UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GABRIELA DE SOUZA GUEDES HENRIQUES

HENRIQUE ALVES BATOCHI

DÁRIO - APLICATIVO SOLIDÁRIO

CURITIBA

2023

GABRIELA DE SOUZA GUEDES HENRIQUES
HENRIQUE ALVES BATOCHI

DÁRIO - APLICATIVO SOLIDÁRIO

Dário - Solidarity App

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharelado em Sistemas de Informação e Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Orientadora: Profª. Drª Ana Cristina Barreiras Kochem Vendramin

Coorientadora: Profª. Drª Keiko Verônica Ono Fonseca

CURITIBA

2023

GABRIELA DE SOUZA GUEDES HENRIQUES
HENRIQUE ALVES BATOCHI

DÁRIO - APLICATIVO SOLIDÁRIO

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharelado em Sistemas de Informação e Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 01/dezembro/2023

Leyza Baldo Dorini
Dotorado
Universidade Tecnológica Federal do Paraná

Paulo Roberto Bueno
Doutorado
Universidade Tecnológica Federal do Paraná

Ana Cristina Barreiras Kochem Vendramin
Doutorado
Universidade Tecnológica Federal do Paraná

Keiko Verônica Ono Fonseca
Doutorado
Universidade Tecnológica Federal do Paraná

CURITIBA

2023

AGRADECIMENTOS

Gostaríamos de expressar nossa sincera gratidão a todas as pessoas que contribuíram de maneira significativa para a realização deste trabalho. Este Trabalho de Conclusão de Curso representa o resultado de esforços coletivos e o apoio de muitos indivíduos que tornaram possível a conclusão deste projeto.

Em primeiro lugar, queremos agradecer às nossas orientadoras, Profª. Drª Ana Cristina Barreiras Kochem Vendramin e Profª. Drª Keiko Verônica Ono Fonseca, pela orientação valiosa, paciência e incentivo constante ao longo deste processo. O conhecimento e a orientação de ambas foram fundamentais para o desenvolvimento deste trabalho, e estamos profundamente gratos por toda a dedicação.

Expressamos nossa gratidão aos membros da banca examinadora, Profª. Drª Leyza Baldo Dorini e Prof. Dr. Paulo Roberto Bueno, por dedicarem seu tempo e expertise na análise e avaliação deste trabalho. Suas sugestões e *insights* foram extremamente importantes para aprimorar a qualidade deste trabalho.

Eu, Gabriela, agradeço aos meus pais, Valéria e Claudio, por todo o seu apoio ao longo da minha jornada acadêmica. Por terem me apoiado em seguir meus sonhos e me suportar nos momentos de estresse e dúvidas, por todo incentivo, compreensão e palavras de ânimo, tudo isso foi fundamental para enfrentar os desafios encontrados neste processo. Um agradecimento especial à minha irmã, Letícia, por sempre me apoiar e me escutar, por sempre me ajudar e auxiliar em diversos trabalhos acadêmicos nesse período, e especialmente pelas contribuições e incentivos para a realização deste trabalho.

Aos meus amigos, meu alicerce durante toda a jornada acadêmica, meu sincero agradecimento. Em momentos de estresse, dúvidas e dificuldades, vocês foram meu porto seguro. Sem o apoio de vocês, com seus ombros amigos e ouvidos sempre abertos, o percurso teria sido muito mais complexo e difícil de enfrentar.

E ao Henrique, meu parceiro de TCC e grande amigo, agradeço em dose dupla. Primeiramente, pelo convite para embarcar nesse projeto incrível. Sua parceria foi vital para o sucesso dessa empreitada. Mas não para por aí; obrigada também pela amizade que se fortaleceu ao longo desse desafio. Ter você ao meu lado tornou cada etapa mais leve e significativa.

Por último, e que com certeza foram super importantes, agradeço a todos os professores e colegas de curso que contribuíram para a minha formação acadêmica, a todos os professores que auxiliaram na minha formação profissional e marcaram de alguma forma minha vida pessoal, levarei pra sempre vários momentos e conversas na minha mente e no coração.

Este trabalho não seria possível sem o apoio e as contribuições de cada uma dessas pessoas, e por isso, expresso minha profunda gratidão a todos vocês.

Eu, Henrique, agradeço aos meus pais, Laércio e Cleonice, por cada momento de apoio e de palavras de incentivo que permearam essa jornada acadêmica. Com certeza foram de total importância em momentos decisivos e possibilitaram transpor desafios que por muitas vezes me pareciam intransponíveis. Poder contar com a fé inabalada deles em mim possibilitou a conclusão dessa jornada. A minha mãe, de forma especial, que desde o início desse ano vem encarando o tratamento de um câncer e que jamais deixou se abater, seu exemplo de resiliência e perseverança moldaram não apenas para si mesma e suas lutas, mas para mim um suporte continuo e presente incentivando, questionando e ajudando a conclusão deste trabalho. Ao meu irmão Bruno que sempre vibrou com as minhas vitórias e sempre esteve presente torcendo por mim, sua amizade e parceria ao longo dos anos assegura dentro de mim um lugar especial e uma pessoa de suporte que para tudo sei que posso contar.

Uma jornada como essa não se faz sozinho e tive a chance de encontrar pessoas que marcaram para sempre a memórias dessa conquista, mais do que apenas celebrar vitórias aqueles que permaneceram ao meu lado em momentos de dificuldade para sempre irei carregar o mais leve dos sentimentos que uma pessoa pode ter por outra, a gratidão.

Dentre elas preciso agradecer em especial minha amiga Geovana Franco Santos que desde os semestres iniciais partilhou de dificuldades, compartilhou conhecimentos, ajudou em trabalhos, errou em cálculos comigo em madrugadas de estudo, me fez compreender partes de Circuitos Elétricos, dividimos moradia e construímos ao longo desses anos memórias, momentos e celebrações que para sempre poderemos nos lembrar com carinho.

A Gabriela, minha parceira de execução deste trabalho e por sorte uma grande amiga para vida. Seu empenho e comprometimento com a execução deste trabalho, partilha de conhecimento, sugestões e descobertas permitiram não apenas desenvolver e concluir esse trabalho como criar momentos únicos de superação e por maiores que fossem as pressões das circunstâncias permitiu que essa jornada fosse leve, alegre e repleta de uma total parceria. No grande palco da vida para essas duas mulheres sempre estarei aplaudindo cada vitória, torcendo pela realização de cada desejo e quando necessário reafirmando a grandiosidade delas.

Aos professores do DAINF e demais departamentos que compartilharam seus conhecimentos, desafiaram nossos limites - alguns não precisavam ter desafiado tanto - as oportunidades, conselhos e sugestões e a cada momento dedicado à Educação, meu mais sincero obrigado.

Não poderia encerrar essa jornada sem agradecer a existência, manutenção e expansão da Universidade Pública Brasileira que de vez enquanto é atacada, questionada e por muitas vezes não celebrada por toda sociedade. Sua existência mudou os caminhos diretos da minha vida, abriu portas e transformou a linha do meu horizonte de forma permanente e sólida. Poder saber que para sempre irei carregar seu nome, legado e reconhecimento da competência intelectual me enche de orgulho.

RESUMO

O projeto Dário, uma proposta no cenário acadêmico, proporciona uma solução tanto para estudantes em busca de atividades complementares quanto para instituições que almejam gerenciar eficientemente eventos sociais voluntários. No âmbito desse projeto, foi desenvolvido um sistema que estabelece uma conexão significativa entre indivíduos interessados em participar de ações sociais voluntárias e os eventos catalogados por instituições. Os voluntários dispõem de um aplicativo móvel para *iOS*, onde têm acesso a eventos publicados, podendo consultar, inscrever-se e acompanhar informações sobre o desenvolvimento dessas atividades. Esses eventos são analisados e selecionados com base em um sistema de recomendação personalizado, levando em consideração as preferências de cada usuário. Além disso, o aplicativo oferece recursos para gerenciar atividades anteriores, abrangendo áreas sociais, culturais, esportivas, entre outras. As instituições, por sua vez, beneficiam-se de uma aplicação web, permitindo não apenas o seu cadastro, mas também o registro de eventos na plataforma. Essas instituições têm a capacidade de monitorar o número de inscritos e enviar atualizações relevantes relacionadas aos eventos. Esse ecossistema interativo busca integrar de maneira eficaz as necessidades dos voluntários e das instituições, proporcionando uma plataforma dinâmica para a promoção e execução de ações sociais voluntárias.

Palavras-chave: atividades voluntárias; projetos sociais; aplicativo.

ABSTRACT

The Dário project, a proposal in the academic scenario, provides a solution for both students seeking complementary activities and institutions aiming to efficiently manage voluntary social events. In the scope of this project, a system has been developed to establish a meaningful connection between individuals interested in participating in voluntary social actions and events cataloged by institutions. Volunteers have access to a mobile application for iOS, where they can browse, enroll, and follow information about the development of these activities. These events are analyzed and selected based on a personalized recommendation system, taking into account the preferences of each user. Additionally, the application offers features to manage previous activities, covering social, cultural, sports, and other areas. Institutions, in turn, benefit from a web application, enabling not only their registration but also the recording of events on the platform. These institutions have the ability to monitor the number of participants and send relevant updates related to the events. This interactive ecosystem seeks to effectively integrate the needs of volunteers and institutions, providing a dynamic platform for the promotion and execution of voluntary social actions.

Keywords: volunteer; social projects; application.

LISTA DE FIGURAS

LISTA DE SIGLAS

API	Interface de Programação de Aplicação, do inglês *Application Programming Interface*
ARC	Contagem Automática de Referência, do inglês *Automatic Reference Counting*
CSS	Folhas de estilo em cascata, do inglês *Cascading Style Sheets*
FBC	Filtragem Baseada em Conteúdo
FC	Filtragem Colaborativa
HTML	Linguagem de Marcação de Hipertexto, do inglês *HyperText Markup Language*
HTTP	Protocolo de Transferência de Hipertexto, do inglês *Hypertext Transfer Protocol*
MVC	Controlador de visualização de modelo, do inglês *Model-View-Controller*
SBC	Sociedade Brasileira de Computação
SQL	Linguagem de consulta estruturada, do inglês *Structured Query Language*
URL	Localizador Padrão de Recursos, do inglês *Uniform Resource Locator*
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

No contexto acadêmico, as atividades complementares e os projetos de extensão desempenham um papel fundamental na formação dos estudantes, proporcionando uma experiência prática e enriquecedora para além das disciplinas curriculares. Essas atividades têm como objetivo complementar o aprendizado teórico, promovendo o desenvolvimento de habilidades, competências e valores essenciais para a formação integral dos estudantes.

As atividades complementares compreendem uma variedade de ações extracurriculares, tais como participação em eventos acadêmicos, cursos, seminários, estágios, monitorias, trabalhos voluntários, entre outras. Elas visam ampliar o conhecimento dos estudantes em áreas relacionadas ao seu curso, permitindo que eles aprofundem temas específicos, desenvolvam habilidades práticas e estabeleçam conexões com o mercado de trabalho (UTFPR, 2018a).

Por outro lado, os projetos de extensão são iniciativas voltadas para a aplicação do conhecimento acadêmico em benefício da sociedade. Esses projetos buscam levar os conhecimentos e as competências adquiridas na universidade para além dos muros acadêmicos, atendendo às demandas e necessidades da comunidade. Eles podem abranger diferentes áreas, tais como saúde, educação, cultura, meio ambiente, entre outras, e envolvem a realização de ações sociais, programas, eventos e serviços em parceria com instituições públicas, privadas e da sociedade civil (UTFPR, 2018b).

Tanto as atividades complementares quanto os projetos de extensão desempenham um papel relevante na formação cidadã e no desenvolvimento profissional dos estudantes. Ambos permitem que os estudantes vivenciem situações reais, apliquem seus conhecimentos na prática e desenvolvam competências como trabalho em equipe, liderança, responsabilidade social e empreendedorismo.

Nesse contexto, com base em uma demanda de professores responsáveis pelas atividades complementares da Universidade Tecnológica Federal do Paraná (UTFPR) e na necessidade de incentivar a participação da sociedade em trabalhos sociais, este trabalho apresenta o projeto de desenvolvimento de um aplicativo que integra todas essas características. O objetivo é criar um sistema, chamado Dário, que conecte pessoas interessadas em realizar ações sociais voluntárias a eventos cadastrados por instituições, fornecendo informações sobre esses eventos e permitindo que os usuários se inscrevam e acompanhem sua participação. Além disso, o Dário permite o gerenciamento de atividades já realizadas pelos voluntários.

As instituições cadastradas utilizam um sistema web para cadastrar eventos, acompanhar o número de inscritos, visualizar as informações pessoais dos voluntários que desejam participar e emitir atualizações.

Espera-se que esse sistema facilite a conexão entre voluntários e instituições e incentive a participação em ações sociais. Portanto, este trabalho apresenta uma solução para promover ações voluntárias e envolver a sociedade em projetos que contribuam para o bem comum.

1.1 Justificativa

A maioria dos estudantes enfrenta dificuldades em completar suas horas complementares de forma fácil e eficiente, especialmente diante da necessidade de participar de atividades de diferentes formações, obter e guardar os certificados de participação em eventos emitidos pelas instituições parceiras.

O desenvolvimento do aplicativo Dário, cujo nome foi intencionalmente inspirado na palavra “solidário”, baseou-se nessa necessidade da comunidade acadêmica.

A palavra “solidário” tem raízes no latim “solidarius” e está intrinsecamente ligada à “solidariedade”. Ela expressa união e cooperação, onde indivíduos se reúnem em prol de causas comuns, compartilhando a responsabilidade de apoiar uns aos outros. A solidariedade é um valor fundamental que promove a justiça, igualdade e harmonia na sociedade.

O nome “Dário” foi escolhido para o aplicativo devido à sua associação com a solidariedade. Ele representa a plataforma que une pessoas com interesses semelhantes em ações voluntárias e solidárias, trabalhando juntas para causas sociais. Essa escolha homenageia a etimologia da palavra “solidário” e reflete o compromisso do aplicativo em incentivar a solidariedade e a participação da sociedade em projetos de bem comum.

Além disso, o serviço voluntário é uma das vertentes do exercício da cidadania, em que o cidadão é responsável por participar ativamente na transformação de uma sociedade, tornando-a gradativamente mais justa e igualitária. Por isso, é importante criar mecanismos que facilitem a participação em eventos sociais e promovam a aproximação dos usuários com instituições que necessitam de voluntários.

O aplicativo Dário foi desenvolvido com o objetivo de auxiliar os estudantes que necessitam completar suas atividades complementares a se voluntariem em eventos, disponibilizados por instituições, de forma prática. Além disso, com um sistema de recomendação, que auxilia no momento de procura de eventos através de preferências configuradas e iteração com eventos.

Para as instituições parceiras, o portal web integrado ao aplicativo Dário visa facilitar o cadastro e atualização de eventos, cadastro de formas de doações, verificação de número de inscritos e publicação de informações aos voluntários inscritos.

Assim, o desenvolvimento do aplicativo Dário busca promover a aproximação da sociedade em prol de uma boa causa, utilizando a tecnologia para facilitar a prática de atividades sociais alinhadas com as necessidades tanto do usuário quanto das instituições parceiras.

1.2 Objetivo Geral e Objetivos Específicos

O objetivo geral deste trabalho é desenvolver um aplicativo móvel e uma aplicação web capaz de conectar estudantes e cidadãos a eventos cadastrados por instituições parceiras, com o intuito de promover a participação em eventos sociais e incentivar o engajamento da comunidade em causas sociais.

Os objetivos específicos do presente trabalho são:

- Desenvolver o aplicativo para gerenciamento de atividades sociais realizadas por estudantes;
- Desenvolver a interface web para cadastro, acompanhamento e apresentação dos eventos pelas Instituições associadas;
- Criar métodos e rotina para averiguar a autenticidade dos dados institucionais cadastrados;
- Desenvolver um sistema *back-end* capaz de receber e responder requisições contextuais tanto do aplicativo quanto da interface web e hospedá-lo em um servidor na nuvem;

1.3 Estrutura do trabalho

O trabalho foi estruturado em seis capítulos. Este capítulo contém a introdução do tema a ser abordado, a motivação para a realização do trabalho e os objetivos geral e específicos. O Capítulo 2 apresenta a contextualização de conceitos importantes para o entendimento do tema. O Capítulo 3 descreve o projeto de *software* criado durante o planejamento do aplicativo Dário. O Capítulo 4 apresenta o funcionamento do aplicativo e da aplicação *web*, detalhando o sistema de recomendação implementado. O Capítulo 5 apresenta a conclusão do trabalho e as perspectivas dos trabalhos futuros.

2 CONCEITOS E REFERENCIAL TEÓRICO

O objetivo deste capítulo é apresentar os principais conceitos e uma visão abrangente do contexto do trabalho. Essa base de conhecimento é essencial para compreender e avaliar adequadamente as etapas seguintes do trabalho e os resultados obtidos.

2.1 Atividades Complementares

As atividades complementares são componentes obrigatórios presentes na matriz curricular de muitos cursos de graduação e a Sociedade Brasileira de Computação (SBC) (CORREIA *et al.*, 2017) aponta algumas atividades especificamente para os cursos de computação e que podem enriquecer o currículo dos estudantes. Essas visam proporcionar aos estudantes uma formação mais ampla, além dos conhecimentos específicos do curso, através da participação em atividades extracurriculares. Essas atividades podem incluir a participação em projetos sociais, cursos, palestras, entre outros.

O voluntariado é uma das atividades complementares, a qual é descrita pela participação social em que indivíduos dedicam seu tempo e esforço para ajudar outras pessoas ou causas sociais.

O voluntariado pode ocorrer em diversas áreas, como educação, saúde, meio ambiente, assistência social, entre outras. Ele é uma oportunidade para os estudantes contribuírem para a sociedade, adquirirem experiências práticas e desenvolverem habilidades interpessoais.

2.2 Desenvolvimento de Aplicativos

O desenvolvimento de aplicativos móveis tem se tornado uma tendência, pois oferecem uma forma conveniente e acessível de acesso à informação e interação entre usuários. Os aplicativos proporcionam mobilidade e facilidade de uso, permitindo que os estudantes procurem e acessem as ações voluntárias e se inscrevam nos eventos que mais se interessarem diretamente de seus dispositivos móveis.

Esses aplicativos podem ser desenvolvidos para auxiliar a conexão entre voluntários e eventos, além de se beneficiar de recursos tecnológicos como geolocalização, notificações de eventos, integração com redes sociais e sistema de avaliação. Esses recursos possibilitam uma experiência mais interativa e personalizada, aumentando o engajamento dos usuários e facilitando a organização dos eventos pelas instituições.

2.3 Aprendizagem Computacional

Aprendizagem Computacional ou, como é mais conhecido, *Machine Learning*, é uma subárea da Ciência da Computação e da Estatística que estuda e desenvolve algoritmos e técnicas que aprimoram o seu desempenho e acurácia segundo uma medida de erro e com a experiência que é dada por conjuntos de dados de treinamento (MITCHELL *et al.*, 2007).

Os algoritmos envolvem diversos conceitos e técnicas de Inteligência Artificial e Otimização e, por isso, muitas vezes, os temas são relacionados na literatura.

Algumas das aplicações de aprendizagem computacional são ferramentas de busca, detecção de fraudes em cartões de crédito, segmentação de clientes (usuários), entre outros (RUSSELL, 2010). Em suma, a aprendizagem computacional pode ser utilizada em qualquer meio em que um algoritmo estático baseado em regras não atende as necessidades.

Segundo Russell (2010), a tarefa de aprendizagem computacional pode ser dividida em:

- **Aprendizagem Supervisionada:** o algoritmo recebe exemplos de entradas e saídas e, a partir desses,"aprende" uma regra que mapeia as entradas em saídas;
- **Aprendizagem Não Supervisionada:** não são fornecidos exemplos de saída, somente de entrada. Então, é o próprio algoritmo que define os agrupamentos e os tipos de saída a serem devolvidos.

Existem diversos tipos de algoritmos de aprendizagem computacional cada um com uma metodologia e finalidade diferente e tem-se como exemplos a Aprendizagem por Árvore de Decisão, as Redes Bayesianas e a Análise de Agrupamentos.

2.3.1 Aprendizagem por Árvore de Decisão

A Aprendizagem por Árvore de Decisão é um algoritmo de aprendizagem supervisionada que utiliza uma árvore de decisão como modelo preditivo que mapeia as observações de um item para concluir sobre o valor desejado acerca do item (SANTANA, 2018).

As árvores de decisão recebem como entrada um conjunto de atributos e retornam uma decisão que é o valor predito para a entrada. Sua estrutura é formada por nós e ramos, sendo que cada nó contém um teste em um atributo e cada ramo representa um possível valor do atributo. Os nós que não possuem nenhum ramo saindo dele são chamados folhas e especificam o valor de retorno se a folha for atingida (MAIMON; ROKACH, 2014).

2.3.2 Redes Bayesianas

As Redes Bayesianas são modelos baseados em grafos para tomada de decisão baseado na incerteza, onde os nós representam as variáveis (discreta ou contínua), e os arcos

representam a conexão direta entre eles (BEN-GAL, 2007). Elas possuem esse nome, pois utilizam a probabilidade bayesiana para determinar as chances de ocorrência das diversas possibilidades.

2.3.3 Análise de Agrupamentos

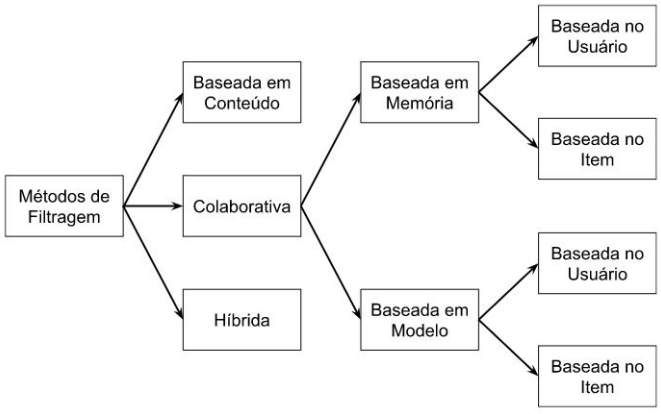
Análise de Agrupamentos ou, como é mais conhecido, *Clustering* é a tarefa de agrupar um conjunto de dados de forma que os objetos que pertencem ao mesmo grupo, chamado de *cluster*, sejam mais semelhantes de acordo com alguns critérios entre eles do que em relação aos objetos que estão em outros grupos (*clusters*) (BAILEY, 1994).

2.4 Sistemas de Recomendação

Diante da crescente exposição a diferentes tipos de informações, faz-se necessário ajudar as pessoas a encontrarem aquilo que procuram e, nesse contexto, é que surgiram os sistemas de recomendação. Estes sistemas, através de modelagem de dados e aplicação de algoritmos, tentam prever a nota, *rating*, ou preferência de um usuário a um determinado item (FRANCESCO, 2011).

Os sistemas de recomendação são um tipo de Sistema de Aprendizagem Computacional e fazem uso de algoritmos que são divididos em três grupos (MELVILLE; SINDHWANI, 2010): Filtragem Colaborativa (FC), Filtragem Baseada em Conteúdo (FBC) e Sistema de Recomendação Híbrida.

Figura 1 – Métodos de Filtragem



Fonte: Autoria Própria (2023).

Os métodos de filtragem e como eles podem ser organizados e divididos pode ser visto na Figura 1, porém, além desses 3 grupos, quando não existem dados suficientes para realizar esses métodos de filtragem, podemos utilizar um outro método chamado Partida a Frio.

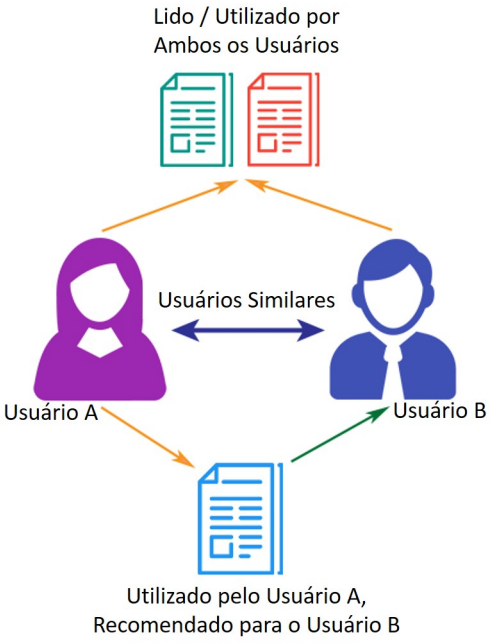
2.4.1 Filtragem Colaborativa

A abordagem da FC ignora as características do conteúdo e foca na interação entre o usuário e o conteúdo. A abordagem parte do princípio que o sistema não precisa saber as características do conteúdo, mas sim quais conteúdos o usuário consumiu para identificar quais outros usuários tiveram o mesmo comportamento de consumo. Dessa forma, é possível "trocar" recomendações entre usuários semelhantes ao processar o coletivo (SANTANA, 2018).

Essas recomendações são muito semelhantes "Os usuários que assistiram *Star Wars* também assistiram Senhor dos Anéis". Essa abordagem tira o usuário da sua lista de preferências e não necessita da definição de similaridade de conteúdo já que as características são ignoradas. A atenção especial desse método é com relação à modelagem da "força de interação" do usuário com o conteúdo para representar o quanto o usuário gostou do conteúdo.

Na prática, esse valor pode ser a nota dada pelo próprio usuário ao conteúdo, a quantidade de vezes que ele acessou o conteúdo, quanto tempo ele ficou interagindo com o conteúdo ou qualquer outra métrica que passe essa ideia de preferência (SANTANA, 2018).

Figura 2 – Filtragem colaborativa



Fonte: Adaptado de Medium (2020).

Através dos itens já comprados ou interagidos pelo usuário B, procura-se por usuários semelhantes que tiveram um comportamento igual ou similar, isto é, que interagiram com os mesmos itens, e seleciona itens com os quais o usuário B ainda não interagiu. Então, recomenda-se aqueles com maior nota entre os usuários (HERLOCKER *et al.*, 2004) isso pode ser visto na Figura 2.

Uma das principais desvantagens desse método de filtragem é a própria escala da solução, visto que processar uma matriz de todos os usuários com todos os conteúdos é um desafio

computacional. Um detalhe que prejudica ainda mais é que essa matriz pode ser esparsa, podendo ter mais valores faltando do que preenchidos, visto que, menos de 3% da matriz de fato é preenchida.

Outra desvantagem dessa abordagem é a necessidade de uma quantidade considerável de registros e *feedback* dos usuários para começar a gerar a recomendação (SANTANA, 2018).

2.4.1.1 Filtragem Colaborativa Baseada em Memória

Segundo LÁZARO (2010), a FC baseada em memória utiliza toda a base de dados que contém da relação entre usuários (u) e item (i) para fazer as predições ou recomendações. Uma matriz usuário-item R de dimensão n x m pode ser utilizada para representar a relação dos n usuários sobre os m itens. Cada célula, r_{nm} da matriz R pode conter: valores de avaliações, valores zeros ou valores um. O valor um significa que o n-ésimo usuário comprou o m-ésimo item, e o valor zero significa que o n-ésimo usuário não comprou o m-ésimo item.

A FC baseada em memória pode ser dividida em baseada no usuário e baseada no item. A filtragem baseada no usuário busca encontrar vizinhos, ou seja, um conjunto de usuários que possuam gostos similares ao usuário ativo, como, por exemplo, usuários que comprem itens similares. Para cálculo da similaridade são atribuídos pesos a todos os usuários indicando seu grau de similaridade com o usuário ativo. Segundo LÁZARO (2010), as medidas de similaridade mais conhecidas são o coeficiente de correlação de Pearson e o método do cosseno, os quais são descritos a seguir.

O coeficiente de correlação de Pearson, $sim(a,b)$ representa a similaridade entre os usuários a e b; $r_{a,i}$ e $r_{b,i}$ são as avaliações dos usuários a e b para o item i respectivamente; e $\bar{r}_a$ e $\bar{r}_b$ são os valores médios de avaliações do usuário a e b, respectivamente, como pode ser visto na Equação 1 (MUKAKA, 2012):

$$sim(a,b) = \frac{\sum_i (r_{a,i} - \bar{r}_a)(r_{b,i} - \bar{r}_b)}{\sqrt{\sum_i (r_{a,i} - \bar{r}_a)^2} \sqrt{\sum_j (r_{b,i} - \bar{r}_b)^2}} \quad (1)$$

O Método do cosseno (MEDEIROS, 2013) pode ser utilizado para definir a matriz de similaridade, que representa o grau de compatibilidade entre as preferências dos usuários. Essa matriz pode ser obtida utilizando a similaridade por cosseno, que utiliza o valor do cosseno do ângulo formado entre dois vetores para determinar o nível de similaridade. Considerando uma matriz usuário-item de ordem mxn, pode-se obter a similaridade entre dois itens A e B através do cosseno entre os vetores de tamanho n, encontrados nas colunas de índices correspondentes a esses dois itens na matriz. A Equação 2 apresenta o cálculo da similaridade por meio do cosseno entre dois vetores (MEDEIROS, 2013):

$$\cos(A,B) = \frac{AB}{\|A\| * \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \sqrt{\sum_{i=1}^{n} (B_i)^2}} \quad (2)$$

Calculada a similaridade, os vizinhos são encontrados através de métodos como o *Center-based* ou Vizinhança agregada (SARWAR *et al.*, 2000). Depois que os vizinhos são formados, os sistemas utilizam algoritmos diferentes capazes de combinar suas preferências e assim fazer uma predição ou recomendação para o usuário ativo (SARWAR *et al.*, 2001).

A FC baseada em memória possui algumas limitações como a escalabilidade. Quando o número de usuários e itens de um sistema cresce muito, o custo computacional de calcular os k vizinhos mais similares torna-se bastante oneroso (FORSATI *et al.*, 2014).

2.4.1.2 Filtragem Colaborativa baseada em Modelo

Na FC baseada em modelo, os dados sobre os usuários e os itens são utilizados para criar um modelo que é, então, usado para fazer predições ou recomendações de itens. Segundo Sarwar *et al.* (2001), para a construção do modelo podem ser utilizados algoritmos de aprendizagem de máquina, como: redes bayesianas, que formulam um modelo probabilístico para a FC; clusterização, que trata a FC como um problema de classificação; métodos baseados em regras, que aplicam algoritmos que descobrem regras de associação entre itens previamente adquiridos e gera a recomendação com base na força da associação entre os itens.

Assim como a FC baseada em memória, a FC baseada em modelo também pode ser dividida em baseada no usuário e no item.

A filtragem baseada no usuário constrói um modelo com base no usuário e, para isso, pode utilizar diferentes técnicas como clusterização, onde usuários similares são agrupados em um mesmo *cluster*. Este modelo é, então, utilizado para estimar a probabilidade de que o usuário alvo pertença a um determinado *cluster*, que pode então ser utilizado para realizar as predições para o usuário ativo.

Na filtragem baseada no item a construção do modelo é com base nos itens, utilizando medidas de similaridade como a Probabilidade Condicional que pode ser expressa pela Equação 3 (DESHPANDE; KARYPIS, 2004):

$$P(j|i) = \frac{Freq(ij)}{Freq(i)} \tag{3}$$

Onde $P(j|i)$ é a probabilidade condicional de se comprar o item j dado que o produto i já tenha sido comprado e $Freq(X)$ representa o número de consumidores que compraram os itens no conjunto X. Esta probabilidade corresponde ao número de consumidores que compraram ambos os itens i e j dividido pelo número total de usuários que compraram i. O resultado da construção deste modelo pode ser expresso através de uma matriz de similaridade S, que contém os resultados da probabilidade condicional, conforme Tabela 1.

Utilizando esta matriz, a recomendação de uma lista contendo N itens pode ser feita ao usuário ativo através do algoritmo de recomendação $Top - N$, proposto por Karypis (2001), conforme será exemplificado a seguir.

Tabela 1 – Matriz de similaridade entre itens S

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
i_1	1	0,1	0	0,3	0,2	0,4	0	0,1
i_2	0,1	1	0,8	0,9	0	0,2	0,1	0
i_3	0	0,8	1	0	0,4	0,1	0,3	0,5
i_4	0,3	0,9	0	1	0	0,3	0	0,1
i_5	0,2	0	0,4	0	1	0,1	0	0
i_6	0,4	0,2	0,1	0,3	0,1	1	0	0,1
i_7	0	0,1	0,3	0	0	0	1	0
i_8	0,1	0	0,5	0,1	0	0,1	0	1

Fonte: Adaptado de Hahsler(2010).

Considerando que o conjunto de itens comprados pelo usuário ativo (u_a) seja representado por $U = i_1, i_5, i_8$, primeiramente o algoritmo identifica um conjunto C de itens candidatos a serem recomendados unindo os K itens mais similares i_1, i_2, ... , i_k para cada item $i \in U$. Desta união, devem ser removidos os itens que já foram comprados pelo usuário ativo. Considerando $K = 3$ e utilizando os dados da matriz S apresentada na Tabela 1, o conjunto C será formado pelos itens $C = i_3, i_4, i_6$. Agora a similaridade entre cada item $c \in C$ e o conjunto U é calculada através da soma das similaridades entre todos os itens $i \in U$ e c, usando somente os k itens mais similares de j. O resultado deste cálculo da similaridade é mostrado em Tabela 2, onde os valores em negrito correspondem aos três itens mais similares aos itens $i \in U$, estes que por sua vez correspondem às linhas destacadas.

Tabela 2 – Soma das similaridades entre todos os itens $i \in Uec$

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
i_1	1	0,1	0	**0,3**	**0,2**	**0,4**	0	0,1
i_2	0,1	1	0,8	0,9	0	0,2	0,1	0
i_3	0	0,8	1	0	0,4	0,1	0,3	0,5
i_4	0,3	0,9	0	1	0	0,3	0	0,1
i_5	**0,2**	0	**0,4**	0	1	**0,1**	0	0
i_6	0,4	0,2	0,1	0,3	0,1	1	0	0,1
i_7	0	0,1	0,3	0	0	0	1	0
i_8	**0,1**	0	**0,5**	**0,1**	0	0,1	0	1
			0,9	0,4		0,5		

Fonte: Adaptado de Hahsler(2010).

Por fim, os itens em C são classificados em ordem decrescente em relação à similaridade, e os primeiros N itens são selecionados como o conjunto de itens $Top - N$ a serem recomendados. Neste exemplo, os $N = 2$ itens recomendados seriam i_6 e i_3.

A FC baseada em modelo consegue superar o problema da escalabilidade presente na FC baseada em memória, pois quando uma recomendação é requisitada, as relações de similaridade entre os usuários ou entre os itens podem ser encontradas através do modelo previamente construído e não calculada em tempo real, como ocorre na FC em memória (LÁZARO, 2010).

2.4.2 Filtragem Baseada em Conteúdo

A FBC gera novas recomendações com base na similaridade de conteúdos que um usuário consumiu ou interagiu no passado. O processo básico, que pode ser visto na Figura 3 consiste em cruzar os atributos do perfil do usuário (interesses e preferências) com os atributos dos itens, para recomendar ao usuário novos itens (LOPS; GEMMIS; SEMERARO, 2011).

Figura 3 – Filtragem Baseada em Conteúdo

Fonte: Adaptado de Medium (2020).

A principal vantagem dessa categoria é que ela não requer muito *feedback* do usuário para começar a recomendar algo “útil”.

2.4.3 Sistema de Recomendação Híbrida

Os sistemas de recomendação híbrida são algoritmos que combinam a FC com a FBC e podem ser desenvolvidos de diversas formas: aplicando os dois separados e juntando os resultados depois, adicionando a capacidade de FC à FBC (ou vice-versa) ou unificando as duas abordagens em um único modelo.

2.4.3.1 Sistemas de Recomendação Sensíveis ao Contexto

Ao contrário dos algoritmos tradicionais, os sistemas de recomendação sensíveis ao contexto incorporam além de informações do item e do usuário, dados do contexto que levaram ao acontecimento do evento como tempo, clima, local, pessoas que estavam juntas, entre outras (PANNIELLO *et al.*, 2009).

2.4.4 Partida a Frio

Um dos problemas mais comuns relacionados à falta de informação em Sistemas de Recomendação é o problema de Partida a Frio (SCHAFER *et al.*, 2007).

Esse problema ocorre quando não é possível realizar recomendações potencialmente úteis devido a um baixo número de avaliações relacionados aos usuários e/ou itens do domínio (BOBADILLA *et al.*, 2012).

Inicialmente, o termo Partida a Frio foi utilizado para referenciar o problema de recomendação em cenários que não existe nenhum avaliação associado ao item ou usuário (SCHEIN *et al.*, 2002).

Entretanto, recentes trabalhos referem-se à Partida a Frio como o problema de recomendação em cenários em que as avaliações são poucas ou estão associadas aos itens ou aos usuários do domínio (BOBADILLA *et al.*, 2012).

O problema de Partida a Frio pode ser identificado sob três categorias: (1) gerar recomendações para usuários pouco participativos; (2) recomendar itens que foram poucos consumidos; e (3) recomendar um item pouco consumido para um usuário pouco participativo (LIKA; KOLOMVATSOS; HADJIEFTHYMIADES, 2014).

2.5 Trabalhos Relacionados

Essa seção apresenta alguns dos trabalhos encontrados que estão relacionados ao presente trabalho.

2.5.1 Conta Comigo

O Conta Comigo é um aplicativo para dispositivos *Android* que conecta voluntários a instituições do terceiro setor, isto é, organizações sem fins lucrativos que atuam pelo bem comunitário, sendo este constituído por fundações, associações comunitárias, organizações não-governamentais (ONGs), entidades filantrópicas, entre outras (BATAGLIN; CANTARELLI, 2019).

O Conta Comigo foi desenvolvido com tecnologia *Flutter* e utilizando o banco de dados *Firebase Cloud Firestore*. O aplicativo visa promover a solidariedade e o engajamento cívico,

facilitando a participação de qualquer pessoa em causas sociais. Ele permite que as organizações divulguem suas necessidades de doações, trabalho voluntário e eventos de arrecadação de fundos, ao mesmo tempo em que os usuários encontram oportunidades de voluntariado de acordo com seus interesses.

A proposta do Conta Comigo é criar uma rede de cooperação, incentivando os indivíduos a incorporarem ações altruístas em sua rotina diária. Através da plataforma, é possível mostrar que contribuir para o bem-estar do próximo é algo mais simples e acessível do que se imagina. Desde pequenas ações, como visitar um asilo, fazer doações de alimentos ou prestar serviços em creches, todas essas iniciativas têm o potencial de causar um grande impacto na vida daqueles que as recebem (BATAGLIN; CANTARELLI, 2019).

2.5.2 *Acropora*

O *Acropora* é uma plataforma informativa que tem como objetivo promover atividades voluntárias, utilizando os campos do terceiro setor e da assistência social como base. O projeto teve como proposta a implementação de um aplicativo, juntamente com a criação de uma estratégia de comunicação eficaz, desenvolvendo um protótipo de aplicativo para *smartphones* que permita aos usuários se conectarem com atividades e ações voluntárias. A plataforma fornece informações sobre projetos, organizações e sua relação com a comunidade, possibilitando o acompanhamento, mapeamento e acesso a informações sobre as instituições e, principalmente, sobre as atividades voluntárias disponíveis. O objetivo é facilitar o envolvimento dos usuários em ações voluntárias, incentivando uma participação mais ativa e um impacto social significativo (SILVA, 2021).

2.5.3 Nós - Nosso Olhar Solidário

O Nós é um aplicativo que visa promover a solidariedade e criar conexões significativas entre pessoas (UTFPR, 2020). Desenvolvido como um projeto de extensão universitária da UTFPR, o aplicativo foi criado para tornar o mundo um lugar melhor, onde todos possam se unir em prol do bem comum (UTFPR, 2020).

O objetivo é promover uma cultura de solidariedade e empatia, onde cada indivíduo possa contribuir para causas importantes de forma fácil e eficiente. Eles acreditam que, juntos, podemos construir uma sociedade mais unida e consciente, onde o apoio mútuo e o cuidado com o próximo sejam valores fundamentais.

2.5.4 Discussões

Ao se buscar por trabalhos relacionados, são encontrados vários aplicativos com o intuito de ofertar trabalhos de voluntariado para quem deseja realizar esse tipo de trabalho, porém não são encontrados trabalhos que relacionem o voluntariado aos estudantes que precisam completar essa etapa nas atividades complementares do seu curso durante a graduação.

Durante a pesquisa não foi encontrado nenhum trabalho que fale sobre certificados de participação nos eventos, que motivou desde o início o desenvolvimento desse trabalho.

No próximo capítulo são descritas as tecnologias pensadas para o desenvolvimento do sistema Dário e como cada uma delas se relaciona com a estrutura do mesmo.

3 PROJETO DE *SOFTWARE*

Para o desenvolvimento do sistema Dário, algumas etapas foram seguidas e estas são descritas a seguir:

- **Definição do escopo do projeto:**

  São estabelecidos os limites e o conjunto de funcionalidades desenvolvidas no projeto, delimitando-se claramente o que o sistema realizará e o que não estará incluído;

- **Definição dos requisitos:**

  É o processo de identificação e documentação das necessidades do sistema, incluindo requisitos funcionais, que descrevem o que o sistema deve fazer, e requisitos não funcionais, que especificam critérios de desempenho, segurança e usabilidade;

- **Definição das tecnologias:**

  É realizada uma pesquisa e uma seleção das ferramentas, linguagens de programação e tecnologias utilizadas no desenvolvimento do projeto. Essa etapa é crucial para escolher as melhores soluções técnicas que atendam às necessidades do sistema proposto;

- **Análise de outros sistemas:**

  Consiste em examinar sistemas similares que já existem no mercado, permitindo identificar as características, funcionalidades e abordagens usadas por esses outros sistemas e que poderá ajudar no processo de execução do projeto;

- **Planejamento da execução do projeto:**

  Envolve a criação de um plano detalhado que descreve como o desenvolvimento será conduzido, incluindo a definição de marcos, prazos, tarefas e recursos necessários;

- **Implementação e teste:**

  O sistema é construído com base nos requisitos e no escopo definidos nas etapas anteriores. Durante essa etapa o código é desenvolvido e implementam-se as funcionalidades. Após a implementação, os testes são realizados para verificar se o sistema funciona conforme o esperado, para se identificar erros e garantir a qualidade do *software* antes de ser disponibilizado.

Esse capítulo detalha o escopo do sistema (Seção 3.1), a modelagem do projeto de *software* apresentando os casos de uso além dos requisitos funcionais e não-funcionais (Seção 3.2), e os recursos de software (Seção 3.3) utilizados no desenvolvimento do sistema.

3.1 Escopo do Sistema

O sistema Dário tem como objetivo o desenvolvimento de um aplicativo para *iOS* e de uma aplicação web que conecta pessoas interessadas em realizar ações sociais voluntárias a eventos cadastrados por instituições. O sistema Dário é composto por dois componentes fundamentais: um aplicativo móvel dedicado aos voluntários para ser utilizado em *smartphones* com sistema operacional *iOS* e uma aplicação *web* destinada às instituições que poderá ser utilizada através de qualquer navegador *web*. Ambos desempenham papéis essenciais na realização de eventos solidários e na promoção de causas benéficas.

O aplicativo Dário é destinado aos voluntários e tem como objetivo fornecer uma plataforma de fácil acesso e interação. O escopo do aplicativo inclui:

- **Consulta de Eventos:** voluntários podem buscar e visualizar informações sobre eventos disponíveis, como datas, horários, locais e descrições;
- **Detalhes de Evento:** os voluntários podem acessar informações abrangentes sobre eventos específicos, incluindo instruções, requisitos de participação e detalhes de contato;
- **Inscrição em Eventos:** voluntários podem se inscrever em eventos que considerem relevantes, manifestando seu compromisso de participação;
- **Acompanhamento de Eventos:** voluntários podem acompanhar os eventos em que se inscreveram, recebendo atualizações e lembretes pertinentes;
- **Consulta de Instituições:** voluntários têm acesso a detalhes das instituições participantes, incluindo suas missões, projetos e necessidades atuais;
- **Detalhes de Instituição:** os voluntários podem obter informações aprofundadas sobre instituições específicas, incluindo seu histórico, projetos atuais e necessidades;
- **Consulta de Doações:** os voluntários podem verificar informações sobre as doações necessárias, incluindo tipos, quantidades e prazos;
- **Cadastro de Usuário:** os voluntários têm a possibilidade de criar contas pessoais no aplicativo, fornecendo informações de perfil e preferências;
- **Login de Usuário:** a autenticação segura permite que os voluntários acessem suas contas com segurança;
- **Calibração de Preferências:** voluntários podem personalizar suas preferências de voluntariado, adaptando as recomendações de eventos e instituições ao seu perfil.

A aplicação web é direcionada às instituições e projetada para fornecer uma maneira eficaz de gerenciar eventos e monitorar a participação dos voluntários. O escopo da aplicação web inclui:

- **Cadastro de Eventos:** as instituições podem cadastrar eventos, inserindo informações detalhadas, como datas, horários, locais, descrições e requisitos de voluntários;
- **Acompanhamento de Eventos:** instituições podem monitorar o progresso dos eventos e a participação dos voluntários, facilitando a gestão de recursos e a tomada de decisões;
- **Validação de Presença dos Voluntários:** as instituições têm a capacidade de registrar a presença dos voluntários nos eventos, garantindo transparência e confiabilidade;
- **Cadastro de Instituições:** as instituições podem criar perfis personalizados para apresentar sua missão, projetos em andamento e necessidades específicas;
- **Login de Instituição:** o acesso seguro permite que as instituições gerenciem eventos e interajam com voluntários com eficácia.

O escopo do sistema Dário não inclui o desenvolvimento de funcionalidades que não estejam diretamente relacionadas à conexão entre voluntários e instituições para a realização de ações sociais voluntárias.

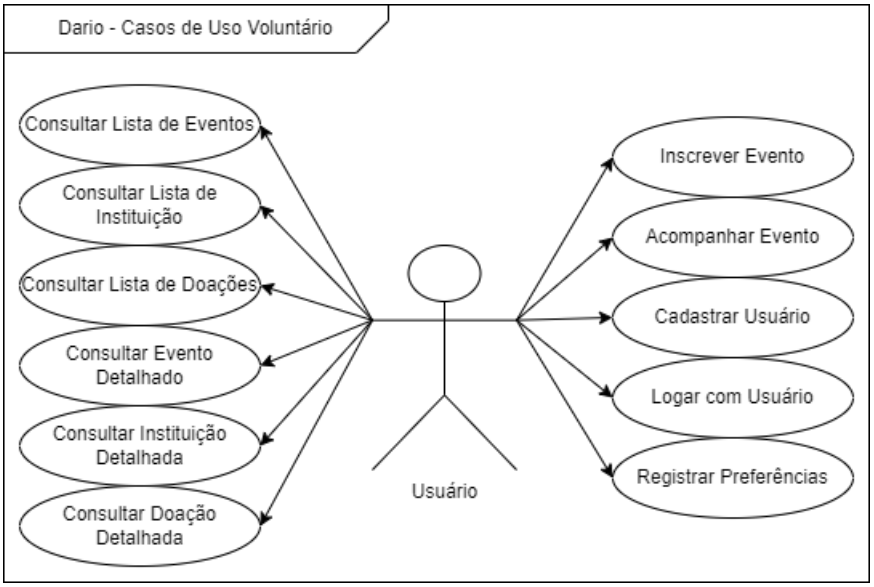
3.2 Modelagem do Sistema

Nesta seção são apresentados os casos de uso do sistema e seus respectivos fluxos, além da descrição dos requisitos funcionais e não funcionais.

3.2.1 Casos de Uso do Aplicativo Dário

A Figura 4 exibe o diagrama de casos de uso que representa as interações entre os atores voluntários e o aplicativo Dário.

Figura 4 – Diagrama de casos de uso - Aplicativo



Fonte: Autoria própria (2023).

Caso de Uso 1 - Consultar Lista de Eventos

- Ator Principal: voluntário;
- Descrição: o voluntário busca eventos disponíveis para participação;
- Pré-condições: o voluntário deve estar autenticado no aplicativo;
- Pós-condições: o voluntário visualiza uma lista de eventos disponíveis;
- Fluxo Básico:

    1. o voluntário inicia a sessão no aplicativo;
    2. o voluntário acessa a função de consulta de eventos;
    3. o sistema exibe uma lista de eventos disponíveis, incluindo datas, locais e descrições;
    4. o voluntário pode selecionar um evento para obter mais informações ou se inscrever.

Caso de Uso 2 - Consultar Lista de Instituição

- Ator Principal: voluntário;
- Descrição: o voluntário consulta a lista de instituições disponíveis;
- Pré-condições: o voluntário deve estar autenticado no aplicativo;
- Pós-condições: o voluntário visualiza uma lista de instituições disponíveis;

- Fluxo Básico:

    1. o voluntário inicia a sessão no aplicativo;
    2. o voluntário acessa a função de consulta de instituições;
    3. o sistema exibe uma lista de instituições disponíveis, incluindo informações sobre suas missões e necessidades;
    4. o voluntário pode selecionar uma instituição para obter mais detalhes.

Caso de Uso 3 - Consultar Lista de Doações

- Ator Principal: voluntário;
- Descrição: o voluntário consulta a lista de doações necessárias pelas instituições;
- Pré-condições: o voluntário deve estar autenticado no aplicativo;
- Pós-condições: o voluntário visualiza uma lista de doações necessárias;
- Fluxo Básico:

    1. o voluntário inicia a sessão no aplicativo;
    2. o voluntário acessa a função de consulta de doações necessárias;
    3. o sistema exibe uma lista de doações necessárias, incluindo tipos de doações e quantidades;
    4. o voluntário pode escolher uma doação para obter mais informações.

Caso de Uso 4 - Consultar Evento Detalhado

- Ator Principal: voluntário;
- Descrição: o voluntário obtém informações detalhadas sobre um evento específico;
- Pré-condições: o voluntário deve estar autenticado no aplicativo;
- Pós-condições:o voluntário visualiza detalhes completos do evento;
- Fluxo Básico:

    1. o voluntário inicia a sessão no aplicativo;
    2. o voluntário consulta a lista de eventos disponíveis;
    3. o voluntário seleciona um evento específico de interesse;
    4. o sistema exibe informações detalhadas sobre o evento, incluindo descrição, requisitos de participação e detalhes de contato.

Caso de Uso 5 - Consultar Instituição Detalhada

- Ator Principal: voluntário;
- Descrição: o voluntário obtém informações detalhadas sobre uma instituição específica;
- Pré-condições: o voluntário deve estar autenticado no aplicativo;
- Pós-condições: o voluntário visualiza detalhes completos da instituição;
- Fluxo Básico:
    1. o voluntário inicia a sessão no aplicativo;
    2. o voluntário consulta a lista de instituições disponíveis;
    3. o voluntário seleciona uma instituição específica de interesse;
    4. o sistema exibe informações detalhadas sobre a instituição, incluindo sua missão, projetos atuais e necessidades.

Caso de Uso 6 - Consultar Doação Detalhada

- Ator Principal: voluntário;
- Descrição: o voluntário obtém informações detalhadas sobre uma doação específica necessária;
- Pré-condições: o voluntário deve estar autenticado no aplicativo;
- Pós-condições:o voluntário visualiza detalhes completos da doação necessária;
- Fluxo Básico:
    1. o voluntário inicia a sessão no aplicativo;
    2. o voluntário consulta a lista de doações necessárias;
    3. o voluntário seleciona uma doação específica de interesse;
    4. o sistema exibe informações detalhadas sobre a doação, incluindo tipo de doação, quantidade necessária e prazo.

Caso de Uso 7 - Inscrever Evento

- Ator Principal: voluntário;
- Descrição: o voluntário se inscreve em um evento de sua escolha;
- Pré-condições: o voluntário deve estar autenticado no aplicativo;

- Pós-condições: o voluntário é registrado para o evento selecionado;
- Fluxo Básico:

    1. o voluntário inicia a sessão no aplicativo;
    2. o voluntário consulta a lista de eventos disponíveis;
    3. o voluntário seleciona um evento de interesse;
    4. o voluntário escolhe a opção “Inscrição” para o evento;
    5. o sistema registra o voluntário como participante do evento.

Caso de Uso 8 - Acompanhar Evento

- Ator Principal: voluntário;
- Descrição: o voluntário acompanha um evento em que está inscrito;
- Pré-condições: o voluntário deve estar autenticado no aplicativo e inscrito no evento;
- Pós-condições: o voluntário pode acompanhar o andamento do evento;
- Fluxo Básico:

    1. o voluntário inicia a sessão no aplicativo;
    2. o voluntário acessa a lista de eventos em que está inscrito.
    3. o voluntário seleciona um evento;
    4. o sistema exibe informações sobre o evento, como datas, localização e status atual;
    5. o voluntário pode acompanhar atualizações e notificações relacionadas ao evento.

Caso de Uso 9 - Cadastrar Usuário

- Ator Principal: voluntário;
- Descrição: o voluntário cria uma conta no aplicativo;
- Pré-condições: nenhuma;
- Pós-condições: o voluntário tem uma conta ativa no aplicativo;
- Fluxo Básico:

    1. o voluntário inicia o aplicativo pela primeira vez;
    2. o voluntário seleciona a opção “Cadastro de Usuário”;

3. o voluntário fornece informações pessoais, como nome, e-mail e senha;
4. o sistema verifica as informações e cria a conta do voluntário;
5. o voluntário pode fazer o login após o cadastro ser concluído.

Caso de Uso 10 - Logar com Usuário

- Ator Principal: voluntário;
- Descrição: o voluntário inicia sessão em sua conta no aplicativo;
- Pré-condições: o voluntário deve ter uma conta ativa no aplicativo;
- Pós-condições: o voluntário tem acesso aos recursos do aplicativo após o login;
- Fluxo Básico:

    1. o voluntário abre o aplicativo;
    2. o voluntário seleciona a opção “Login de Usuário”;
    3. o voluntário insere seu e-mail e senha;
    4. o sistema verifica as credenciais do voluntário;
    5. o voluntário é autenticado e tem acesso aos recursos do aplicativo.

Caso de Uso 11 - Registrar Preferências

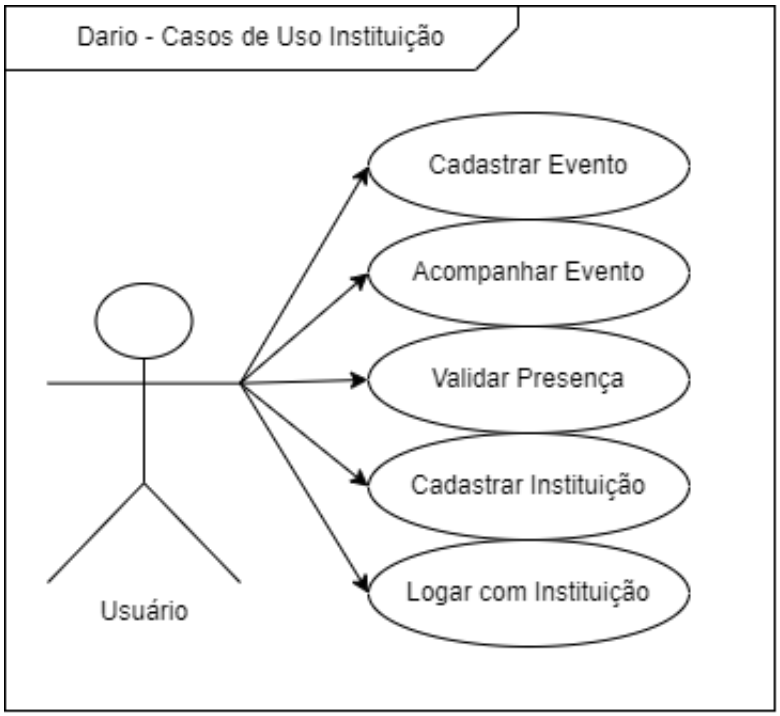
- Ator Principal: voluntário;
- Descrição: o voluntário personaliza suas preferências no aplicativo;
- Pré-condições: o voluntário deve estar autenticado no aplicativo;
- Pós-condições: as preferências do voluntário são atualizadas no sistema;
- Fluxo Básico:

    1. o voluntário inicia sessão no aplicativo;
    2. o voluntário acessa a opção “Calibração de Preferências” no menu;
    3. o voluntário configura suas preferências, como tipos de eventos preferidos, localizações e notificações;
    4. o sistema armazena as preferências do voluntário;
    5. o aplicativo personaliza as recomendações e notificações com base nas preferências do voluntário.

3.2.2 Casos de Uso da Aplicação Web

A Figura 5 exibe o diagrama de casos de uso que representa as interações entre os atores Instituição e a aplicação web.

Figura 5 – Diagrama de casos de uso - Aplicação Web



Fonte: Autoria própria (2023).

Caso de Uso 1 - Cadastrar Evento

- Ator Principal: instituição;
- Descrição: a instituição cadastra um novo evento em sua conta;
- Pré-condições: a instituição deve estar autenticada na plataforma;
- Pós-condições: o novo evento é adicionado à lista de eventos da instituição;
- Fluxo Básico:

    1. a instituição inicia sessão na plataforma;
    2. a instituição acessa a função de cadastro de eventos;
    3. a instituição fornece informações sobre o evento, como data, hora, local e descrição;
    4. o sistema adiciona o novo evento à lista da instituição.

Caso de Uso 2 - Acompanhar Evento

- Ator Principal: instituição;
- Descrição: a instituição acompanha o andamento de um evento que ela criou;
- Pré-condições: a instituição deve estar autenticada na plataforma e deve ter criado o evento;
- Pós-condições: a instituição tem acesso às informações e atualizações do evento;
- Fluxo Básico:
    1. a instituição faz login na plataforma;
    2. a instituição acessa a lista de eventos que ela criou;
    3. a instituição seleciona um evento específico;
    4. o sistema exibe informações detalhadas sobre o evento, incluindo participantes, status atual e *feedback* dos voluntários.

Caso de Uso 3 - Validar Presença

- Ator Principal: instituição;
- Descrição: a instituição registra a presença de voluntários em um evento;
- Pré-condições: o evento deve estar em andamento e a instituição deve estar autenticada na plataforma;
- Pós-condições: a presença dos voluntários é registrada no sistema;
- Fluxo Básico:
    1. a instituição inicia sessão na plataforma;
    2. a instituição acessa a função de validação de presença;
    3. a instituição seleciona um evento em andamento;
    4. a instituição verifica a presença dos voluntários que compareceram ao evento;
    5. o sistema registra as presenças dos voluntários.

Caso de Uso 4 - Cadastrar Instituição

- Ator Principal: instituição;
- Descrição: a instituição cria uma conta na plataforma;
- Pré-condições: nenhuma;

- Pós-condições: a instituição tem uma conta ativa na plataforma;
- Fluxo Básico:
    1. a instituição acessa a página de cadastro na plataforma;
    2. a instituição fornece informações sobre a instituição, como nome, missão, e-mail e senha;
    3. o sistema verifica as informações e cria a conta da instituição;
    4. a instituição pode fazer login após o cadastro ser concluído.

Caso de Uso 5 - Logar com Instituição

- Ator Principal: instituição;
- Descrição: a instituição inicia sessão em sua conta na plataforma;
- Pré-condições: a instituição deve ter uma conta ativa na plataforma;
- Pós-condições: a instituição tem acesso aos recursos da plataforma após o login;
- Fluxo Básico:
    1. a instituição acessa a página de login na plataforma;
    2. a instituição insere seu e-mail e senha;
    3. o sistema verifica as credenciais da instituição;
    4. a instituição é autenticada e tem acesso aos recursos da plataforma.

3.2.3 Requisitos Funcionais

Para o aplicativo, foram levantados os seguinte requisitos funcionais:

- **RF-01:** o sistema deve conter uma lista de eventos solidários para serem utilizados nas recomendações contextuais;
- **RF-02:** o sistema deve conter uma lista de instituições para serem utilizados nas recomendações contextuais;
- **RF-03:** o sistema deve conter uma lista de possíveis doações para serem utilizados nas recomendações contextuais;
- **RF-04:** o sistema deve permitir que o voluntário selecione um evento solidário;
- **RF-05:** o sistema deve permitir que o voluntário selecione uma instituição;

- **RF-06:** o sistema deve permitir que o voluntário selecione uma doação necessária;
- **RF-07:** o sistema deve permitir a visualização dos detalhes de cada evento solidário;
- **RF-08:** o sistema deve permitir a visualização dos detalhes de cada instituição;
- **RF-09:** o sistema deve permitir a visualização dos detalhes de cada doação necessária;
- **RF-10:** o sistema deve permitir que o voluntário se inscreva em eventos solidários;
- **RF-11:** o sistema deve permitir a calibração de preferências pelo voluntário para melhorar as recomendações;
- **RF-12:** o sistema deve apresentar eventos solidários com base nas preferências do voluntário;
- **RF-13:** o sistema deve apresentar instituições com base nas preferências do voluntário;
- **RF-14:** o sistema deve apresentar doações necessárias com base nas preferências do voluntário;
- **RF-15:** o sistema deve apresentar eventos solidários com base nos dados de utilização quando o voluntário não tiver dados suficientes para calibrar o algoritmo;
- **RF-16:** o sistema deve apresentar instituições com base nos dados de utilização quando o voluntário não tiver dados suficientes para calibrar o algoritmo;
- **RF-17:** o sistema deve apresentar doações necessárias com base nos dados de utilização quando o voluntário não tiver dados suficientes para calibrar o algoritmo;
- **RF-18:** o sistema deve utilizar os dados de utilização para otimizar as recomendações.

Para a aplicação web, foram levantados os seguinte requisitos funcionais:

- **RF-19:** o sistema deve permitir que as instituições criem contas na plataforma;
- **RF-20:** o sistema deve permitir que as instituições façam login em suas contas na plataforma;
- **RF-21:** o sistema deve permitir que as instituições criem eventos;
- **RF-22:** o sistema deve permitir que as instituições acompanhem o andamento de eventos que criaram;
- **RF-23:** o sistema deve permitir que as instituições validem a presença dos voluntários em seus eventos;

3.2.4 Requisitos Não Funcionais

Para os sistemas em geral, foram levantados os seguintes requisitos funcionais:

- **RNF-01: compatibilidade com Dispositivos:** o aplicativo deve ser executado em dispositivos *iPhone* com *ioS* 17.0 ou superior;
- **RNF-02: conectividade:** o aplicativo deve utilizar uma conexão à internet para realizar suas funções;
- **RNF-03: linguagem de Programação do Aplicativo:** o aplicativo deve ser escrito na linguagem de programação nativa *Swift* 5.0;
- **RNF-04: linguagem de Programação do Sistema de *Backend*:** o sistema de *backend*, que inclui o sistema de recomendação, deve ser escrito na linguagem de programação *Swift* 5.0, fazendo uso da biblioteca *NodeJS*;
- **RNF-05: linguagem de Programação da Plataforma Institucional:** a plataforma institucional deve ser escrita na linguagem de programação *Python* com o uso do *framework Flask*;
- **RNF-06: hospedagem do Sistema de *Backend*:** o sistema de *backend* deve ser hospedado em serviços *Microsoft Azure*;
- **RNF-07: hospedagem do Sistema de Dados:** o sistema de dados deve ser hospedado em serviços *Microsoft Azure*;
- **RNF-08: hospedagem da Plataforma Institucional:** a plataforma institucional deve ser hospedada em serviços *Microsoft Azure*.

3.3 Recursos de *Software*

Esta seção descreve os principais recursos de *software* utilizados no desenvolvimento do sistema Dário. O sistema foi projetado para atender às necessidades de voluntários e instituições, oferecendo um ambiente interativo e eficiente para que haja colaboração em eventos solidários ou para encontrar meios de realizar doações.

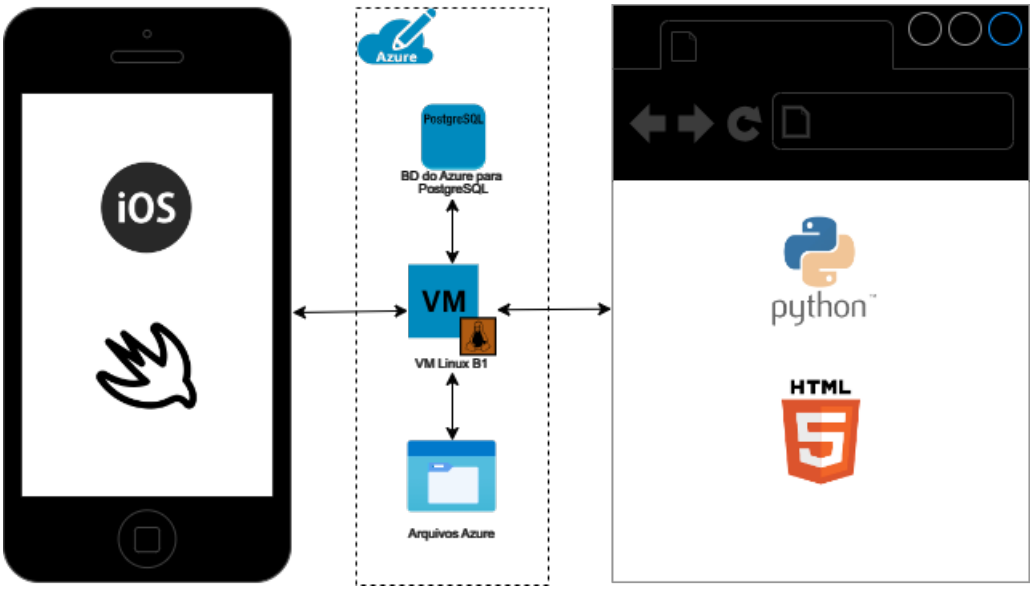
A Figura 6 apresenta uma visualização que representa as interações entre os componentes do sistema. Esta demonstra a comunicação bidirecional entre o aplicativo móvel e o *backend*, além da aplicação *web* e o *backend* também, este que é hospedado na plataforma *Azure*.

O sistema é composto por várias partes essenciais. Primeiramente, temos as máquinas virtuais que abrigam os componentes do sistema. Para o aplicativo móvel, o desenvolvimento foi

feito principalmente em *Swift* (ver Seção 3.3.2.2), enquanto a aplicação *web* foi construída utilizando *Python* (ver Seção 3.3.3.1) e *Linguagem de Marcação de Hipertexto, do inglês HyperText Markup Language (HTML)* (ver Seção 3.3.3.3).

Além disso, o sistema utiliza um banco de dados hospedado na plataforma *Azure*, empregando o *PostgreSQL* (ver Seção 3.3.1) para a eficaz armazenagem de dados. Também conta com um sistema de arquivamento na plataforma *Azure*, onde são armazenadas as imagens necessárias para o funcionamento do sistema.

Figura 6 – Diagrama de blocos de todo sistema Dário



Fonte: Autoria própria (2023).

A seguir é apresentada uma visão geral dos componentes do sistema.

3.3.1 Banco de Dados PostgreSQL

O *PostgreSQL* é um poderoso sistema de banco de dados objeto-relacional de código aberto que usa e estende a linguagem Linguagem de consulta estruturada, do inglês *Structured Query Language* (SQL) combinada com muitos recursos que armazenam e dimensionam com segurança as cargas de trabalho de dados mais complicadas (POSTGRESQL, 2023b). As origens do *PostgreSQL* remontam a 1986 como parte do projeto *Postgres* da Universidade da Califórnia em Berkeley e tem mais de 35 anos de desenvolvimento ativo na plataforma principal.

O *PostgreSQL* conquistou uma forte reputação por sua arquitetura comprovada, confiabilidade, integridade de dados, conjunto robusto de recursos, extensibilidade e a dedicação da comunidade de código aberto por trás do *software* para fornecer consistentemente soluções inovadoras e de alto desempenho. O *PostgreSQL* é executado em todos os principais sistemas operacionais.

O *PostgreSQL* vem com muitos recursos destinados a ajudar os desenvolvedores a criar aplicativos, os administradores a proteger a integridade dos dados e a criar ambientes tolerantes a falhas, além de ajudá-lo a gerenciar seus dados, independentemente do tamanho do conjunto de dados (POSTGRESQL, 2023b). Além de ser gratuito e de código aberto, o *PostgreSQL* é altamente extensível.

O *PostgreSQL* tenta estar em conformidade com o padrão SQL onde tal conformidade não contradiz os recursos tradicionais ou pode levar a decisões arquitetônicas ruins. Muitos dos recursos exigidos pelo padrão SQL são suportados, embora às vezes com sintaxe ou função ligeiramente diferente.

As capacidades máximas do *PostgreSQL* deve ser levado em consideração antes da sua adotação, sendo (POSTGRESQL, 2023a):

- Tamanho máximo do banco de dados: ilimitado;
- Tamanho máximo de uma tabela: 32 TB;
- Tamanho máximo de uma linha: 1,6 TB;
- Tamanho máximo de um campo: 1 GB;
- Quantidade máxima de linhas por tabela: ilimitado;
- Quantidade máxima de colunas por tabela: 250 - 1600, dependendo do tipo de coluna;
- Quantidade máxima de índices por tabela: ilimitado.

O *PostgreSQL* utiliza o modelo cliente-servidor. Uma sessão do *PostgreSQL* consiste nos seguintes processos (programas) cooperando entre si (POSTGRESQL, 2023a):

- Um processo servidor, que gerencia os arquivos de banco de dados, recebe conexões dos aplicativos cliente e executa ações no banco de dados em nome dos clientes. O programa servidor de banco de dados se chama *postmaster*;
- O aplicativo cliente do usuário (*frontend*) que deseja executar operações de banco de dados. Os aplicativos cliente podem ter naturezas diversas: o cliente pode ser um aplicativo gráfico, um servidor *Web* que acessa o banco de dados para mostrar páginas *Web*, ou uma ferramenta especializada para manutenção do banco de dados.

O cliente e o servidor podem estar em hospedeiros diferentes. Neste caso, eles se comunicam através de uma conexão de rede *TCP/IP*. O servidor *PostgreSQL* pode tratar várias conexões simultâneas de clientes. Para esta finalidade, inicia-se um novo processo (*fork*) para cada conexão. Desde ponto em diante, o cliente e o novo processo servidor se comunicam sem intervenção do processo *postmaster* original. Portanto, o *postmaster* está sempre executando e

aguardando por novas conexões dos clientes, enquanto os clientes e seus processos servidores associados iteram em seu ciclo de vida da aplicação.

O padrão SQL não dá nenhuma garantia sobre a ordem das linhas na tabela. Quando a tabela é lida, as linhas aparecem em uma ordem aleatória, a não ser que a classificação seja requisitada explicitamente. Além disso, o SQL não atribui identificadores únicos para as linhas e, portanto, é possível existirem várias linhas totalmente idênticas na tabela. Isto é uma consequência do modelo matemático subjacente ao SQL, mas geralmente não é desejável.

Cada coluna possui um tipo de dado. O tipo de dado restringe o conjunto de valores que podem ser atribuídos à coluna e atribui semântica aos dados armazenados na coluna, de forma que estes possam ser processados.

O *PostgreSQL* possui um extenso conjunto de tipos de dados nativos, adequados para muitos aplicativos. Os usuários também podem definir seus próprios tipos de dado.

As tabelas são objetos centrais da estrutura de um banco de dados relacional, porque armazenam os dados, mas não são os únicos objetos que existem no banco de dados. Podem ser criados vários objetos de outros tipos, para tornar o uso e o gerenciamento dos dados mais eficiente ou mais conveniente, tais como (POSTGRESQL, 2023a):

- *Views*;
- Funções e operadores;
- Tipos de dado e domínio;
- Gatilhos e regras de reescrita.

3.3.1.1 Linguagem PL/pgSQL

A linguagem *PL/pgSQL* é uma linguagem procedural carregável desenvolvida para o sistema de banco de dados *PostgreSQL*. Os objetivos de projeto da linguagem *PL/pgSQL* foram no sentido de criar uma linguagem procedural carregável que pudesse (POSTGRESQL, 2023a):

- Ser utilizada para criar procedimentos de funções e de gatilhos;
- Adicionar estruturas de controle à linguagem SQL;
- Realizar processamentos complexos;
- Herdar todos os tipos de dado, funções e operadores definidos pelo usuário;
- Ser definida como confiável pelo servidor;
- Ser fácil de utilizar.

O tratador de chamadas da linguagem *PL/pgSQL* analisa o texto do código fonte da função e produz uma árvore de instrução binária interna, na primeira vez em que a função é chamada (em cada sessão). A árvore de instruções traduz inteiramente a estrutura da declaração *PL/pgSQL*, mas as expressões SQL individuais e os comandos SQL utilizados na função não são traduzidos imediatamente.

Assim que cada expressão ou comando SQL é utilizado pela primeira vez na função, o interpretador do *PL/pgSQL* cria um plano de execução preparado. As execuções posteriores da expressão ou do comando reutilizam o plano preparado. Por isso, uma função com código condicional, contendo muitas declarações que podem requerer um plano de execução, somente prepara e salva os planos realmente utilizados durante o espaço de tempo da conexão com o banco de dados. Isso pode reduzir muito a quantidade total de tempo necessário para analisar e gerar os planos de execução para as declarações na função *PL/pgSQL*. A desvantagem é que erros em uma determinada expressão ou comando podem não ser detectados até que parte da função onde se encontram seja executada.

Uma vez que o *PL/pgSQL* tenha construído um plano de execução para um determinado comando da função, este plano será reutilizado enquanto durar a conexão com o banco de dados. Normalmente há um ganho de desempenho, mas pode causar problemas se o esquema do banco de dados for modificado dinamicamente.

A linguagem SQL é a que o *PostgreSQL* utiliza como linguagem de comandos. É portável e fácil de ser aprendida. Entretanto, todas as declarações SQL devem ser executadas individualmente pelo servidor de banco de dados. Isso significa que o aplicativo cliente deve enviar o comando para o servidor de banco de dados, aguardar que seja processado, receber os resultados, realizar algum processamento, e enviar o próximo comando para o servidor. Tudo isto envolve comunicação entre processos e pode, também, envolver tráfego na rede se o cliente não estiver na mesma máquina onde se encontra o servidor de banco de dados.

Usando a linguagem *PL/pgSQL*, pode-se agrupar um bloco de processamento e uma série de comandos dentro do servidor de banco de dados, juntando o poder da linguagem procedural com a facilidade de uso da linguagem SQL e economizando muito tempo, porque não há necessidade de sobrecarregar a comunicação entre o cliente e o servidor (POSTGRESQL, 2023a). Com isso, pode-se aumentar o desempenho consideravelmente.

Também podem ser utilizados na linguagem PL/pqSQL todos os tipos de dados, operadores e funções da linguagem SQL.

Para exemplificar o ganho de desempenho adquirido com a reutilização da árvore de instrução segue um descritivo do caminho que um comando realiza ao ser enviado para o servidor de banco de dados (POSTGRESQL, 2023a):

- Deve ser estabelecida uma conexão entre o aplicativo e o servidor *PostgreSQL*. O programa aplicativo transmite um comando para o servidor, e aguarda para receber de volta os resultados transmitidos pelo servidor;

- O estágio de análise verifica o comando transmitido pelo programa aplicativo com relação à correção da sintaxe, e cria a árvore de comando;
- O sistema de reescrita recebe a árvore de comando criada pelo estágio de análise, e procura por alguma regra (armazenada nos catálogos do sistema) a ser aplicada na árvore de comando. Depois realizam-se as transformações especificadas no corpo das regras;

  - Uma das aplicações do sistema de reescrita é a criação de *views*. Sempre que é executado um comando em uma *view* (ou seja, uma tabela virtual), o sistema de reescrita reescreve o comando do usuário acessando as tabelas base especificadas na definição da *view*, em vez da própria *view*.

- O planejador/otimizador recebe a árvore de comando (reescrita), e cria o plano de comando que será a entrada do executor;

  - Isto é feito criando todos os caminhos possíveis que levam ao mesmo resultado. Por exemplo, se existe um índice em uma relação a ser varrido, existem dois caminhos para a varredura. Uma possibilidade é uma varredura sequencial simples, e a outra possibilidade é utilizar o índice. Em seguida, estima-se o custo de execução de cada um dos caminhos, e escolhe-se o mais barato. O caminho mais barato é expandido em um plano completo para que o executor possa utilizá-lo.

- O executor caminha recursivamente através da árvore do plano, e traz as linhas no caminho representado pelo plano. O executor faz uso do sistema de armazenamento ao varrer as relações, realiza classificações e junções, avalia as qualificações e, por fim, envia de volta as linhas derivadas.

Todo esse processo fica validado ao utilizar funções em *PL/pgSQL* apenas aguardando a execução da função.

3.3.2 Tecnologias para Desenvolvimento do Aplicativo Dário

Esta seção descreve as tecnologias utilizadas para o desenvolvimento do aplicativo Dário proposto neste trabalho.

3.3.2.1 Sistema Operacional iOS

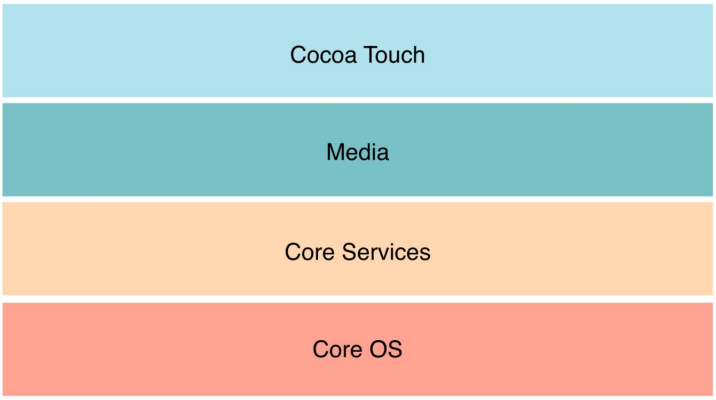
O sistema operacional *iOS*, lançado em 2007 juntamente com o primeiro *iPhone*, é o sistema operacional executado na linha de celulares da empresa. O sistema operacional gerencia o *hardware* do dispositivo e fornece as tecnologias necessárias para implementar aplicativos nativos (APPLE, 2023a).

O *iOS SDK* contém as ferramentas e interfaces necessárias para desenvolver, instalar, executar e testar aplicativos nativos que podem ser utilizados em dispositivos *iOS* (APPLE, 2023a). Aplicativos nativos são criados utilizando as *frameworks* do sistema *iOS* e a linguagem de programação *Swift*, desenvolvida também pela *Apple* para gradualmente substituir o uso de *Objective-C*, linguagem originalmente utilizada pela plataforma (APPLE, 2023b).

Em alto nível, o *iOS* atua como intermediário entre o *hardware* e os aplicativos. Os aplicativos raramente conversam diretamente com o *hardware*, em vez disso, os aplicativos se comunicam com este por meio de um conjunto de interfaces de sistema bem definidas, que protegem o aplicativo contra alterações de *hardware* (APPLE, 2023a).

A implementação das tecnologias *iOS* também pode ser vista através da representação de camadas, como pode ser observado na figura 7.

Figura 7 – Arquitetura do Sistema Operacional *iOS*



Fonte: Apple (2023a).

A *Apple* define as seguintes funcionalidades das camadas da arquitetura do sistema operacional *iOS* (APPLE, 2023a):

- ***Cocoa Touch*:** contém as principais estruturas para a criação de aplicativos *iOS*. Esta camada define a infraestrutura básica de aplicativos e suporte para tecnologias chave, como multitarefa, entrada baseada em toque, *push notification* e muitos serviços de sistema de alto nível;

- **Media:** contém as tecnologias de gráficos, áudio e vídeo voltadas para a criação de experiências multimídia. As tecnologias nessa camada foram projetadas para facilitar a criação de aplicativos com ótimos recursos visuais e sonoros;

- ***Core Services*:** contém os serviços fundamentais do sistema que todos os aplicativos usam, como localização, telefonia e *threads*. Ainda que alguns aplicativos não utilizem esses serviços diretamente, muitas partes do sistema são construídas sobre eles;

- ***Core OS*:** contém os recursos de baixo nível em que a maioria das outras tecnologias é desenvolvida. Ainda que muitos aplicativos não utilizem essa tecnologia diretamente,

elas provavelmente serão usadas por outros *frameworks*. Em situações que o aplicativo precisa lidar explicitamente com segurança ou se comunicar com um *hardware* externo acessório, isto é feito usando os *frameworks* desta camada.

A maioria das interfaces de sistema é entregue em pacotes especiais chamados *framework*. Um *framework* é um diretório que contém uma biblioteca compartilhada dinâmica e os recursos, como arquivos de cabeçalho, imagens, aplicativos auxiliares, entre outros, necessários para dar suporte a essa biblioteca (APPLE, 2023a).

O ambiente de desenvolvimento utilizado para criar, testar, depurar e ajustar os aplicativos é o *Xcode*, que é um invólucro para todas as outras ferramentas necessárias para criar os aplicativos (APPLE, 2023b).

Para executar aplicativos diretamente em dispositivos *iOS*, ao invés de simuladores, utilizar recursos avançados e para distribuir aplicativos na *App Store*, loja de aplicativos *iOS*, é necessário pagar por uma licença do *Apple Developer Program* (APPLE, 2023b).

3.3.2.2 Linguagem de Programação Swift

Swift é uma linguagem de programação moderna e poderosa, criada pela *Apple* para o desenvolvimento de aplicativos para *iOS, macOS, watchOS, ipadOS e tvOS* (APPLE, 2023b).

Desde 1990, a maioria dos desenvolvedores tem escrito aplicativos para a plataforma da *Apple* em *Objective-C*, uma linguagem construída sobre a linguagem de programação *C*. O *Objective-C* tem mais de 30 anos e a linguagem *C* existe há mais de 40 anos. Ambas as linguagens serviram bem a comunidade de desenvolvedores e não vão se extinguir em um futuro próximo. Entretanto, pode ser um desafio aprender a linguagem *Objective-C*. Como a tecnologia avançou muito rápido nos últimos anos, a *Apple* viu, então, a oportunidade de criar uma linguagem moderna, mais fácil de aprender e mais fácil de escrever e manter. Como isso em mente, surgiu a linguagem *Swift*.

A *Swift* possui várias características distintas que a tornam uma escolha popular entre os desenvolvedores, sendo (APPLE, 2023c):

- Sintaxe clara e concisa: a sintaxe da *Swift* é projetada para ser clara e fácil de ler, com ênfase na legibilidade do código. Ela utiliza uma abordagem mais simples e concisa em comparação com outras linguagens de programação, o que torna o código mais fácil de ser escrito e mantido;

- Inferência de tipos: a linguagem *Swift* possui um sistema de inferência de tipos avançados, o que significa que o desenvolvedor não precisa declarar explicitamente o tipo de uma variável ou constante se o compilador puder inferi-lo a partir do valor atribuído. Isso torna a escrita de código mais rápida e reduz a quantidade de código redundante;

- Segurança de tipos: *Swift* é uma linguagem fortemente tipada, o que significa que ela impõe regras rigorosas sobre como os tipos de variáveis são usados. Isso ajuda a prevenir erros comuns durantes o desenvolvimento, melhorando a segurança e a estabilidade do código;
- Gerenciamento automático de memória: a *Swift* utiliza o Contagem Automática de Referência, do inglês *Automatic Reference Counting* (ARC), um sistema de gerenciamento automático de memória que acompanha e gerencia automaticamente a alocação e desalocação de memória. Isso retira dos desenvolvedores a necessidade de lidar manualmente com a alocação e liberação de memória, reduzindo a ocorrência de vazamentos de memória;
- Suporte a opcionais: os opcionais são um recurso fundamental do *Swift* que permitem tratar valores que podem ser nulos de maneira segura. Isso ajuda a evitar erros de acesso a valores nulos, forçando o desenvolvedor a lidar com a possibilidade de um valor ausente de forma explícita;
- Padrões modernos de programação: a *Swift* incorpora muitos conceitos modernos de programação, como programação orientada a objetos, programação funcional e programação reativa. Ela fornece recursos e estruturas que facilitam a adoção desses padrões, permitindo que os desenvolvedores escrevam códigos mais expressivos e modular;
- Suporte à interoperabilidade com *Objective-C*: a linguagem *Swift* foi projetada para interoperar perfeitamente com a linguagem *Objective-C*, permitindo que os desenvolvedores usem códigos existente em *Objective-C* em projetos *Swift* e vice-versa. Isso é particularmente útil durante a transição gradual de aplicativos legados para *Swift*.

Algoritmos desenvolvidos com a linguagem *Swift* são rápidos e dinâmicos. Um algoritmo comum de busca, por exemplo, obtém o resultado muito mais rápido com *Swift*: até 2,6x mais rápido que *Objective-C* e até 8,4x mais rápido que *Python* 2.7 (APPLE, 2023b).

A linguagem *Swift* também possui uma ferramenta para gerenciar a distribuição do código *Swift* chamada *Swift Package Manager* (APPLE, 2023b). Ela é integrada ao sistema de compilação *Swift* para automatizar o processo de *download*, compilação e vinculação de dependências. A linguagem *Swift* organiza o código em módulos, onde cada módulo especifica um *namespace* e impõe controles de acesso a partes especificas do código que podem ser utilizadas fora do módulo (APPLE, 2023b).

Um programa pode ter todo o seu código em um único módulo ou pode importar outros módulos como dependências. Quando um módulo separado é utilizado para resolver um problema específico, esse código pode ser reutilizado em outras situações. A utilização de módulos permite a construção de soluções com base no código de outros desenvolvedores, em vez de reimplementar a mesma funcionalidade.

Em Dezembro de 2015, a linguagem *Swift*, bibliotecas suplementares, depurador e gerenciador de pacotes, tornaram-se *open source* sob a licença *Apache 2.0* com uma *Runtime Library Exception*, e o site *swift.org* foi criado para hospedar o projeto. O código fonte é hospedado no *GitHub*.

3.3.2.3 *Framework Nodejs*

O *Node.JS* é na realidade um *framework JavaScript* capaz de fazer com que a linguagem *JavaScript* possa ser também executada e interpretada do lado do servidor em sistemas de arquitetura cliente-servidor (NODEJS, 2023).

O *Node.JS* é construído sob uma arquitetura baseada em eventos e capaz de trabalhar com requisições assíncronas não bloqueantes. Isso faz com que o sistema seja leve e bastante eficiente. Essas são soluções ideais para resolver problemas de tráfego intenso de usuários e requisições na rede em que o sistema esteja sendo executado (NODEJS, 2023).

Desenvolver sistemas concorrentes utilizando múltiplas *threads* é uma atividade custosa que exige muito recurso computacional. Independente de quão bem se trabalha com *threads* no servidor, o seu número tenderá a crescer, pois este modelo dedica uma *thread* separada a cada nova requisição. Desta forma, há um aumento significativo no consumo de memória e CPU, levando ao declínio do desempenho e a escalabilidade do sistema (WELSH *et al.*, 2000) (IYER, 2013).

Uma das partes fundamentais da arquitetura do *Node.JS* é o evento em *loop*, método que o *Javascript* utiliza para lidar com os eventos. O evento em *loop* permite que, ao invés do sistema operacional, o *Node.JS* gerencie a mudança entre as tarefas a serem executadas (CANTELON *et al.*, 2014).

O evento em *loop* se trata de um *loop* principal responsável por esperar e despachar eventos que tem seus recursos disponibilizados. Esses eventos são assinados na *thread* principal da aplicação passando funções como parâmetros, denominadas de *callbacks*, que são executadas quando esses eventos são disparados, levando consigo o resultado da operação *Input/Output* (ZELDOVICH *et al.*, 2003).

Levando em consideração que o modelo orientado a evento tende a ser robusto em relação à carga de trabalho, se a implementação dos eventos e o estado de empacotamento das tarefas forem eficientes, o pico de *throughput* (ou vazão) pode ser alto e não haverá degradação quando o número de conexões simultâneas aumentar (IYER, 2013).

Com 3,5 milhões de adeptos e um crescimento anual de cem por cento, o Node.JS se tornou a tecnologia de desenvolvimento mais significante e com maior crescimento nos últimos anos, tornando-se uma plataforma universal para o desenvolvimento de aplicações *web*, *real-time*, *mobile*, *desktop*, microsserviços e internet das coisas (NODEJS, 2011).

3.3.2.4 Biblioteca Angular

O Angular é uma biblioteca desenvolvida por engenheiros do *Google*, que decidiram por criá-lo com intuito de facilitar manutenção de alguns de seus serviços *Web*. Originalmente, o trabalho de 18 mil linhas de código foi reduzido para 1.5 mil linhas, representando aproximadamente 91% de redução de linhas de código (SESHADRI; GREEN, 2014).

Há quatro princípios básicos que fundamentam o Angular e que permitem aos desenvolvedores criar aplicações complexas e de grande porte de forma rápida e fácil (SESHADRI; GREEN, 2014):

- **Orientação a dados:**

  A principal funcionalidade do Angular - que pode fazer com que milhares de linhas de código *boilerplate* sejam evitadas - é o seu *data-binding*. Basta efetuar o *binding* dos dados no HTML e o Angular cuidará de fazer esses valores chegarem até a Interface do Usuário. O Angular oferece o *data-binding* bidirecional, garantindo que o controlador e a Interface do Usuário compartilhem o mesmo modelo, de modo que a atualizações de um deles, fará o outro ser atualizado automaticamente.

- **Declarativo:**

  Uma aplicação *Web single-page* é composta de vários trechos de HTML e de dados associados por meio de *JavaScript*. Com muita frequência, acaba-se com *templates* HTML que não oferecem nenhum indício daquilo em que se transformaram. Essencialmente, esse é o paradigma imperativo, em que dizemos exatamente a aplicação como realizar toda e qualquer ação. Isso é totalmente feito por meio de código *JavaScript*, e não no local em que o HTML realmente deve ser alterado. O HTML não reflete nenhuma parte desta lógica. O Angular, por sua vez, promove um paradigma declarativo, em que declara-se diretamente no HTML o que estamos tentando realizar. Isso é feito por meio de um recurso que o Angular chama de diretiva. Basicamente as diretivas estendem o vocabulário do HTML para ensinar-lhe novos truques.

- **Injeção de dependência:**

  O Angular é um dos poucos *frameworks JavaScript* com um sistema completo de injeção de dependência incluído. A injeção de dependência corresponde ao conceito de solicitar as dependências de um controlador ou de um serviço em particular, em vez de instanciá-las online por meio de um operador novo ou chamar explicitamente uma função. Isso é útil porque:

  - O controlador, o serviço ou a função que solicitar a dependência não precisa saber como construir suas dependências e pode percorrer toda a cadeia, por mais longa que seja;

  - As dependências são explícitas, de modo que podemos saber de imediato o que é necessário antes de começarmos a trabalhar com nossa porção de código;
  - É mais fácil de testar porque pode-se substituir dependências por *mocks* adequados aos testes.

  A injeção de dependências do Angular é usada em todas as partes, que incluem desde os controladores e serviços até os módulos e testes. Ela permite implementar um código modular e reutilizável facilmente e pode ser usada de modo simples e fácil conforme for necessário.

- **Extensível:**

  As diretivas representam a maneira de o Angular ensinar novos truques ao navegador e ao *HTML*, que variam desde lidar com cliques e condicionais até criar novas estruturas e novos estilos. O Angular expõe uma Interface de Programação de Aplicação, do inglês *Application Programming Interface* (API) para que qualquer desenvolvedor possa estender as diretivas existentes ou criar suas próprias.

A arquitetura de um projeto Angular é desenvolvida utilizando componentes em árvore. Os componentes da aplicação devem ter uma comunicação entre si, sendo organizados em *NgModules* que fornecem um contexto para os componentes serem compilados para determinado domínio, existindo pelo menos um módulo raiz (ANGULAR, 2023).

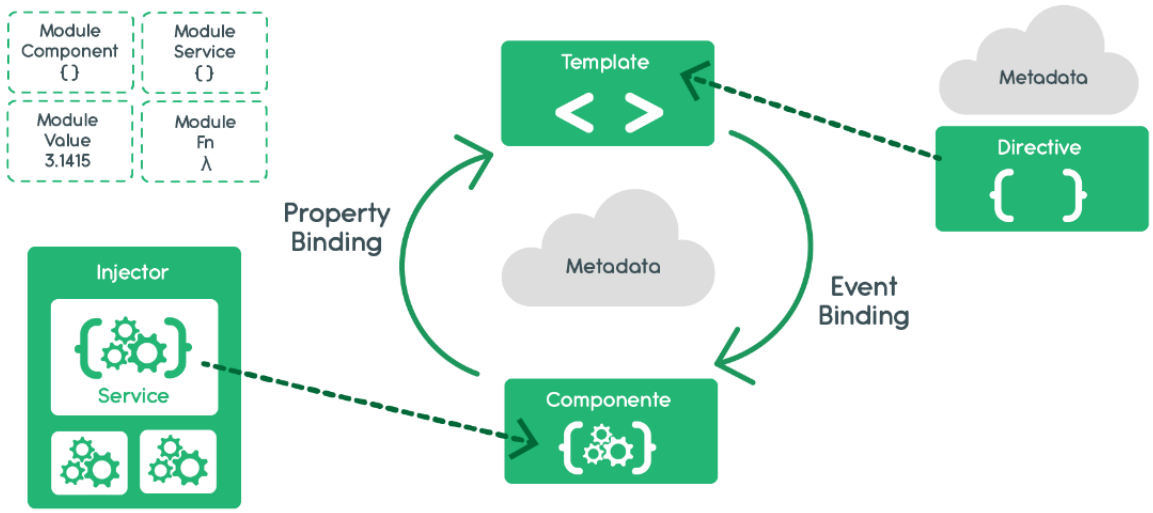
Como pode ser visto na Figura 8, os elementos básicos do Angular são:

- **Componente:** define uma classe, que está associada a um *template HTML* e os serviços para atribuir uma funcionalidade para seu módulo.
- ***Service*:** contém toda a regra de negócio da aplicação.
- **Diretivas:** adiciona as classes, definindo comportamentos específicos nos elementos DOM.
- **Módulo:** agrupa componentes, serviços e diretivas a um objetivo ou funcionalidade.

3.3.3 Tecnologias para Desenvolvimento da Aplicação *Web*

Uma aplicação web é um programa que é acessado e utilizado através de um navegador web, como o *Google Chrome*, *Microsoft Edge* ou *Safari*. Diferentemente de aplicativos tradicionais que precisam ser instalados em um dispositivo, uma aplicação web é hospedada e executada remotamente em servidores na internet, onde os clientes podem interagir com ela,

Figura 8 – Elementos Básicos de uma aplicação Angular



Fonte: Afonso (2018).

enviar e receber dados, realizar tarefas e acessar recursos diretamente a partir de um navegador.

As aplicações web podem variar em termos de funcionalidades e finalidades, podendo incluir aplicações de *e-mail*, como o *Gmail*, serviços de armazenamento em nuvem, como o *Google Drive*, redes sociais, como o *Facebook* e *Instagram*, e até mesmo comércio eletrônico, como diversos sites de lojas que podem ser encontrados online.

Para funcionar, uma aplicação web é construída usando uma combinação de tecnologias, incluindo *HTML*, *Folhas de estilo em cascata, do inglês Cascading Style Sheets (CSS)*, *JavaScript* que irão compor o *frontend*, além de um *backend* que geralmente inclui um servidor que gerencia os dados e a lógica da aplicação, podendo ser desenvolvido em diversas linguagens de programação, como por exemplo o *Python*.

As próximas seções descrevem as tecnologias utilizadas para o desenvolvimento da aplicação *web* proposto neste trabalho.

3.3.3.1 Linguagem de Programação Python

Python é uma linguagem de programação de alto nível utilizada em diversas áreas da computação, como desenvolvimento de *software*, análise de dados, aprendizado de máquina, entre várias outras aplicações. Desenvolvido por Guido van Rossum e lançada em 1991, *Python* é conhecida por sua simplicidade, clareza e facilidade de leitura de código, se tornando assim uma linguagem ideal para iniciantes em programação (RAMALHO, 2019).

As principais características que tornam essa linguagem mais popular são (PYTHON, 2023):

- **Legibilidade:** enfatiza a legibilidade do código, usando sintaxe clara e identação significativa, tornando o código mais organizado;
- **Biblioteca:** possui uma biblioteca padrão abrangente com diversos módulos para auxiliar em tarefas comuns;
- **Multiplataforma:** é executável em várias plataformas, tornando o programa portátil e intercambiável entre sistemas operacionais;
- **Versatilidade:** é uma linguagem que pode ser utilizada para desenvolver diferentes aplicações, como aplicativo web, aplicativo *desktop*, *script* de automação, jogos, entre outros;
- **Comunidade:** possui uma comunidade vasta e ativa, permitindo uma constante contribuição para criação de novas bibliotecas;
- **Aprendizado de Máquina e Ciência de Dados:** é uma das linguagens mais utilizadas para aprendizado de máquina, inteligência artificial e análise de dados;
- **Código Aberto:** é uma linguagem de código aberto e é distribuída gratuitamente.

Essas características tornam o *Python* uma linguagem fácil de ser utilizada e aplicável em diferentes campos, tornando-se assim uma escolha popular para diferentes níveis e tipos de programadores.

3.3.3.2 *Framework Flask*

Flask é um *microframework* web em *Python* que se destaca por sua simplicidade e facilidade de uso. Desenvolvido por Armin Ronacher, o *Flask* foi lançado em 2010 e se tornou uma escolha popular para criar aplicativos *web* em *Python*. É considerado “micro” para enfatizar a abordagem minimalista, fornecendo as ferramentas necessárias para se desenvolver uma aplicação *web* de forma eficaz e que é possível estender da forma que melhor se adapte ao projeto (FLASK, 2023).

Entre todas as características desse *framework* as que mais se destacam são (FLASK, 2023):

- **Roteamento Simples:** o *Flask* permite definir rotas facilmente, associando funções *Python* a Localizador Padrão de Recursos, do inglês *Uniform Resource Locator* (URL) específicas;
- **Extensibilidade:** embora o *Flask* seja definido como “micro”, ele é altamente extensível, extensões essas disponíveis para adicionar funcionalidades à aplicação;

- ***Template Engines*:** *flask* suporta sistemas que permitem a criação de aplicações dinâmicas, facilitando a renderização de páginas HTML que possuem comunicação com base de dados;

- **Desenvolvimento Rápido:** devido a ser um *framework* simples e de fácil uso, torna o desenvolvimento mais rápido de projetos considerados menores.

A arquitetura do *Flask* é baseada no modelo Controlador de visualização de modelo, do inglês *Model-View-Controller* (MVC), porém por se tratar de um *microframework*, ele fornece as ferramentas essenciais, porém não impõem uma estrutura rígida para o desenvolvimento do projeto (DWYER, 2016).

3.3.3.3 Linguagem de Marcação HTML

A história do *HTML* remonta à década de 1980, com Tim Berners-Lee propondo o sistema de hipertexto chamado *"WorldWideWeb"*, que incluía a primeira especificação do HTML, em 1989. Desde então, a *web* evoluiu, passando por diferentes versões de *HTML*, chagando ao mais atual, o *HTML5*, que introduziu diversos recursos avançados (HTML, 2023).

Em termos de estrutura, os sites *HTML* são geralmente estáticos, construídos com linguagem *HTML*, ocasionalmente incorporando *CSS* e *JavaScript* (HTML, 2023). Eles são ideais para exibir informações, como *blogs*, portfólios e páginas de produtos, com desenvolvimento simplificado em comparação com plataformas web mais complexas.

A arquitetura de um site *HTML* é fundamental para compreender como as diferentes partes interagem. Ela segue o modelo cliente-servidor, com o navegador atuando como cliente. O navegador interpreta o *HTML, CSS e JavaScript*, renderizando a página. O servidor *web*, onde os arquivos do site são armazenados, processa as solicitações *Protocolo de Transferência de Hipertexto, do inglês Hypertext Transfer Protocol (HTTP)*, interagindo, se necessário, com servidores de aplicação e bancos de dados.

O *HTML*, componente central da arquitetura, segue uma estrutura hierárquica. O elemento raiz *(<html>)* encapsula o cabeçalho *(<head>)* e o corpo *(<body>)*, este último contendo o conteúdo visível da página. Elementos estruturais, de texto e multimídia, juntamente com atributos, formam a base da marcação *HTML* (W3SCHOOLS, 2023).

Quando um usuário acessa um *site*, o processo inicia-se com a requisição do navegador. Ao digitar a URL ou clicar em um *link*, uma solicitação é enviada ao servidor web que hospeda o *site*, utilizando o método *GET*. O servidor processa essa solicitação, buscando o arquivo HTML correspondente à página desejada. Em seguida, a resposta do servidor, composta pelo código HTML que define a estrutura e o conteúdo da página, é enviada de volta ao navegador do usuário.

Além disso, quando o usuário interage com o *site*, por exemplo, preenchendo um formulário e enviando informações, ocorre uma nova solicitação ao servidor, desta vez utilizando o

método *POST*. Isso permite que dados do usuário sejam enviados ao servidor para processamento, contribuindo para a interatividade e dinamismo da página web.

O capítulo seguinte apresenta informações relacionadas ao processo de desenvolvimento do sistema, o aplicativo em si e a utilização do mesmo.

4 SISTEMA DÁRIO

O objetivo deste capítulo é apresentar o sistema Dário descrevendo inicialmente o sistema de recomendação utilizado e em seguida as interfaces gráficas do aplicativo móvel e da aplicação web.

4.1 Sistema de Recomendação

Nesta seção é apresentado o sistema de recomendação construído para este projeto.

O sistema é composto por uma função escrita na linguagem de programação *Swift* que é executada dentro de um serviço Máquina Virtual *Linux Azure* e acessado através de uma *API RESTful* criada também dentro do serviço Máquina Virtual *Linux Azure*. Para o armazenamento e leitura dos dados que alimentam o sistema, é utilizado o serviço Banco de dados do *Azure* para *PostgreSQL* e para armazenamento de imagens é utilizado o serviço Armazenamento de *Blob* do *Azure*.

Através de um *endpoint* configurado para esta *API*, o aplicativo Dário envia os dados dos usuários e obtém uma resposta contendo os dados dos eventos a serem recomendados.

4.1.1 Recebendo Preferências do Usuário

Para enviar ao sistema de recomendação as preferências do usuário dentre as opções disponíveis, o aplicativo Dário encapsula as preferências selecionadas em um objeto *JSON*, juntamente com um identificador único vinculado ao usuário, para a localização dos dados deste usuário no sistema. É realizada uma requisição *HTTP POST* para o *endpoint* "*./vol/preferencias", configurado na *API Restful*.

4.1.2 Processamento dos Dados de Entrada

As informações recebidas pela *API* são utilizadas como entrada para a função que executa o sistema de recomendação. Ele recupera o identificador de usuário contido no corpo da requisição e o procura dentre os usuários existentes no sistema, que estão armazenados na estrutura Banco de dados do *Azure* para *PostgreSQL*. Caso o usuário não exista, ele é criado no sistema e suas preferências são salvas, atribuindo uma classificação numérica de valor 1 a cada preferência selecionada por ele. Caso o usuário já exista no sistema, a classificação de cada preferência enviada na solicitação é atualizada, acrescentando-se o valor 1 à nota já existente para aquela opção e salvando o valor atualizado na base de dados.

A resposta obtida através da chamada da *API* descrita na subseção 4.1.1 também é em formato *JSON*, contendo um vetor que representa a lista de eventos recomendados a partir da

lista enviada. Essa lista é o resultado do processamento de funções matemáticas utilizadas no sistema de recomendação descrito no Capítulo 2.

Inicialmente, tem-se a matriz usuário x item, na qual as linhas representam os usuários, as colunas representam os eventos e o valor contido em cada intersecção linha x coluna é a classificação dada pelo usuário à atuação do evento, sendo que eventos não correspondentes às suas preferências são preenchidos com o valor 0. A partir dela é obtida a matriz de similaridade, utilizando o módulo *pgvector* da biblioteca do *PostgreSQL*. Esse módulo contém uma função para o cálculo de similaridade por cosseno, que é aplicado à matriz usuário x item.

Tendo essas informações, o sistema busca na matriz de similaridade, dentre todos os outros usuários, aqueles que sejam mais similares ao usuário em questão, ou seja, que possuam preferências parecidas. Na prática, ele busca na matriz os valores mais altos da linha deste usuário, que não seja ele mesmo. A quantidade k de usuários mais similares que deseja-se analisar é definida no código, sendo neste projeto definido como 1. Em seguida, obtém-se a média da classificação desse único usuário com as preferências mais semelhantes para cada evento, descartam-se os eventos já inscritos pelo usuário a quem será dada a recomendação, e retornam-se os n eventos de nota mais alta, cuja quantidade é definida no código, sendo neste projeto definido como 3. Ambos os valores k e n são valores fixos definidos na codificação do algoritmo de recomendação.

Por fim, a lista das recomendações obtidas pelo sistema é retornada em formato *JSON*, como exibido no código apresentado na Listagem 1

O código do sistema de recomendação pode ser consultado no link do Github do projeto: https://github.com/halvesbatochi/DarioAPI

Listagem 1 – Exemplo de retorno de recomendações

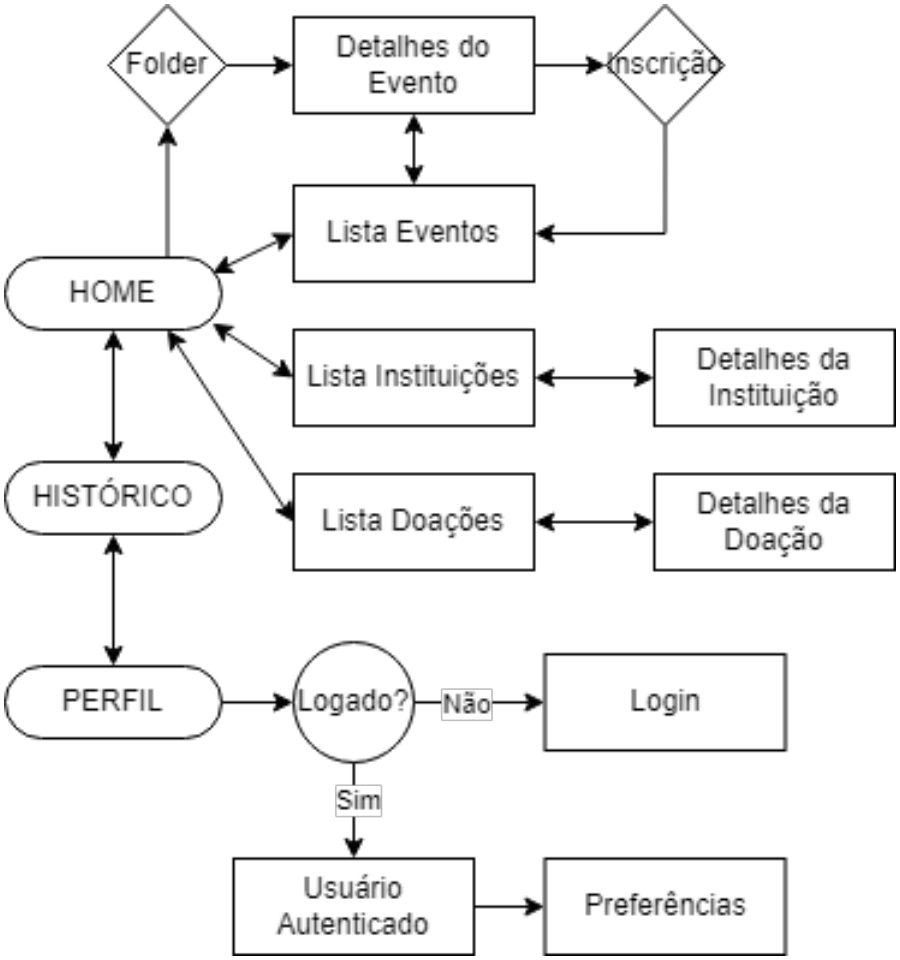
```
{
    "ev001_it_id" : 4,
    "ev001_it_inst" : 11,
    "ev001_vc_end" : "Rua Candido Lopes",
    "ev001_it_num" : 231,
    "ev001_vc_compl" : "Apto 91",
    "ev001_vc_bairro" : "Centro",
    "ev001_vc_cidade" : "Curitiba",
    "ev001_vc_cep" : "80020060",
    "ev001_vc_estado" : "PR",
    "ev001_vc_pais" : "BR",
    "ev001_vc_titulo" : "Show de Doações",
    "ev001_dt_inic" : 20231130,
    "ev001_hr_inic" : 900,
    "ev001_dt_fim" : 20231130,
    "ev001_hr_fim" : 930,
    "ev001_it_npart" : 20,
    "ev001_vc_fmsg1" : "Precisamos de você!",
    "ev001_vc_fmsg2" : "Convidados a todos com vontade de ajudar a construir
    bons momentos aos nossos residentes.",
    "ev001_vc_fmsg3" : "Unindo forças em áreas de interesse comum temos como
    meta contruir um evento solidário!",

    "ev001_vc_fmsg4" : "Escolha sua área de interesse em ajudar e venha
    construir a diferença em um mundo mais solidário. Juntando nossas
    diferenças em prol de algo comum construíremos um mundo melhor.",

    "ev001_vc_fmsg5" : "Agradecemos ao seu interesse em construir momentos
    melhores e com isso um mundo melhor! Agora só falta confirmar a sua
    participação!",

    "ev001_vc_pmsg1" : "Escolha sua área de interesse em ajudar e venha
    construir a diferença em um mundo mais solidário. Juntando nossas
    diferenças em prol de algo comum construíremos um mundo melhor.",

    "ev001_vc_pmsg2" : "Sua ajuda fará a diferença e construirá um
    momento único. Você poderá expandir suas amizades, desenvolver
    atividades sociais e solidárias e construir um mundo melhor.",

    "ev001_vc_img1" : "logofolderazureblobs.com",
    "ev001_vc_img2" : "logopaginaazureblobs.com",
    "ev001_it_atv1" : 5,
    "ev001_it_atv2" : 6,
    "ev001_it_atv3" : 7,
    "ev001_it_situac" : 1,
    "ev001_dt_ultatu" : "2023-11-14T04:46:32.857Z",
    "ev001_dt_inclus" : "2023-11-14T04:46:32.857Z"
}
```

Fonte: Autoria própria (2023).

4.2 O Aplicativo

Na Figura 9 é possível visualizar o fluxo de telas do aplicativo.

Figura 9 – Fluxo de Telas do Aplicativo



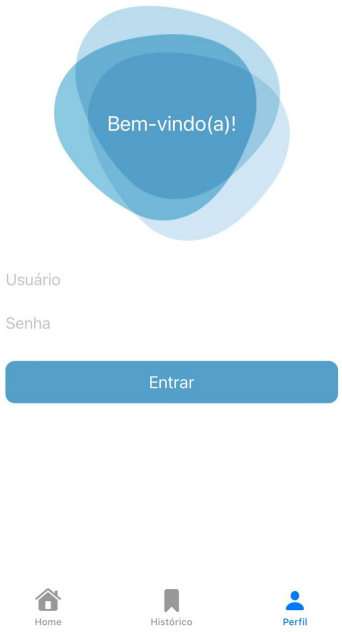
Fonte: Autoria Própria (2023).

A seguir são apresentadas as telas que compõem a interface gráfica do aplicativo móvel e a explicação de cada uma delas.

4.2.1 Tela de Perfil

Na Figura 10 pode-se ver a tela de login do usuário. No topo, ela apresenta a logo do aplicativo, além dos campos para inserção dos dados de usuário e senha e o botão “Entrar”. Na parte inferior da tela, existe um menu fixo com três opções: *Home*, Histórico e Perfil.

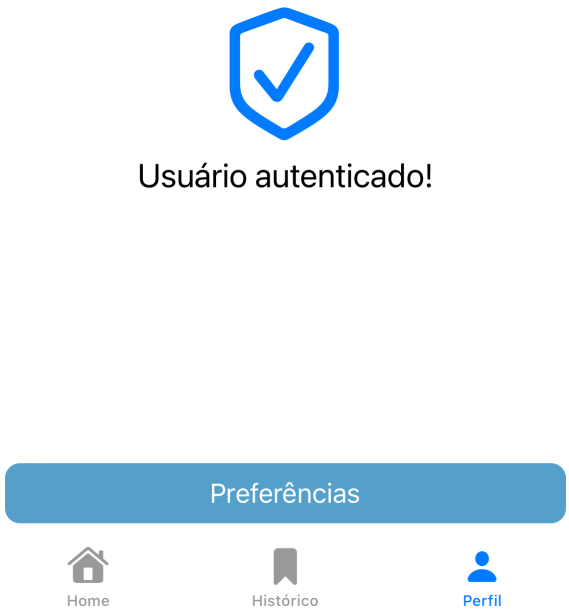
Figura 10 – Tela de *Login*



Fonte: Autoria Própria (2023).

Caso o usuário já esteja logado no sistema, ao clicar na opção “Perfil” o aplicativo mostrará uma mensagem de que o usuário está autenticado, conforme pode-se ver na Figura 11.

Figura 11 – Tela de Perfil



Fonte: Autoria Própria (2023).

4.2.2 Tela Preferências

A tela apresentada na Figura 12 aparece quando é selecionada a opção “Preferências” apresentada na Figura 11. Na tela Preferências são listadas todas as opções de áreas de interesse para que o usuário possa configurar suas preferências de eventos. No topo da tela apresenta-se uma frase de introdução e na parte principal aparecem algumas áreas de possível atuação nos eventos, listadas para que o usuário possa selecionar. Na parte inferior da tela, existe um menu fixo com três opções: *Home*, Histórico e Perfil.

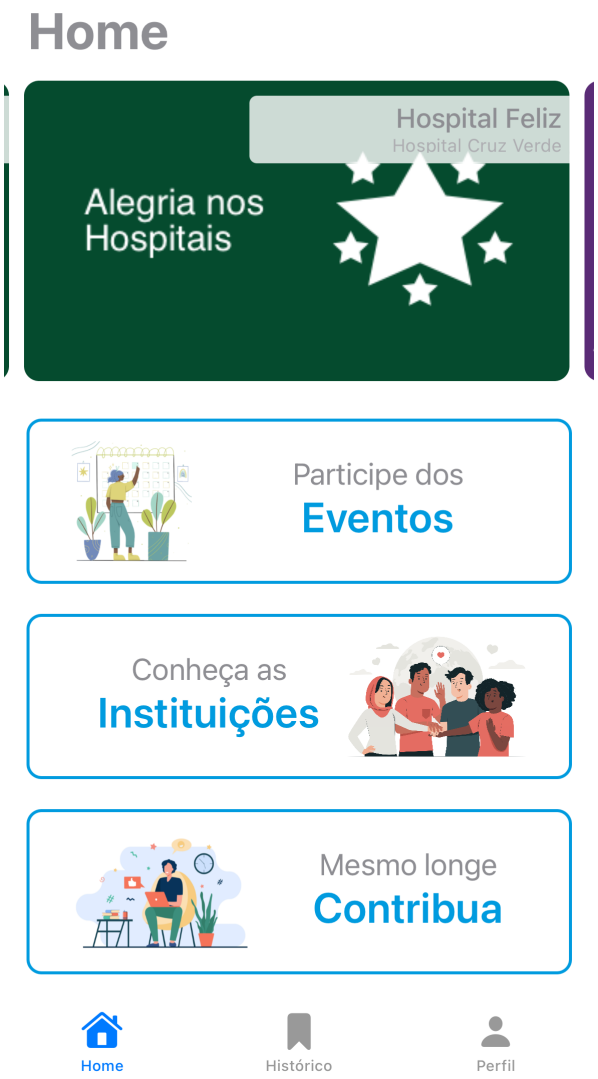
Figura 12 – Tela Preferências

Fonte: Autoria Própria (2023).

4.2.3 Tela *Home*

Na Figura 13 pode-se ver a tela *Home*, que é a tela inicial apresentada ao usuário. A parte principal da tela possui um carrossel de *folders*, que está descrito na Seção 4.2.4. É possível ver também três opções para seleção: “Participe dos eventos”, que está apresentada na Seção 4.2.5, “Conheça as Instituições”, que está apresentada na Seção 4.2.6 e “Mesmo longe Contribua”, que está apresentada na Seção 4.2.7. Na parte inferior da tela, existe um menu fixo com três opções: *Home*, Histórico e Perfil.

Figura 13 – Tela *Home*

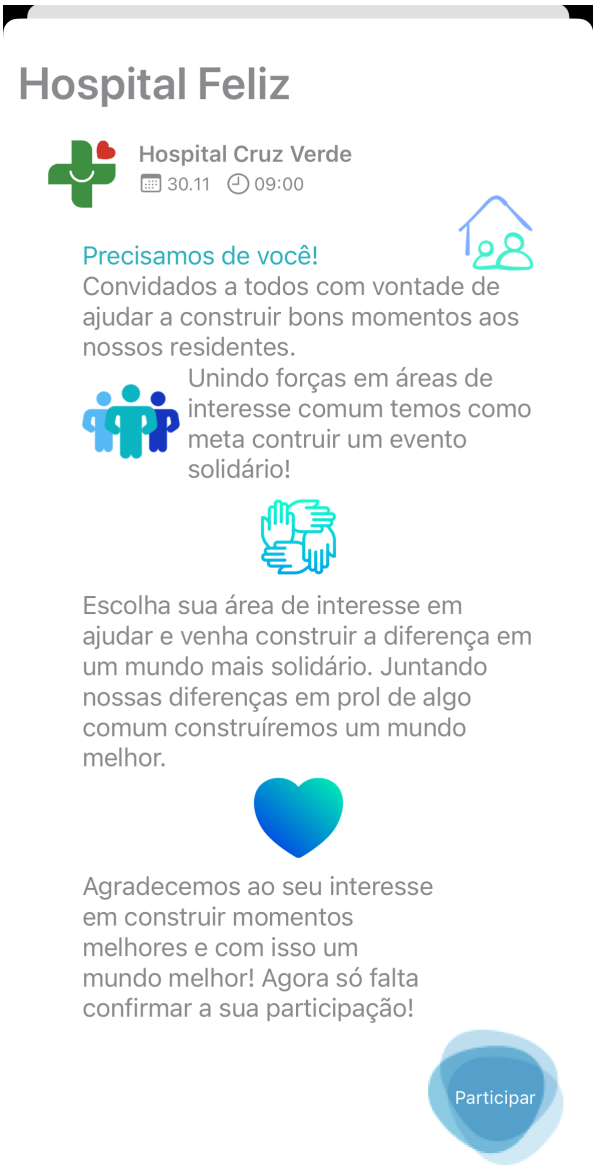


Fonte: Autoria Própria (2023).

4.2.4 Tela *Folder*

Na Figura 14 é possível ver a tela que será apresentada ao selecionar uma das opções do carrossel de *folders* da tela *Home*, apresentada na Figura 13. Nela são apresentadas as ações ou os eventos cadastrados no sistema e é possível ver as Instituições que possuem eventos ou filtrar pelo tipo da ação que desejar participar.

Figura 14 – Tela *Folder*

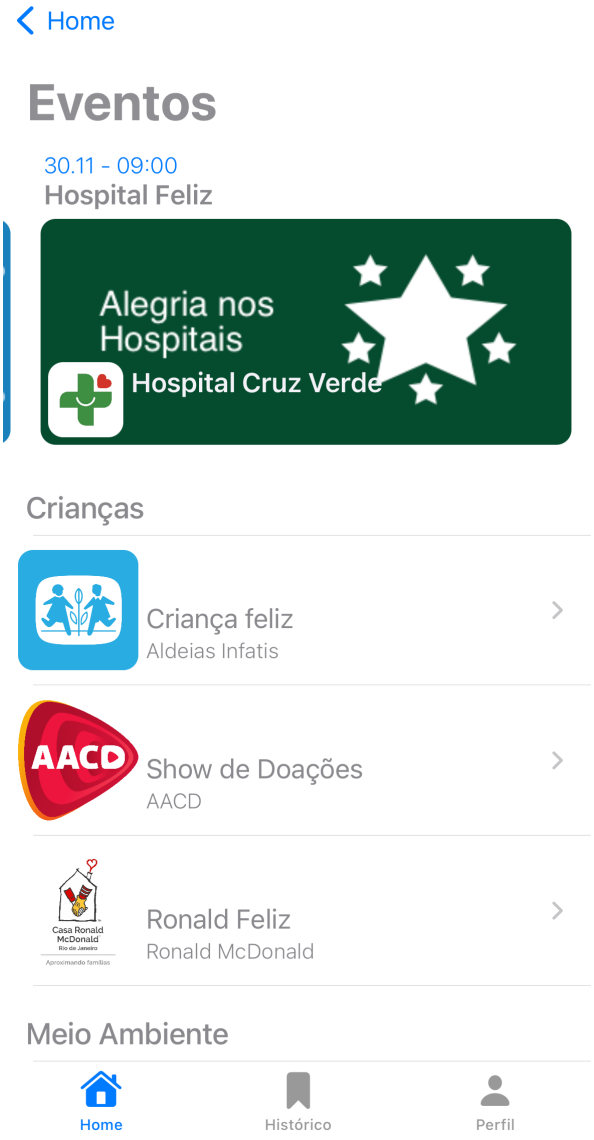


Fonte: Autoria Própria (2023).

4.2.5 Tela Lista de Eventos

A tela apresentada na Figura 15 aparece quando é selecionada a opção “Participe dos Eventos” na tela de *Home*. Nela são listadas todas as opções de Eventos cadastrados no sistema, apresentando as opções de acordo com as preferências do usuário e separadas pelas áreas de atuação. Na parte inferior da tela, existe um menu fixo com três opções: *Home*, Histórico e Perfil.

Figura 15 – Tela Lista de Eventos



Fonte: Autoria Própria (2023).

4.2.5.1 Tela Detalhes do Evento

Após selecionar um evento listado na tela Lista de Eventos (ver Figura 15), a tela Detalhes do Evento é apresentada, conforme ilustra a Figura 16. Essa tela apresenta todas as informações cadastradas relacionadas ao evento selecionado, além das áreas de atuação que são associadas ao mesmo. Na parte inferior da tela, existe um menu fixo com três opções: *Home*, Histórico e Perfil.

Figura 16 – Tela Detalhes do Evento

Fonte: Autoria Própria (2023).

4.2.6 Tela Lista de Instituições

A tela apresentada na Figura 17 aparece quando é selecionada a opção “Conheça as Instituições” na tela *Home*. Nela são listadas todas as Instituições que estão cadastradas no sistema, onde no topo apresenta-se uma frase de introdução e na parte principal aparecem as instituições listadas. Na parte inferior da tela, existe um menu fixo com três opções: *Home*, Histórico e Perfil.

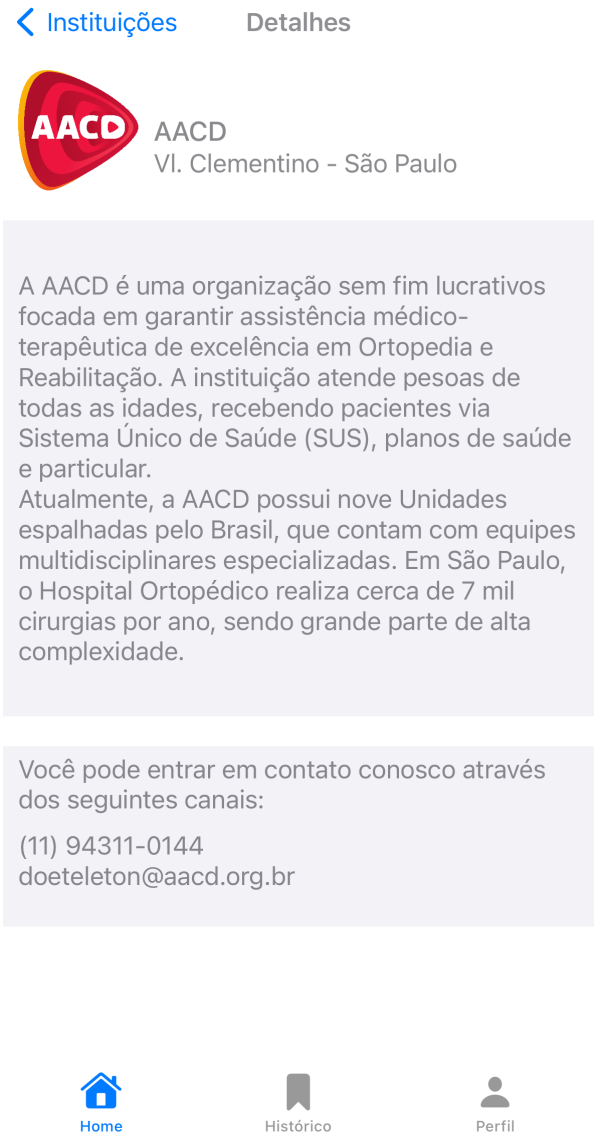
Figura 17 – Tela Lista de Instituições

Fonte: Autoria Própria (2023).

4.2.6.1 Tela Detalhes da Instituição

Após selecionar uma Instituição listada na tela de Lista de Instituições (ver Figura 17), a tela Detalhes da Instituição é apresentada, conforme ilustra a Figura 18. Essa tela apresenta todas as informações cadastradas relacionadas à Instituição selecionada, além das ações que são associadas a mesma. Na parte inferior da tela, existe um menu fixo com três opções: *Home*, Histórico e Perfil.

Figura 18 – Tela Detalhes da Instituição

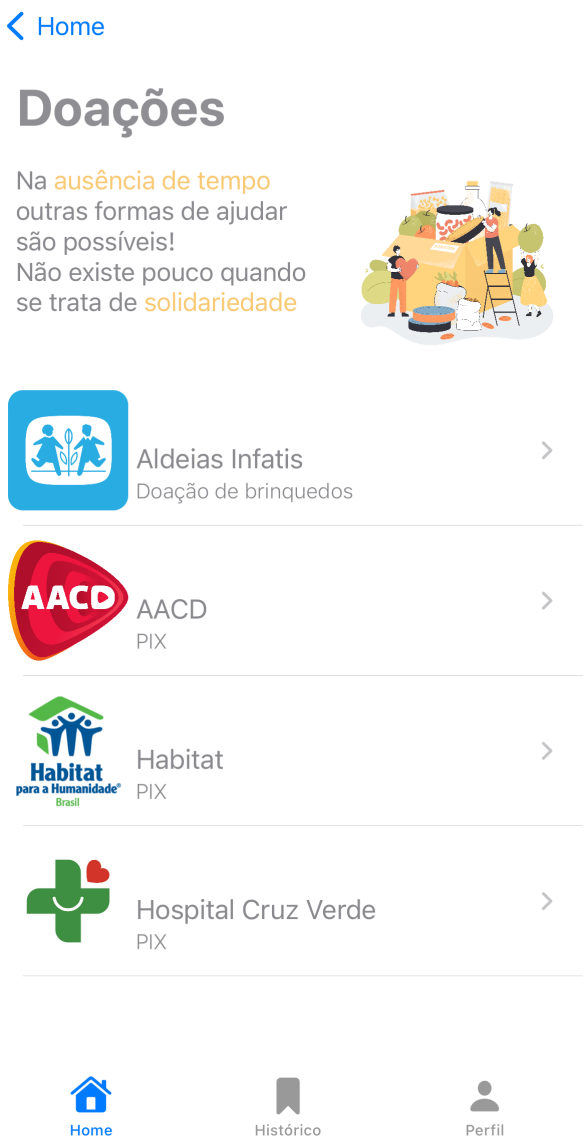


Fonte: Autoria Própria (2023).

4.2.7 Tela Lista de Doações

A tela apresentada na Figura 19 aparece quando é selecionada a opção “Mesmo longe Contribua” na tela *Home*. Nela são listadas todas as opções de Doações, que podem ser feitas à distância e que estão cadastradas no sistema, apresentando algumas opções de locais para doação ou instituições que necessitam de ajuda financeira. Na parte inferior da tela, existe um menu fixo com três opções: *Home*, Histórico e Perfil.

Figura 19 – Tela Lista de Doações



Fonte: Autoria Própria (2023).

4.2.8 Tela Detalhes da Doação

Após selecionar uma Doação listada na tela de Lista de Doações (ver Figura 19), a tela de Detalhes da Doação será apresentada, conforme ilustra a Figura 20. Essa tela apresenta todas as informações cadastradas relacionadas à doação selecionada, além das ações que são associadas a mesma. Na parte inferior da tela, existe um menu fixo com três opções: *Home*, Histórico e Perfil.

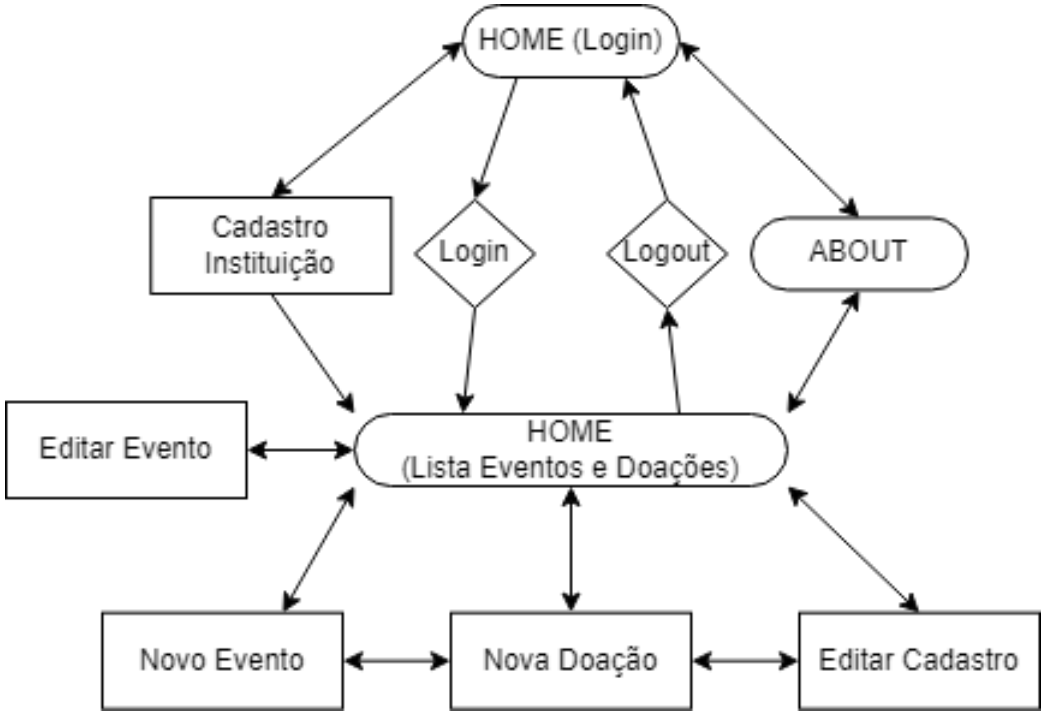
Figura 20 – Tela Detalhes da Doação

Fonte: Autoria Própria (2023).

4.3 A Aplicação *Web*

Na Figura 21 é possível visualizar o fluxo de telas da aplicação web.

Figura 21 – Fluxo de Telas da Aplicação Web



Fonte: Autoria Própria (2023).

A seguir são apresentadas as telas que compõem a interface gráfica da aplicação web.

4.3.1 Tela de *Login*

Na Figura 22 pode-se ver a tela de *login* do usuário. Na parte principal pode-se ver a logo do aplicativo, além dos campos para inserção dos dados de usuário e senha e o botão “Entrar”.

Na parte superior da tela, existe um menu fixo com três opções: Menu representado por “três tracinhos” na horizontal, que está apresentada na Seção 4.3.2, “Cadastre-se”, que está apresentada na Seção 4.3.3, e *“Login”*, que ao ser clicado retorna para a tela principal.

Figura 22 – Tela de *Login*

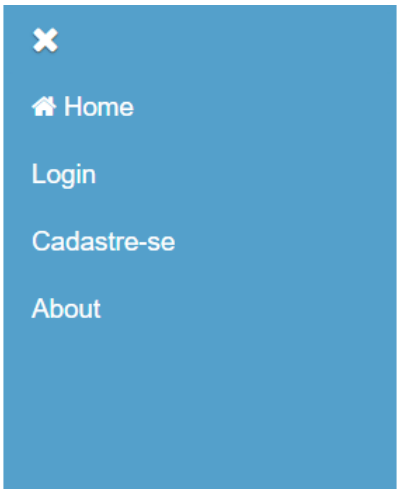
Fonte: Autoria Própria (2023).

4.3.2 Menu Lateral

Nas Figuras 23 e 24 pode-se ver a barra lateral que surge na tela ao clicar no botão com “três tracinhos” na horizontal, que pode ser visto na Figura 22 apresentada na seção anterior.

Quando o usuário não está logado, esse menu (ver Figura 23) apresenta quatro opções: *“Home”*, que abrirá a tela de *Login* apresentada na Seção 4.3.1, *“Login”*, apresentada na seção anterior, *“Cadastre-se”*, que é apresentada na Seção 4.3.3, e *“About”* que apresenta algumas informações sobre o projeto.

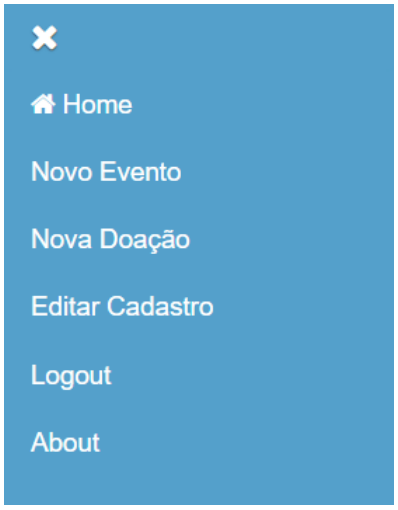
Figura 23 – Menu Lateral Instituição Deslogada



Fonte: Autoria Própria (2023).

Quando o usuário está logado, esse menu (Figura 24) apresenta seis opções: *“Home”*, que abrirá a tela inicial da Instituição, apresentada na Seção 4.3.3, *“Novo Evento”*, *“Nova Doação”*, *“Editar Cadastro”*, *“Logout”* e *“About”*.

Figura 24 – Menu Lateral Instituição Logada



Fonte: Autoria Própria (2023).

4.3.3 Telas Instituição

A tela apresentada na Figura 25 aparece quando é selecionada a opção “Cadastre-se” na tela de *Login*. Nela existe o formulário disponível para que as instituições possam se cadastrar e utilizar o sistema.

Na parte superior da tela, existe um menu fixo com três opções: Menu, representado por “três tracinhos” na horizontal, “Cadastre-se” e *“Login”*.

Figura 25 – Tela Cadastro Instituição

Fonte: Autoria Própria (2023).

Quando o usuário está logado no sistema, ao clicar na opção “Editar Cadastro”, apresentada no Menu Lateral, é possível editar o cadastro da instituição, como pode-se ver na Figura 26. Na parte superior da tela, existe um menu fixo com duas opções: Menu, representado por “três tracinhos” na horizontal e *Logout*.

Figura 26 – Tela Editar Cadastro

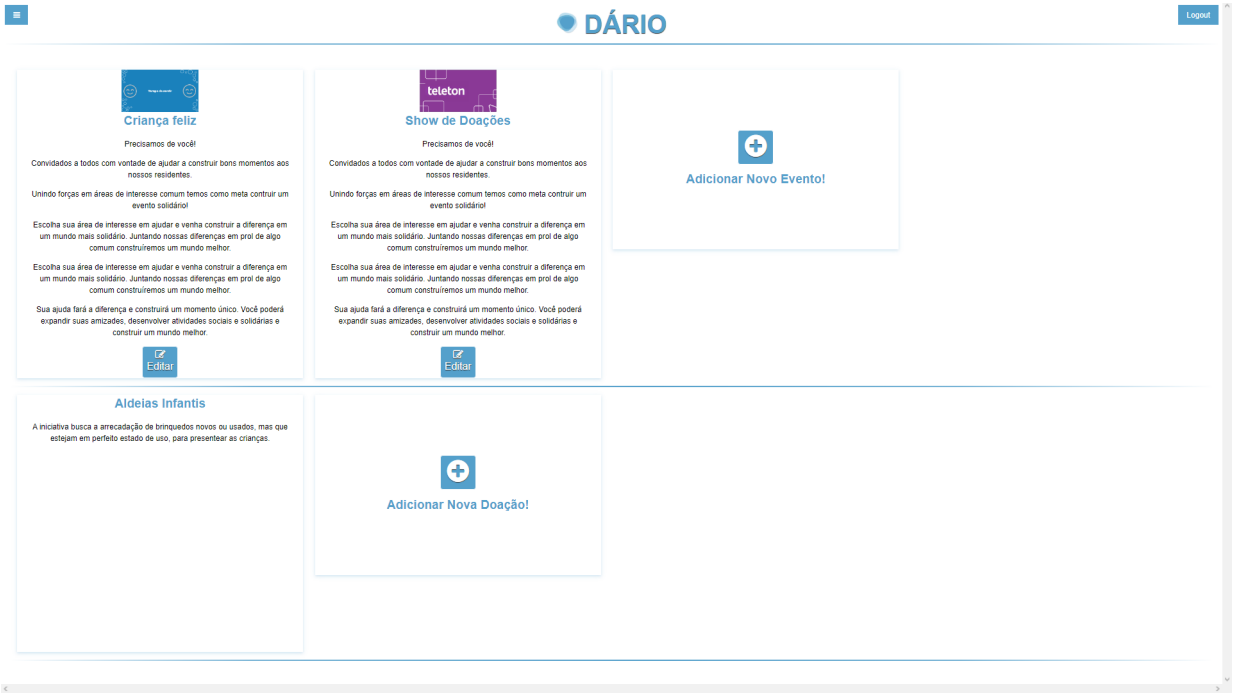
Fonte: Autoria Própria (2023).

4.3.4 Tela Lista de Eventos e Doações

A tela apresentada na Figura 27 aparece quando a Instituição realiza o *login* no sistema. Nessa tela existem duas partes: na parte superior é apresentada a lista dos eventos cadastrados e na parte inferior é apresentada a lista das opções de doações já cadastrados no sistema pelas Instituições. Essa tela também irá aparecer caso o usuário clique no botão *"Home"* presente no Menu Lateral.

Na parte superior da tela, existe um menu fixo com duas opções: Menu, representado por "três tracinhos" na horizontal e *Logout*.

Figura 27 – Tela Lista de Eventos e Doações



Fonte: Autoria Própria (2023).

4.3.4.1 Tela Cadastro Evento

Ao selecionar a opção “Adicionar Novo Evento!” (ver Figura 27), a tela para cadastrar um novo evento é apresentada, conforme ilustra a Figura 28. Nessa tela existe o formulário disponível para que as instituições possam cadastrar os eventos que são apresentados no aplicativo para os voluntários.

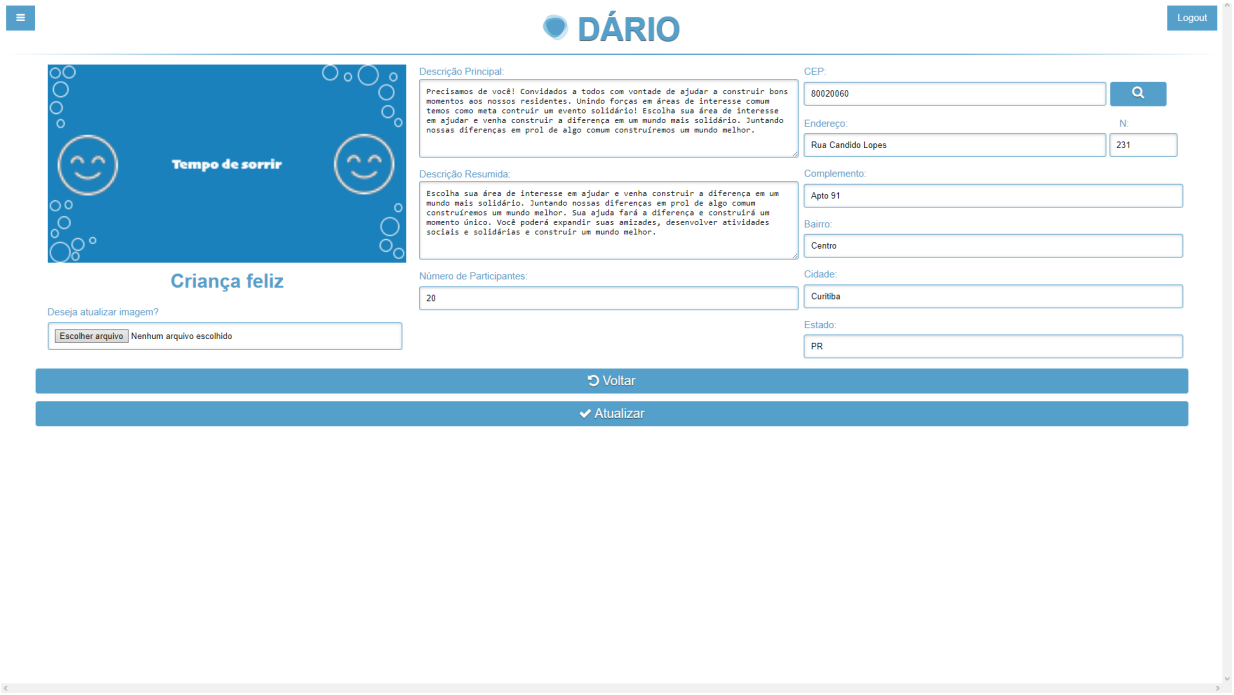
Na parte superior da tela, existe um menu fixo com duas opções: Menu, representado por “três tracinhos” na horizontal e *Logout*.

Figura 28 – Tela Novo Evento

Fonte: Autoria Própria (2023).

Ao selecionar o botão “Editar” que aparece abaixo de cada evento (ver Figura 27), a tela para editar o evento é apresentada, conforme ilustra a Figura 29.

Figura 29 – Tela Editar Evento



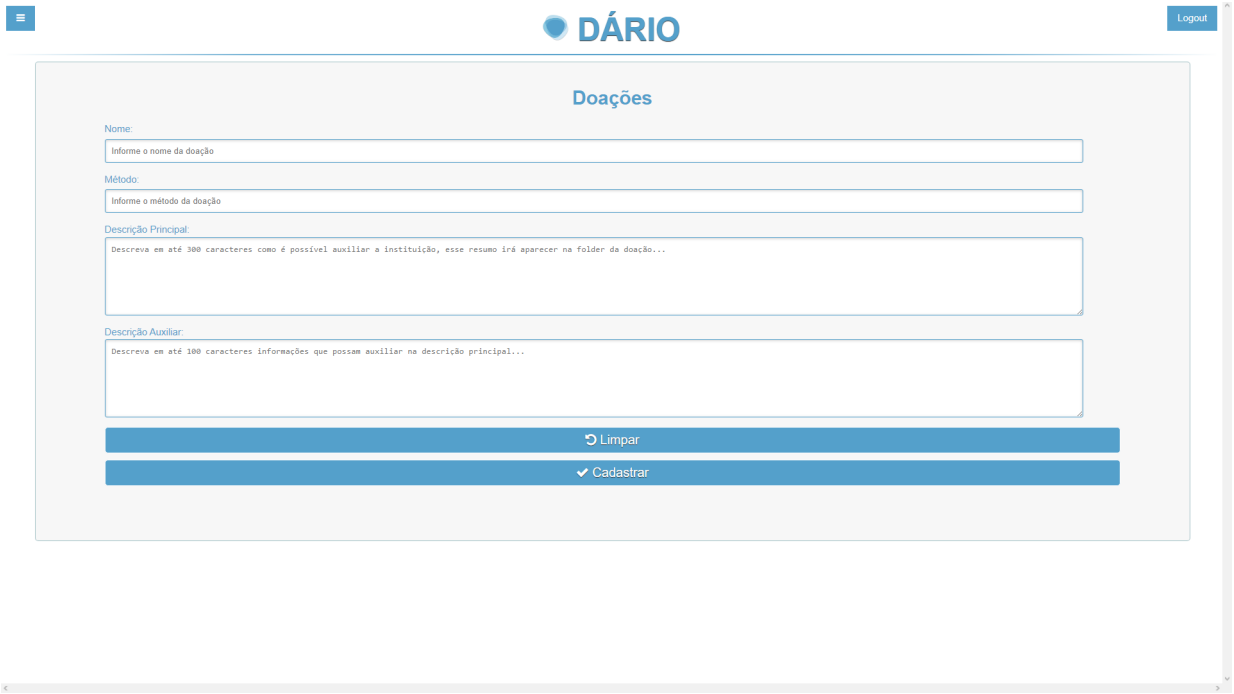
Fonte: Autoria Própria (2023).

4.3.4.2 Tela Cadastro Doação

Ao selecionar a opção “Adicionar Nova Doação!” (ver Figura 27), a tela para cadastrar um nova doação é apresentada, conforme ilustra a Figura 30. Nessa tela existe o formulário disponível para que as instituições possam cadastrar as formas de doação que são apresentadas no aplicativo para os voluntários.

Na parte superior da tela, existe um menu fixo com duas opções: Menu, representado por “três tracinhos” na horizontal e *Logout*.

Figura 30 – Tela Nova Doação



Fonte: Autoria Própria (2023).

5 CONCLUSÃO

Neste trabalho de conclusão de curso desenvolvemos o aplicativo móvel e a aplicação web Dário, um sistema completo para estudantes e instituições com o intuito de promover e melhorar a experiência no trabalho voluntário para ambos os públicos.

O aplicativo Dário não se limita apenas a fornecer uma plataforma de busca e inscrição em eventos, ele proporciona a criação de uma comunidade consistente, promovendo a solidariedade e o compromisso social. A escolha do nome “Dário” reflete a essência solidária do aplicativo, que visa conectar pessoas e instituições em prol de um objetivo e bem comum.

O Projeto Dário é composto por um aplicativo móvel para *iOS* e uma aplicação web, com o objetivo de auxiliar os estudantes na busca por atividades complementares sociais e também oferecer às instituições uma ferramenta eficaz para gerenciar eventos e voluntários, com a utilização da ferramenta de busca e inscrição.

Através da implementação de um sistema de recomendação, integrado ao aplicativo Dário, melhorando a experiência do usuário, que após um tempo de uso e interação com o aplicativo, recebe sugestão de eventos alinhados às suas preferências pessoais, buscando maximizar a correspondência entre voluntários e eventos, promovendo maior engajamento e participação.

Ainda no âmbito do voluntário, o aplicativo proporciona uma plataforma intuitiva para a busca e inscrição em eventos sociais, permitindo aos usuários acompanhar suas participações, receber certificados digitais e personalizar suas preferências de voluntariado, de forma simples, objetiva e de fácil acesso, potencializando a experiência nas atividades.

Referente às instituições, a aplicação web possibilita o cadastro eficiente de eventos, acompanhamento em tempo real da participação dos voluntários nestes eventos, com a validação de presença e a gestão transparente de recursos utilizados, encontrando voluntários que se encaixem com suas preferências.

No âmbito acadêmico, o projeto Dário é uma ferramenta valiosa para estudantes que buscam complementar sua formação e ampliar seus conhecimentos, acompanhando seus certificados e horas de dedicação, melhorando a experiência através da conexão de instituições e voluntários de acordo com seus valores e visão pessoal.

Em síntese, o aplicativo Dário não apenas atende às demandas por horas complementares, mas também fortalece o espírito voluntário e contribui para a construção de uma sociedade mais colaborativa e solidária, facilitando a união entre voluntários e instituições com ideias semelhantes. Este trabalho reflete não apenas a implementação de um aplicativo, mas a materialização de uma visão que busca unir tecnologia e solidariedade para o crescimento e interação de ambos, proporcionando benefício para todos.

5.1 Trabalhos Futuros

Como trabalhos futuros, sugere-se:

- Melhorar o sistema de autenticação do aplicativo e da aplicação web;
- Melhorar o *design* da aplicação web;
- Aprimorar as possíveis categorias e atuações que podem ser associadas aos eventos e instituições, abrangendo mais possibilidades para as preferências;
- Incluir *PUSH* de notificações;
- Incluir a criptografia do banco de dados, para manter os dados mais seguros;
- Utilizar requisições *HTTPS*, ao invés de requisições *HTTP*;
- Melhorar as validações e conformidade de dados;
- Empregar novos algoritmos no sistema de recomendação.

REFERÊNCIAS

AFONSO, A. **O que é Angular?** 2018. Disponível em: https://blog.algaworks.com/o-que-e-angular/. Acesso em: 10 jun. 2023.

ANGULAR. **Angular - NgModules**. 2023. Disponível em: https://angular.io/guide/ngmodules. Acesso em: 10 jun. 2023.

APPLE. **iOS & iPadOS Release Notes**. 2023. Disponível em: https://developer.apple.com/documentation/ios-ipados-release-notes.

APPLE. **Swift. Uma linguagem aberta e poderosa para todo mundo criar apps incríveis.** 2023. Disponível em: http://www.apple.com/br/swift/. Acesso em: 01 jun. 2023.

APPLE. **Swift.org**. 2023. Disponível em: https://swift.org. Acesso em: 01 jun. 2023.

BAILEY, K. Numerical taxonomy and cluster analysis. **Typologies and taxonomies**, v. 34, p. 24, 1994.

BATAGLIN, M. L. R.; CANTARELLI, G. S. **Conta Comigo: Aplicação Móvel para Gerenciamento de Doações, Eventos e Voluntariado do Terceiro Setor**. 2019 — Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação), Universidade Franciscana, Santa Maria, Rio Grande do Sul, 2019.

BEN-GAL, I. Bayesian networks (pdf). **Ruggeri, Fabrizio; Kennett, Ron S.; Faltin, Frederick W."Encyclopedia of Statistics in Quality and Reliability". Encyclopedia of Statistics in Quality and Reliability. John Wiley & Sons. doi**, v. 10, p. 9780470061572, 2007.

BOBADILLA, J. *et al.* A collaborative filtering approach to mitigate the new user cold start problem. **Knowledge-based systems**, Elsevier, v. 26, p. 225–238, 2012.

CANTELON, M. *et al.* **Node. js in Action**. [S.l.]: Manning Greenwich, 2014.

CORREIA, R. *et al.* **Referenciais de Formação para os Cursos de Graduação em Computação**. [S.l.]: SBC, 2017.

DESHPANDE, M.; KARYPIS, G. Item-based top-n recommendation algorithms. **ACM Transactions on Information Systems (TOIS)**, ACM New York, NY, USA, v. 22, n. 1, p. 143–177, 2004.

DWYER, G. **Flask By Example**. [S.l.]: Packt Publishing Ltd, 2016.

FLASK. **Welcome to Flask - Flask Documentation**. 2023. Disponível em: https://flask.palletsprojects.com/en/3.0.x/. Acesso em: 5 nov. 2023.

FORSATI, R. *et al.* Matrix factorization with explicit trust and distrust side information for improved social recommendation. **ACM Transactions on Information Systems (TOIS)**, ACM New York, NY, USA, v. 32, n. 4, p. 1–38, 2014.

FRANCESCO, R. **Recommender Systems Handbook. Ed. by Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor**. [S.l.]: Boston, MA: Springer US, 2011.

HERLOCKER, J. L. *et al.* Evaluating collaborative filtering recommender systems. **ACM Transactions on Information Systems (TOIS)**, ACM New York, NY, USA, v. 22, n. 1, p. 5–53, 2004.

HTML. **HTML Standard**. 2023. Disponível em: https://html.spec.whatwg.org/multipage/. Acesso em: 05 nov. 2023.

IYER, G. Node. js: Eventdriven concurrency for web applications. 2013. Disponível em: https://www.researchgate.net/publication/280546121_Nodejs_Event-driven_Concurrency_for_Web_Applications. Acesso em: 11 jun. 2023.

KARYPIS, G. Evaluation of item-based top-n recommendation algorithms. In: **Proceedings of the tenth international conference on Information and knowledge management**. [S.l.: s.n.], 2001. p. 247–254.

LÁZARO, A. d. S. **Análise e Seleção de Algoritmo de Filtragem de Informação para Solução do Problema Cold-Start Item**. 2010 — Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação), Universidade Federal de Lavras, Lavras, Minas Gerais, 2010.

LIKA, B.; KOLOMVATSOS, K.; HADJIEFTHYMIADES, S. Facing the cold start problem in recommender systems. **Expert systems with applications**, Elsevier, v. 41, n. 4, p. 2065–2073, 2014.

LOPS, P.; GEMMIS, M. D.; SEMERARO, G. Content-based recommender systems: State of the art and trends. **Recommender systems handbook**, Springer, p. 73–105, 2011.

MAIMON, O. Z.; ROKACH, L. **Data mining with decision trees: theory and applications**. [S.l.]: World scientific, 2014. v. 81.

MEDEIROS, I. **Estudo sobre Sistemas de Recomendação Colaborativos**. 2013 — Trabalho de Conclusão de Curso (Graduação em Ciência da Computação), Universidade Federal de Pernambuco, Recife, Pernambuco, 2013.

MEDIUM. **Collaborative based Recommendation system Using SVD**. 2020. Disponível em: https://medium.com/analytics-vidhya/collaborative-based-recommendation-system-using-svd-9adc5b6b3b8. Acesso em: 01 jun. 2023.

MELVILLE, P.; SINDHWANI, V. Recommender systems. **Encyclopedia of machine learning**, v. 1, p. 829–838, 2010.

MITCHELL, T. M. *et al.* **Machine learning**. [S.l.]: McGraw-hill New York, 2007. v. 1.

MUKAKA, M. M. A guide to appropriate use of correlation coefficient in medical research. **Malawi medical journal**, v. 24, n. 3, p. 69–71, 2012.

NODEJS. **An Easy Way to Build Scalable Network Programs**. 2011. Disponível em: https://nodejs.org/en/blog/uncategorized/an-easy-way-to-build-scalable-network-programs. Acesso em: 10 jun. 2023.

NODEJS. **About Node.js®**. 2023. Disponível em: https://nodejs.org/en/about. Acesso em: 10 jun. 2023.

PANNIELLO, U. *et al.* Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems. In: **Proceedings of the third ACM conference on Recommender systems**. [S.l.: s.n.], 2009. p. 265–268.

POSTGRESQL. **PostgreSQL 14.10 Documentation**. 2023. Disponível em: https://www.postgresql.org/docs/14/index.html. Acesso em: 08 jun. 2023.

POSTGRESQL. **PostgreSQL: About**. 2023. Disponível em: https://www.postgresql.org/about/. Acesso em: 08 jun. 2023.

PYTHON. **Python 3.12.1 documentation**. 2023. Disponível em: https://docs.python.org/3/. Acesso em: 5 nov. 2023.

RAMALHO, L. **Fluent python**. [S.l.]: "O'Reilly Media, Inc.", 2019.

RUSSELL, S. J. **Artificial intelligence a modern approach**. [S.l.]: Pearson Education, Inc., 2010.

SANTANA, M. **Deep Learning para Sistemas de Recomendação (Parte 1) - Introdução**. 2018. Disponível em: https://medium.com/data-hackers/deep-learning-para-sistemas-de-recomendacao-parte-1-introducao-b19a896c471e. Acesso em: 05 jun. 2023.

SARWAR, B. *et al.* Analysis of recommendation algorithms for e-commerce. In: **Proceedings of the 2nd ACM Conference on Electronic Commerce**. [S.l.: s.n.], 2000. p. 158–167.

SARWAR, B. *et al.* Item-based collaborative filtering recommendation algorithms. In: **Proceedings of the 10th international conference on World Wide Web**. [S.l.: s.n.], 2001. p. 285–295.

SCHAFER, J. B. *et al.* Collaborative filtering recommender systems. **The adaptive web: methods and strategies of web personalization**, Springer, p. 291–324, 2007.

SCHEIN, A. I. *et al.* Methods and metrics for cold-start recommendations. In: **Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval**. [S.l.: s.n.], 2002. p. 253–260.

SESHADRI, S.; GREEN, B. **Desenvolvendo com AngularJS: aumento de produtividade com aplicações web estruturadas**. [S.l.]: Novatec Editora, 2014.

SILVA, Á. F. G. da. **Acropora: um aplicativo para promoção do voluntariado**. 2021 — Trabalho de Conclusão de Curso (Graduação em Design), Universidade de Brasília, Brasília, Distrito Federal, 2021.

UTFPR. **Atividades Complementares**. 2018. Disponível em: http://www.utfpr.edu.br/cursos/coordenacoes/graduacao/curitiba/ct-sistemas-de-informacao/area-academica/atividades-complementares.

UTFPR. **Regulamento para Programas e Projetos de Extensão**. 2018. Disponível em: http://www.utfpr.edu.br/documentos/relacoes-empresariais-e-comunitarias/direxT/regulamentos/regulamento-para-programas-e-projetos-de-extensao/view.

UTFPR. **Nós - Nosso olhar solidário**. 2020. Disponível em: https://nossoolharsolidario.com.br/quem-somos-nos. Acesso em: 01 jun. 2023.

W3SCHOOLS. **HTML Element Reference**. 2023. Disponível em: https://www.w3schools.com/tags/default.asp.

WELSH, M. *et al.* A design framework for highly concurrent systems. **University of California, Berkeley**, 2000.

ZELDOVICH, N. *et al.* Multiprocessor support for event-driven programs. In: **USENIX Annual Technical Conference, General Track**. [S.l.: s.n.], 2003. p. 239–252.