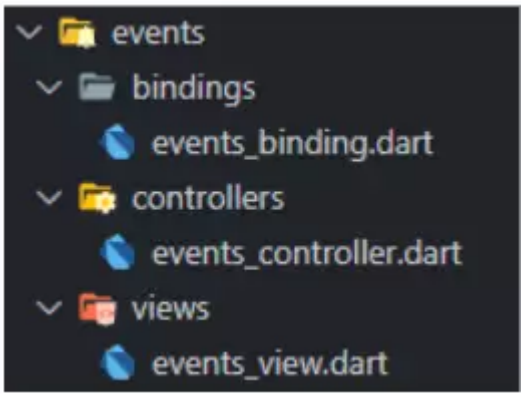
4.2 **Aplicativo**

Foi utilizado para o desenvolvimento da aplicação o framework híbrido Flutter. Tal escolha é decorrente do fato que o framework utilizado permite que a aplicação produzida não esteja limitada a um sistema operacional, além de que as tecnologias usadas no desenvolvimento, são similares das tecnologias web, o que torna possível reutilizar o conhecimento que o desenvolvedor já tem no âmbito da web para a produção do aplicativo. Dessa forma, é possível atender, além dos usuários Android, os usuários de IOS de forma mais rápida, sem ter que codificar novamente toda a aplicação.

4.2.1 **Estrutura Organizacional do Aplicativo**

O projeto do aplicativo móvel segue o modelo MVC onde possibilita separar a lógica de funcionamento, da interface e dos dados para uma melhor organização do código. As telas são dividas em módulos e cada módulo possui sua classe responsável pela interface, manipulação dos dados e a integração de ambos. A utilização dessa abordagem auxilia o desenvolvimento do código facilitando a manutenção e a modificação do mesmo. Veja abaixo o esquema de organização de alguns módulos do aplicativo:

Figura 12 — Módulo de Evento



Fonte: O autor (2022)

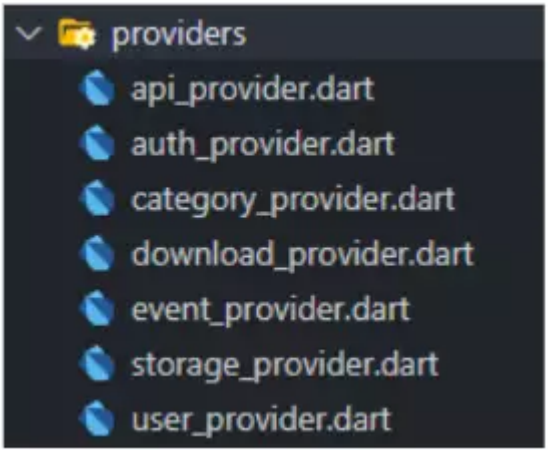
As classes bindings são responsáveis por carregar todos os dados necessários pro módulo em específico funcionar, e os controllers manipulam os dados para serem exibidos na view. Esse padrão arquitetural é disponibilizado pelo framework do Flutter chamado GetX [10].

4.2.1.1 **Providers**

Além dos módulos existem as classes providers, responsáveis pela chamada aos métodos da API, provendo os dados para o aplicativo.

10 GetX. Disponível em: https://pub.dev/packages/get. Acesso em 03 de Fevereiro de 2022.

Figura 13 — Providers

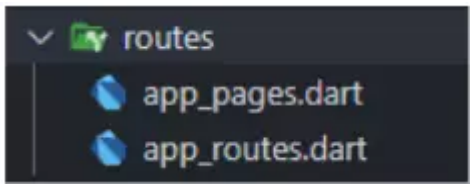


Fonte: O autor (2022)

4.2.1.2 **Routes**

Classes que executam o gerenciamento das páginas;

Figura 14 — Routes

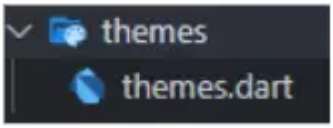


Fonte: O autor (2022)

4.2.1.3 **Themes**

Classe responsável pelas configurações dos esquemas de cores Claro e Escuro da aplicação.

Figura 15 — Themes



Fonte: O autor (2022)

5 DESENVOLVIMENTO

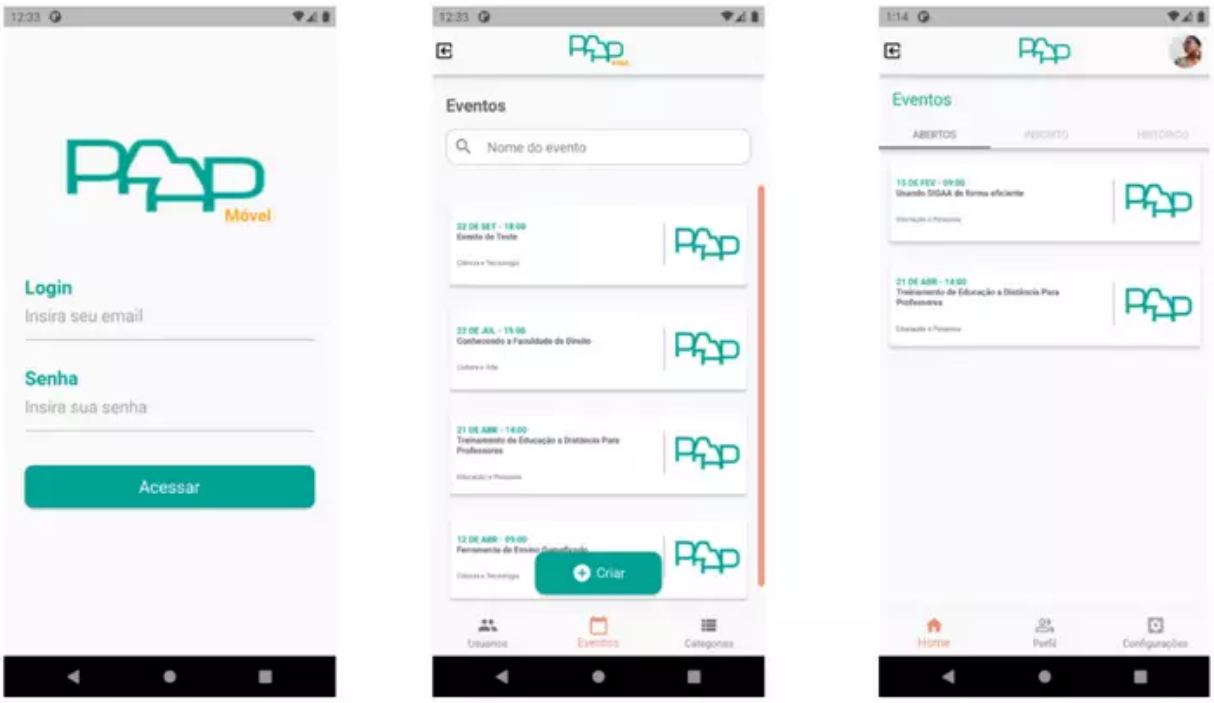
5.1 Aspectos Funcionais

O aplicativo possui as funcionalidades de login de usuários, cadastro e atualização de informações (eventos, usuários, categorias), disponibilização dos dados para consulta, processo de registro de presença e download de certificados.

5.1.1 Login e telas iniciais

Na Figura 16, são exibidas as telas de login e respectivos módulos de administrador e docente. Nelas, é possível entrar na aplicação utilizando os dados de e-mail e senha do usuário. Dependendo do tipo de função que o usuário possui, ele poderá ser redirecionado para a tela de home de administrador ou de docente. Não há página ou link para criação de conta, pois esse aplicativo é restrito para a administração do programa, sendo assim, somente administradores podem realizar criação de usuário.

Figura 16 — Tela de login e respectivos módulos de administrador e docente

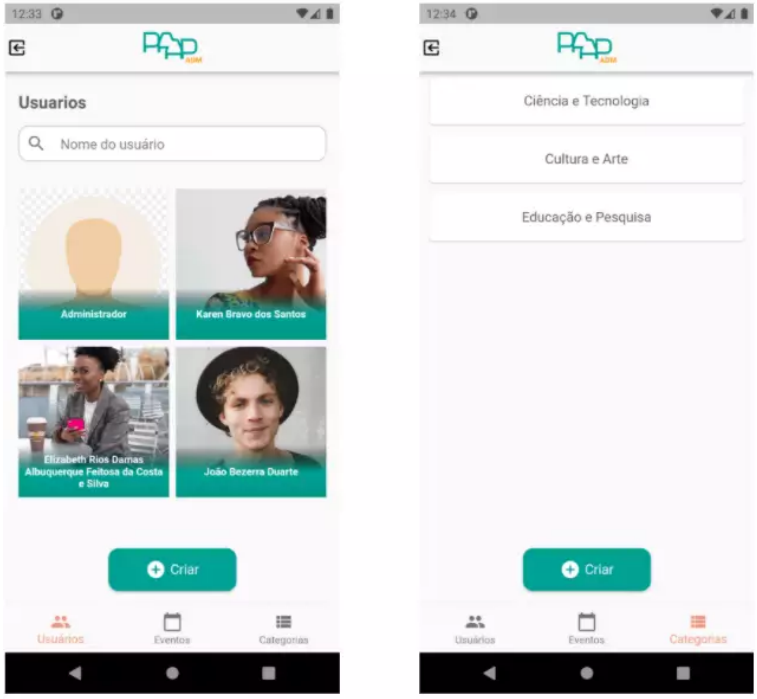


Fonte: O autor (2022)

5.1.2 **Módulo Administrador**

Na Figura 17, são apresentadas as telas de listagem de usuários e categorias. Nesta tela, após acessar o aplicativo com uma conta de administrador, o usuário é levado a uma tela de listagem de todos eventos (a tela inicial do administrador), onde é possível ver os detalhes clicando no cartão do respectivo evento, também é possível pesquisar eventos através do botão de busca. Há também uma barra de navegação com abas onde é possível navegar para as telas de visualização de usuários ou de categorias. Todas as telas possuem o mesmo padrão, uma listagem dos itens referidos e um botão para criação.

Figura 17 — Telas de usuário e categoria



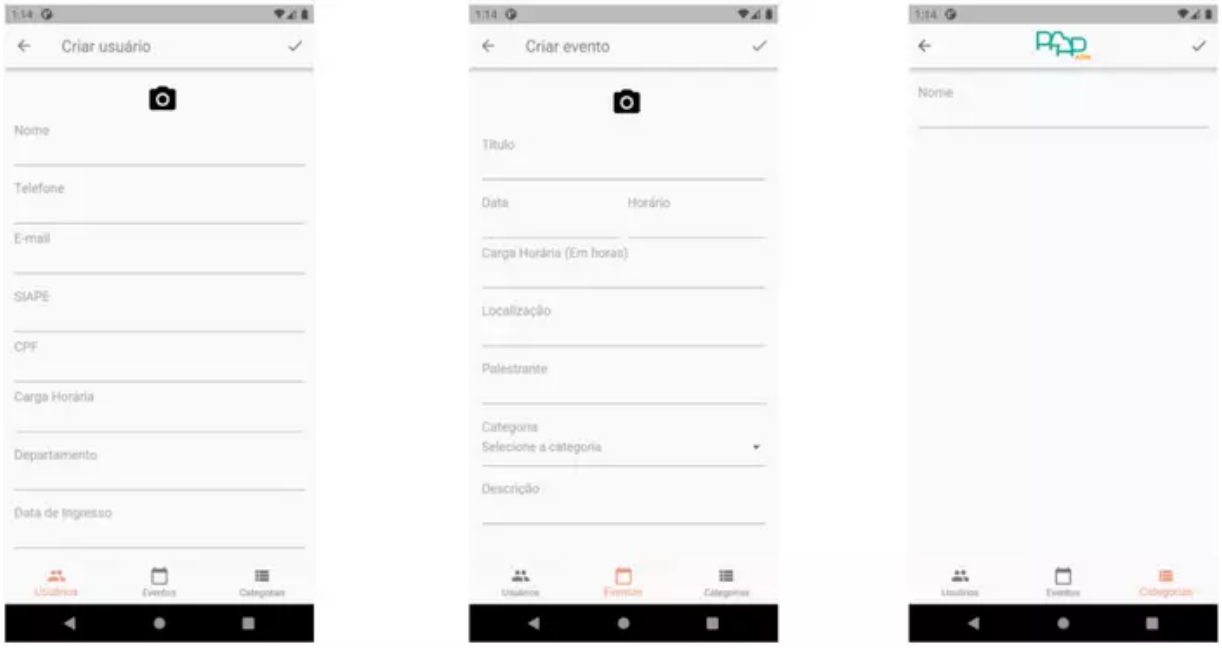
Fonte: O autor (2022)

5.1.3 **Telas de Cadastro**

No módulo de administrador é possível realizar a criação de um usuário, evento ou categoria clicando no respectivo botão da tela. O aplicativo redireciona para uma página que solicita as informações a serem salvas no banco de dados. Para criar um usuário, a regra de negócio verifica se o e-mail do usuário já está em uso (necessário para fazer login), caso já esteja sendo utilizado, um alerta é exibido ao usuário. Todos os usuários por padrão são gerados com a senha “paap”, e se desejarem trocar a senha, podem acessar a aba de

configurações, como mostra a Figura 22b. Ao criar um evento, a regra de negócio gera um QR Code para ser disponibilizado no dia do evento, somente o administrador pode visualizá-lo. As telas de cadastro, conforme apresentadas na Figura 18, também são usadas para edição. As telas aceitam imagens, mas não é uma funcionalidade obrigatória, caso o usuário não insira, o evento e usuário serão cadastrados com imagens padrões do aplicativo. O plugin utilizado para selecionar imagens foi o image_picker[11].

Figura 18 — Telas de cadastro



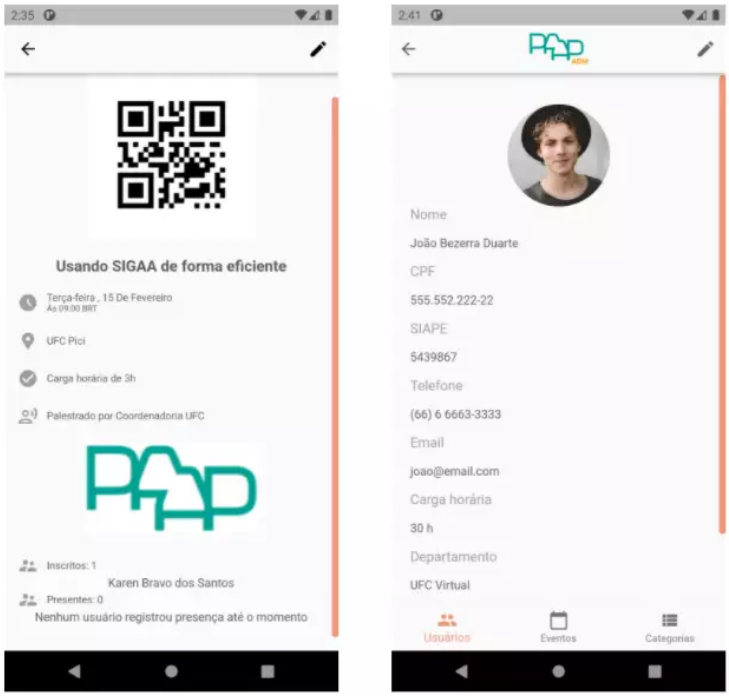
Fonte: O autor (2022)

5.1.4 **Telas de mais detalhes**

Na Figura 19, são apresentadas as telas de detalhes. Nelas, é possível encontrar as informações detalhadas de eventos e usuários cadastrados. Para acessá-las, deve-se clicar no cartão do evento ou usuário desejado. Em “detalhes do evento” (Figura 19a), o administrador tem acesso ao QR Code, consulta dos inscritos no evento e também os usuários que registraram presença, além das informações comuns. Já em “detalhes do usuário” (Figura 19b), é possível acompanhar a carga horária atual do docente e suas informações. Ambas as telas possuem um botão para edição localizado no canto superior direito.

11 image_picker. Disponível em: https://pub.dev/packages/image_picker. Acesso em 04 de Fevereiro de 2022.

Figura 19 — Telas de mais detalhes



Fonte: O autor (2022)

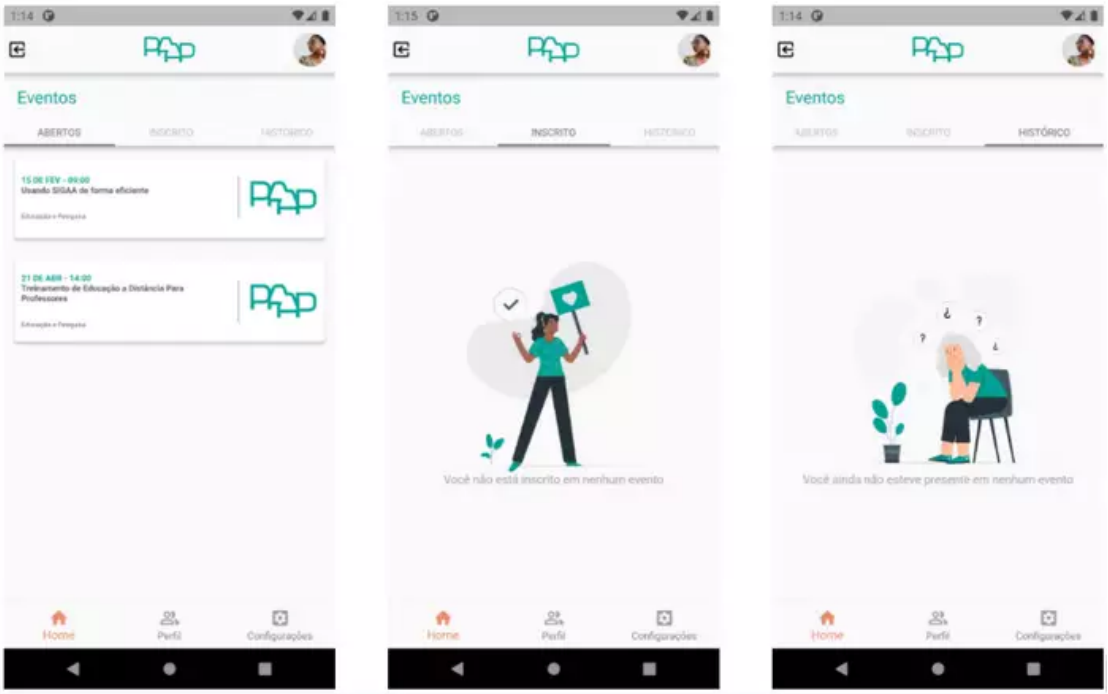
5.1.5 **Módulo Usuário**

Ao acessar este módulo com uma conta de usuário docente, o utilizador é redirecionado para home, onde encontra-se uma barra de navegação inferior onde é possível navegar para perfil ou configurações. Na home o usuário encontra tudo a respeito dos eventos, listados a seguir:

- Eventos abertos (Figura 20a): São os eventos que ainda irão acontecer. De acordo com a regra de negócios, um evento aberto é aquele cujo a data de acontecimento é inferior à data atual.
- Eventos inscritos (Figura 20b): São os eventos que o usuário manifestou interesse em participar.
- Histórico de eventos (Figura 20c): São os eventos que o usuário já registrou sua presença.

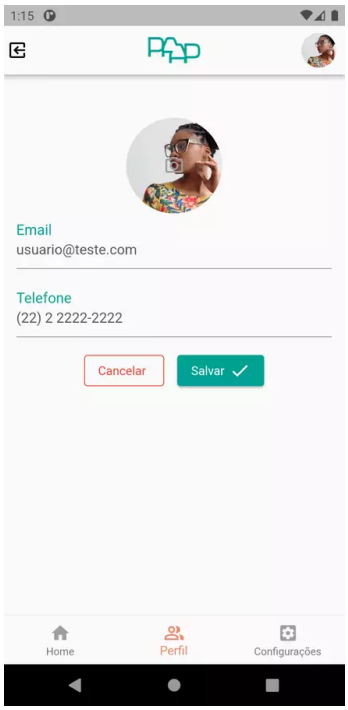
Na tela de perfil (Figura 21), são exibidos os dados básicos do usuário com um botão para editar caso o usuário prefira. Na aba de configurações (Figura 22), o usuário pode escolher o tema do aplicativo, alterar sua senha e acessar informações sobre o aplicativo.

Figura 20 — Tela de eventos para docente



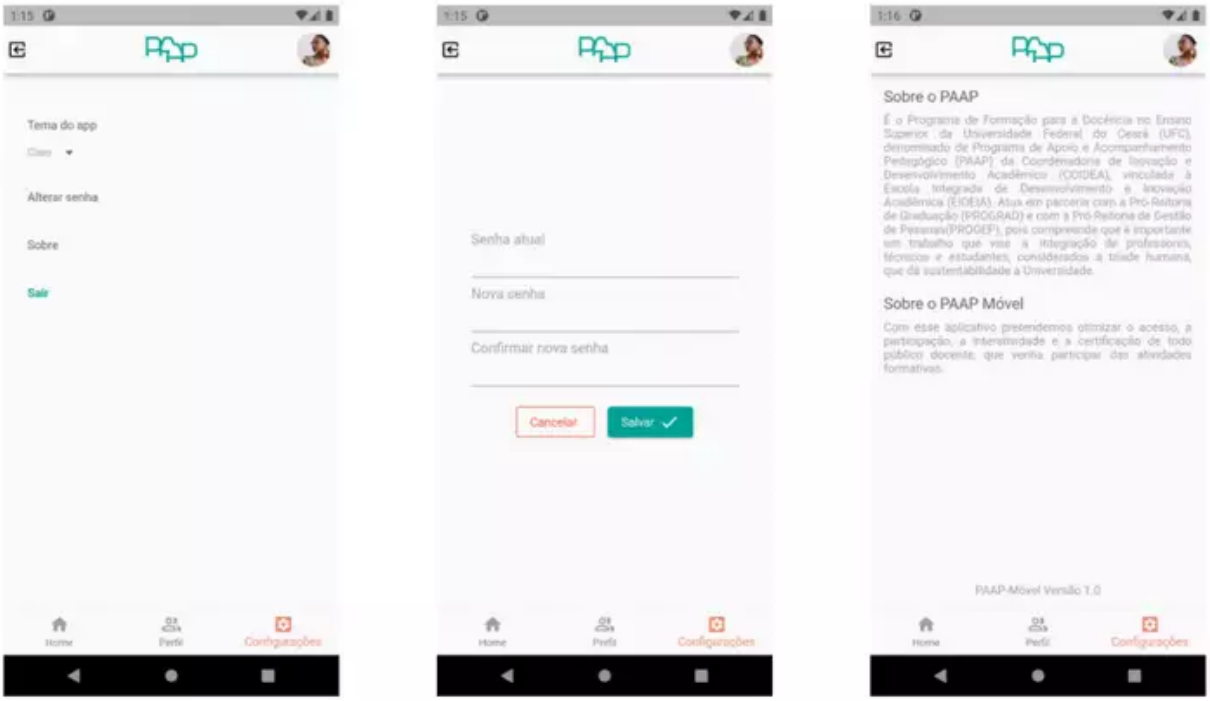
Fonte: O autor (2022)

Figura 21 — Perfil de usuário



Fonte: O autor (2022)

Figura 22 — Tela de configurações



Fonte: O autor (2022)

5.1.6 **Inscrição e presença no evento**

Ao clicar no cartão do evento desejado, o usuário docente é redirecionado para a tela de detalhes. Esta é uma página complexa que abrange 3 estados, como mostra a Figura 23:

- Usuário não inscrito: Primeira interação do usuário com a tela ou usuário interagiu com o botão de remover sua inscrição.
- Usuário inscrito: O usuário já interagiu com essa página clicando no botão de participar do evento.
- Usuário presente: O usuário já registrou a presença neste evento.

Figura 23 — Diagrama de Estados Tela de Inscrição

Fonte: O autor (2022)

Inicialmente, o usuário está no estado de "não inscrito" (Figura 24) ao acessar essa tela. Ao clicar em participar do evento, o usuário modifica o estado para "inscrito" (Figura 25). Neste estado (inscrito), são exibidos dois botões, o de registrar a presença através de QR Code, e o de remoção da inscrição no evento. Para a funcionalidade de escaneamento foi utilizado o plugin flutter_barcode_scanner [12].

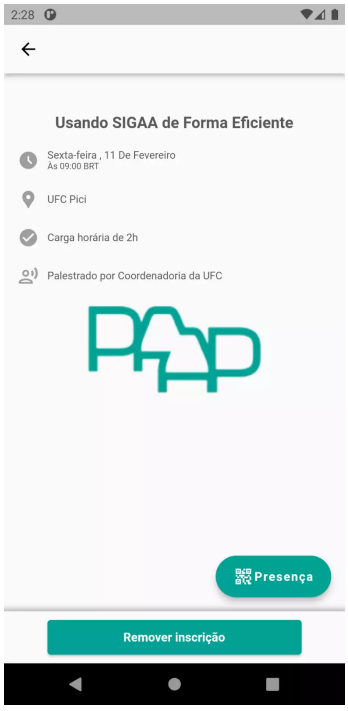
Para evitar erros na aplicação, diversas regras de negócio foram aplicadas. A primeira, só é possível registrar presença quando o evento já estiver iniciado, evitando que usuários registrem presença antes do mesmo acontecer. A segunda regra diz que se o usuário tentar escanear qualquer outro QRCode que não seja o do evento em específico, é exibido para este que o código é inválido. Após escanear e tudo correr de forma bem sucedida, a presença é automaticamente registrada no evento. Então o usuário vai para o último estado da tela, que é o de presente (Figura 26). Nesse momento o botão disponível para o usuário é o de download de certificado.

12 flutter_barcode_scanner. Disponível em https://pub.dev/packages/flutter_barcode_scanner. Acesso em 04 de Fevereiro de 2022.

Figura 24 — Estado não inscrito

Fonte: O autor (2022)

Figura 25 — Estado inscrito



Fonte: O autor (2022)

Figura 26 — Estado presente

Fonte: O autor (2022)

6 **CONCLUSÃO**

Este trabalho apresentou o desenvolvimento de uma aplicação móvel que tem como objetivo otimizar as atividades burocráticas e organizacionais realizadas pela administração do programa PAAP da UFC (perfil administrador) e além disso, facilitar o acesso ao registro e consulta referente às ações promovidas pelo PAAP aos professores docentes recém ingressos na universidade (perfil usuário). Foi definido um conjunto de requisitos que, quando desenvolvidos, suprem grande parte das necessidades administrativas que o programa possui. A implementação foi mostrada no decorrer do projeto e o código fonte da API , do aplicativo e os testes unitários e de integração estão disponibilizados no Github[13]. Atualmente o servidor se encontra disponível na nuvem em fase de testes. Entre as dificuldades enfrentadas no decorrer do desenvolvimento estão a dificuldade em gerir o projeto e ao mesmo tempo desenvolver, não tendo uma divisão e definição exata das atividades.

Como trabalhos futuros, intenta-se tornar a aplicação mais reativa, implementando notificações para que os usuários sejam alertados quando surgirem novos eventos, sem a necessidade de acesso constante ao aplicativo para verificar se há novos eventos. Além disso, objetiva-se notificar administradores quando usuários concluírem sua jornada formativa; Integrar a aplicação para enviar e-mails para os usuários; Desenvolver uma tela de relatório geral, na qual mostrará gráficos de presença e frequência de forma mais detalhada. Implementar funcionalidade que valide o certificado emitido pela aplicação; Abranger novos perfis, como por exemplo, estudantes, ou público comunitário que participa dos eventos.

13 Repositório da aplicação. Disponível em https://github.com/iagoGB/PAAP-APP Acesso em 02 de Fevereiro de 2022. Repositório da API. Disponível em https://github.com/iagoGB/PAAP-BackEnd Acesso em 02 de Fevereiro de 2022.

REFERÊNCIAS

AZEVEDO, Rafael. **Seleção de padrões para a arquitetura de software: Uma abordagem em procura de termos e sinônimos**. Disponível em: https://www.locus.ufv.br/bitstream/handle/123456789/2686/texto%20completo.pdf. Acesso em: 7 set. 2020.

BRUNO, Brito. **Padrão Rest**. Disponível em: https://www.brunobrito.net.br/padrao-rest/. Acesso em: 6 set. 2020.

CASa. **Comunidade de cooperação e aprendizagem significativa**. Disponível em: https://casa.ufc.br/wp-content/uploads/2019/05/carta-casa-atualizada-2019-1.pdf. Acesso em: 14 ago. 2020.

COUTO, Lívia . **Arquitetura Rest**. Disponível em: http://www.monografias.ice.ufjf.br/tcc-web/exibePdf?id=17. Acesso em: 6 set. 2020.

Figma. **Figma**. Disponível em: https://www.figma.com/ux-design-tool/. Acesso em: 6 set. 2020.

G1. **Em 10 anos no Brasil, Android foi de 2 smartphones para sistema operacional dominante do mercado**. Disponível em: https://g1.globo.com/economia/tecnologia/noticia/2019/11/26/ha-10-anos-no-brasil-android-foi-de-2-smartphones-para-sistema-operacional-dominante-do-mercado.ghtml. Acesso em: 9 jul. 2020.

HECK, Fernando. **Sistema Móvel de Controle de Presença**. Disponível em: https://www.lume.ufrgs.br/bitstream/handle/10183/100288/000931702.pdf. Acesso em: 8 out. 2020.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA(IBGE). **Uso de internet, televisão e celular no Brasil**. Disponível em: https://educa.ibge.gov.br/jovens/materias-especiais/20787-uso-de-internet-televisao-e-celular-no-brasil.html#subtitulo-1. Acesso em: 5 ago. 2020.

JÚNIOR, Neri; MERCADO, Neyza. **Vantagens e desvantagens da utilização do Ionic framework para o desenvolvimento de aplicativos móveis**. Disponível em: https://semanaacademica.org.br/system/files/artigos/artigo_74.pdf. Acesso em: 6 set. 2020.

LIMA, Thiago. **Performance em APIs RESTful: Cache, compressão e mais!**. Disponível em: https://www.linkapi.solutions/blog/tecnico/performance-em-apis-restful/. Acesso em: 6 set. 2020.

Matera. **Boas Práticas para Desenvolvimento de APIs REST**. Disponível em: http://www.matera.com/blog/post/boas-praticas-para-desenvolvimento-de-apis-rest. Acesso em: 6 set. 2020.

MATOS, Beatriz; SILVA, João. **Estudo comparativo entre o desenvolvimento de aplicativos móveis utilizando plataformas nativas e multiplataforma**. Disponível em: http://fga.unb.br/articles/0001/5113/Beatriz_Joao_TCC_Aplicativos_M_veis.pdf. Acesso em:

9 set. 2020.

MULLER, Gabriel; SOARES, Inali. **Estudo Comparativo Sobre Ferramentas de Desenvolvimento Multiplataforma para Aplicações Móveis.**. Disponível em: https://www2.unicentro.br/decomp/files/2019/03/TCC-Gabriel-M%E3%80%95ler.pdf?. Acesso em: 7 set. 2020.

PAREDES, Arthur. **20 ferramentas de prototipagem, UX e usabilidade na web**. Disponível em: https://www.iebschool.com/pt-br/blog/analitica-web/usabilidade-e-ux/20-ferramentas-de-prototipagem-e-usabilidade-na-web/. Acesso em: 9 set. 2020.

SAMPAIO, Cleuton. **Desenvolvimento de APIs Rest.**. Disponível em: http://www.softwarelivre.gov.br/palestras-tecnicas-cisl/slide_rest. Acesso em: 6 set. 2020.

SERVIÇO BRASILEIRO DE APOIO A MICROS E PEQUENAS EMPRESAS(SEBRAE). **Tendências para o desenvolvimento de Apps para 2021**. Disponível em: https://comunidadesebrae.com.br/tecnologia/tendencias-para-o-desenvolvimento-de-apps-para. Acesso em: 25 ago. 2020.

SHAW, Mery; GARLAN, David. **An Introduction to Software Architecture**. Disponível em: https://www.cs.cmu.edu/afs/cs/project/able/ftp/intro_softarch/intro_softarch.pdf. Acesso em: 9 set. 2020.

SisEventos. **Organizar um evento nunca foi tão simples**. Disponível em: http://cev.urca.br/siseventos/. Acesso em: 8 out. 2020.

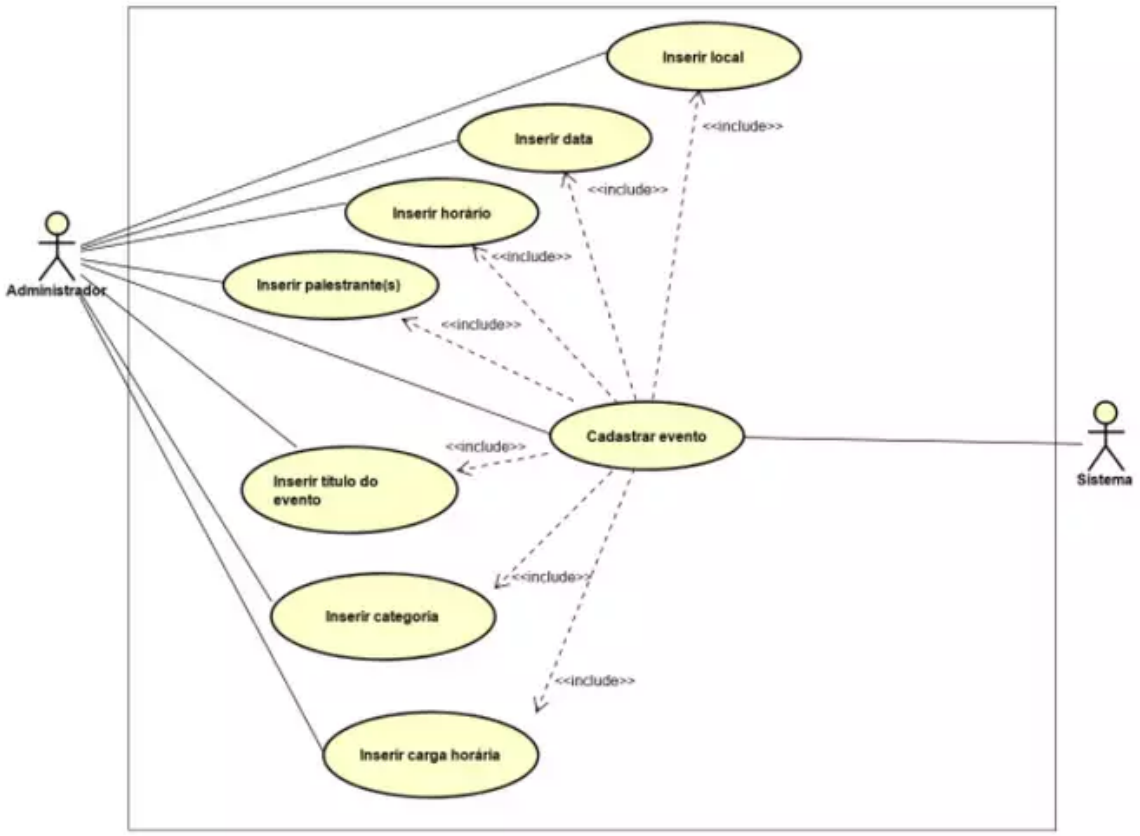
SOMMERVILLE, IAN. **ENGENHARIA DE SOFTWARE**. 9 ed. São Paulo: Pearson Prentice Hall, 2011.

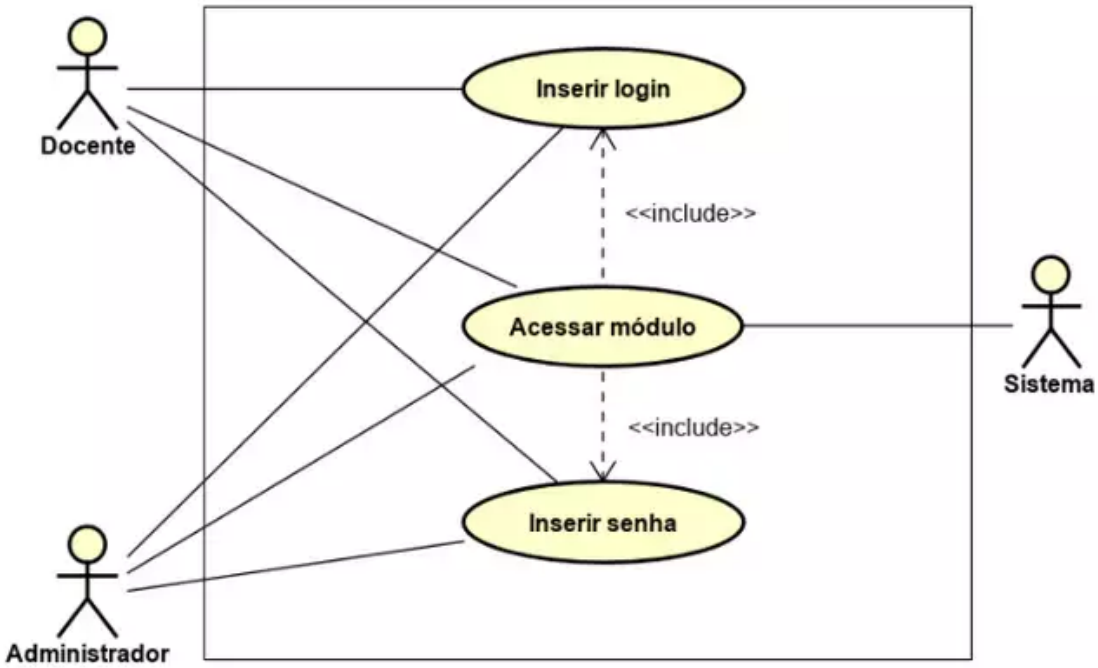
SOUZA, Ivan. **Entenda o que é Rest API e a importância dele para o site da sua empresa**. Disponível em: https://rockcontent.com/br/blog/rest-api/. Acesso em: 6 set. 2020.

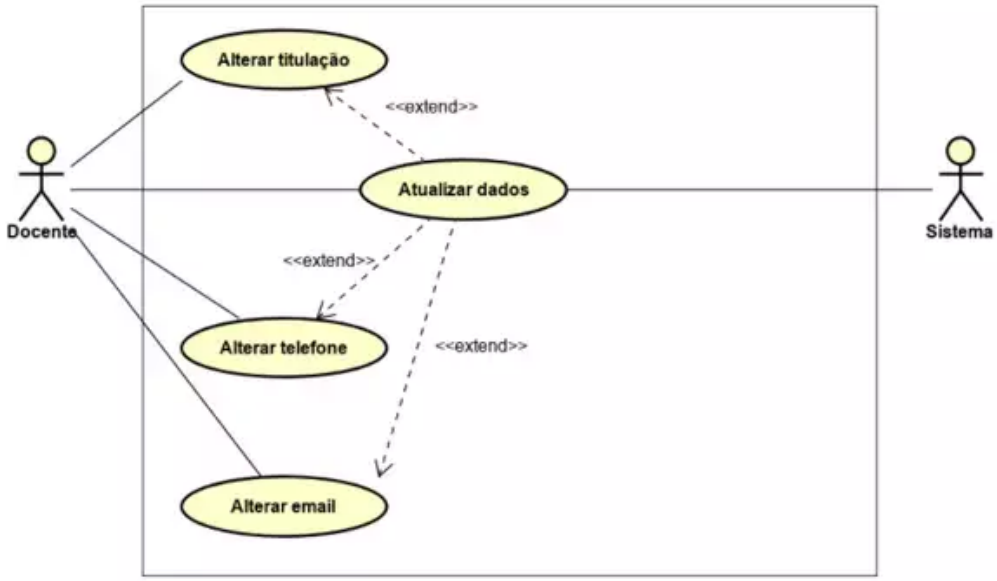
Tic Domicílios. **Pesquisa sobre o uso de tecnologias de informação e comunicação no domicílios brasileiros**. Disponível em: https://cetic.br/media/docs/publicacoes/2/12225320191028-tic_dom_2018_livro_eletronico.pdf. Acesso em: 5 ago. 2020.

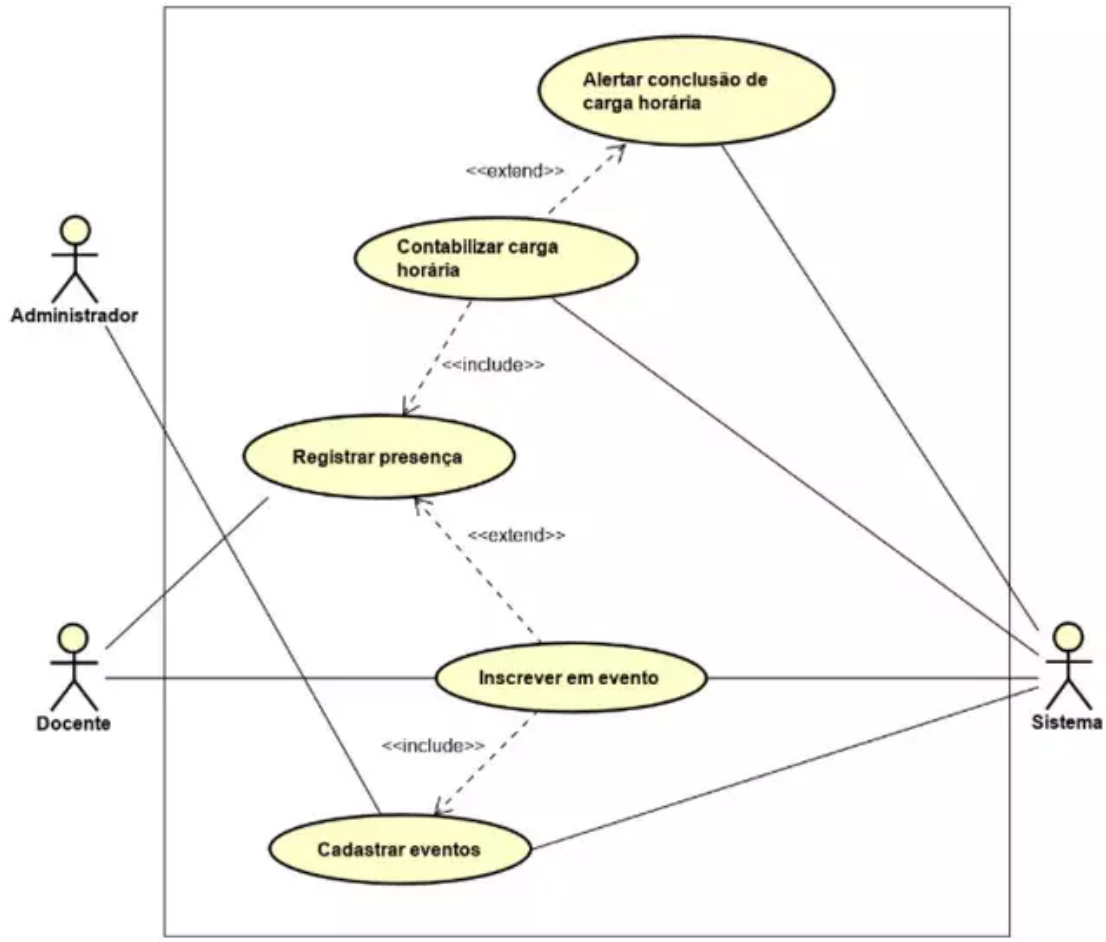
TOBALDINI, Ricardo. **Aplicação do Estilo Arquitetural REST a um Sistema de Congressos**. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/183993/TCCRicardoGhisiTobaldini_final.pdf. Acesso em: 6 set. 2020.

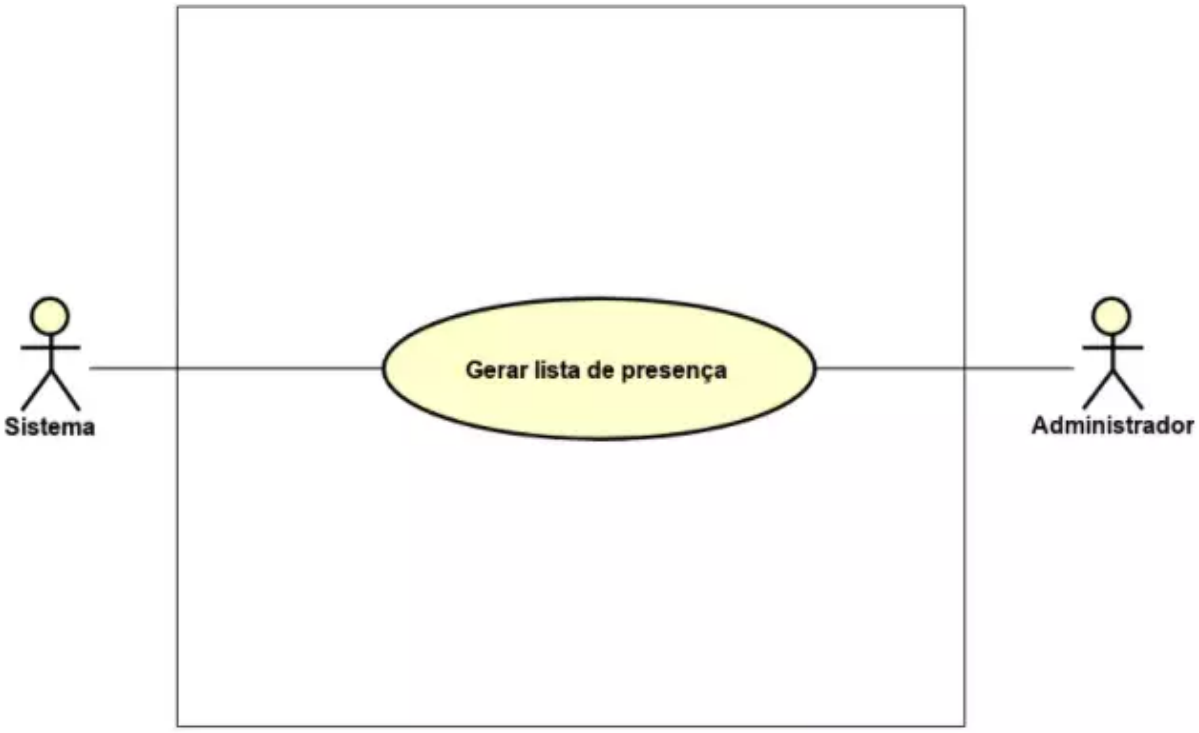
ANEXO A — DIAGRAMAS DE CASOS DE USO






Alterar titulação
<<extend>>
Atualizar dados
Docente
Sistema
<<extend>>
Alterar telefone
<<extend>>
Alterar email


Alertar conclusão de carga horária
<<extend>>
Contabilizar carga horária
Administrador
<<include>>
Registrar presença
<<extend>>
Docente
Inscrever em evento
Sistema
<<include>>
Cadastrar eventos


Gerar lista de presença
Sistema
Administrador

ANEXO B — ESPECIFICAÇÃO DOS CASOS DE USO

Nome	Acessar diferentes módulos
Sumário	Usuários acessam determinadas funções do sistema dependendo do seu tipo de usuário
Pré-condições	Logar
Atores	Usuários docentes, usuários administradores, sistema
Fluxo Principal	1. Usuário insere seu usuário e senha 2. Sistema valida as informações 3. Sistema valida o tipo de usuário (Docente ou Administrador) 4. Sistema permite acesso às funcionalidades do sistema a partir do tipo de usuário.
Fluxos Alternativos	-
Exceções	Usuário não encontrado, Usuário desativado
Pós-condições	Usuário só pode acessar as funcionalidades permitidas para seu tipo de usuário: Docente: Visualizar eventos, inscrever-se em eventos, alterar seus dados pessoais (Com exceção da carga horária) Administrador: CRUD Eventos, CRUD Usuários, relatórios.
Regras de negócio	-

Nome	Cadastrar Evento
Sumário	Usuário do sistema incluindo um novo evento
Pré-condições	Estar logado, usuário administrador
Atores	Usuário administrador, sistema
Fluxo Principal	1. Usuário administrador acessa a tal Eventos 2. Usuário seleciona adicionar novo evento 3. O sistema apresenta um formulário para inserção de um novo evento. Possui os campos: - Título do evento (Editável) - Categoria - Palestrante(Editável) - Data e Horário do Evento (DatePicker) - Local (Editável) - Carga Horária (Editável) 4. Clicar no botão de criar novo evento.
Fluxos Alternativos	-
Exceções	Campos nulos ou inválidos
Pós-condições	-
Regras de negócio	-

Nome	Atualizar dados
Sumário	Usuário docente modificando informações pessoais
Pré-condições	Estar logado
Atores	Usuário docente, Sistema
Fluxo Principal	1. Usuário docente acessa a tab perfil 2. Clica no ícone editar 3. Insere o novo dado 4. Seleciona salvar 5. O sistema salva o novo dado no banco
Fluxos Alternativos	-
Exceções	Campo vazio
Pós-condições	-
Regras de negócio	-

Nome	Inscrever em evento
Sumário	Usuário docente inscreve-se em evento
Pré-condições	Estar logado, evento cadastrado e em período de inscrição
Atores	Usuário docente, Sistema
Fluxo Principal	1. Usuário docente acessa a aba eventos 2. Clica no evento ao qual deseja participar 3. Verifica a as informações do evento 4. Clica em participar 5. O Sistema insere o usuário na lista de inscritos mas sua presença está por default como false
Fluxos Alternativos	-
Exceções	-
Pós-condições	-
Regras de negócio	-

Nome	Registrar presença
Sumário	Usuário docente registra sua presença através da ção com QRCode
Pré-condições	Estar logado, estar cadastrado no evento
Atores	Usuário docente, Sistema
Fluxo Principal	1. Usuário docente acessa a aba eventos 2. Clica no evento ao qual deseja registrar frequência 3. Usuário interage com QRCode disponibilizado no evento. 4. Sistema modifica o status isPresent para true
Fluxos Alternativos	-
Exceções	Código do QRCode Inválido, Evento ainda não está na data de acontecimento
Pós-condições	
Regras de negócio	A presença é contabilizada apenas quando o usuário interage com o QRCode e se a data estiver válida.

Nome	Baixar certificado
Sumário	Usuário docente baixa seu certificado de participação.
Pré-condições	Estar logado, estar presente no evento
Atores	Usuário docente, Sistema
Fluxo Principal	1. Usuário docente acessa a aba eventos 2. Clica no evento ao qual deseja registrar frequência 3. Usuário clica no botão de download 4. Sistema gera o certificado de participação
Fluxos Alternativos	-
Exceções	Usuário não presente no evento
Pós-condições	-
Regras de negócio	-

Nome	Notificar Evento
Sumário	Sistema envia uma notificação aos usuários se cadastrado um novo evento
Pré-condições	Cadastrar evento
Atores	Usuário administrador, Sistema
Fluxo Principal	1. Usuário administrador cadastra evento 2. Sistema verifica a inserção de um novo evento 3. Sistema emite alerta aos usuários docentes
Fluxos Alternativos	-
Exceções	-
Pós-condições	-
Regras de negócio	-