Ministério da Saúde
FIOCRUZ
Fundação Oswaldo Cruz
Instituto Oswaldo Cruz
Laboratório de Comunicação Celular
Pós-Graduação *Stricto sensu* em Ensino em Biociências e Saúde

Rio de Janeiro

Junho de 2023

Sumário

1. Introdução

SeeForMe é uma revolucionária plataforma de assistência visual e aplicativo móvel. Seu propósito é estabelecer uma ponte entre indivíduos com deficiência visual, voluntários e inteligência artificial. O aplicativo visa facilitar e potencializar a rotina diária de pessoas com deficiências visuais, proporcionando-lhes assistência visual imediata conforme necessário, unindo voluntários prontos para prestar ajuda por meio de videochamadas ao vivo e respostas imediatas, ou por meio das novas tecnologias de inteligência artificial e machine learning.

O SeeForMe proporciona uma abordagem simples e eficiente para superar obstáculos visuais que muitas pessoas enfrentam, permitindo solicitações de ajuda em atividades que necessitam de visão, como a leitura de etiquetas, identificação de objetos, orientação em locais desconhecidos, entre outros. Ao utilizar a câmera de um smartphone ou tablet, os usuários podem utilizar da inteligência artificial do SeeForMe para oferecer áudio descrição dos objetos transmitidos pela câmera do celular, além da possibilidade de participar de videochamadas ao vivo com os voluntários, que, por sua vez, oferecem instruções e descrições visuais para auxiliá-los.

A principal força do SeeForMe é sua comunidade de voluntários, inscritos para contribuir com o aplicativo e disponibilizar seus conhecimentos visuais aos usuários. Essa malha de suporte global possibilita a conexão de pessoas com deficiências visuais a voluntários ao redor do mundo, falantes de diversas línguas, ampliando assim o espectro de ajuda disponível. Além disso, o SeeForMe empenha-se na garantia da privacidade e segurança dos usuários, aplicando medidas de proteção de dados e assegurando a transparência em suas políticas de privacidade.

O aplicativo SeeForMe tem um papel crucial na promoção da inclusão e independência de pessoas com deficiências visuais, oferecendo uma plataforma pioneira que une tecnologia móvel e a generosidade da comunidade para tornar o mundo visualmente acessível.

2. Proposta de estrutura geral de um projeto de software

Desenvolver um software é uma tarefa complexa que envolve várias etapas. Cada uma dessas etapas possui sua importância e especificidades.

Abaixo, descrevo cada etapa com mais detalhes e fundamentação com base em autores renomados na área de desenvolvimento e engenharia de software:

2.1. Definição de Requisitos

2.1.1 Coleta de Requisitos:

Exemplo prático: Imagine que você está desenvolvendo um aplicativo de entrega de comida. Durante a coleta de requisitos, você realiza uma pesquisa online com os usuários para identificar suas preferências alimentares, restrições dietéticas e funcionalidades desejadas no aplicativo. Com base nessas informações, você pode definir os requisitos para o sistema, como suporte a diferentes tipos de alimentos, filtros de pesquisa por dieta e um recurso de favoritos para os usuários salvarem seus restaurantes preferidos.

2.1.2 Análise de Requisitos:

Exemplo prático: Continuando com o exemplo do aplicativo de entrega de comida, ao analisar os requisitos coletados, você identifica que alguns usuários solicitaram um recurso de pagamento em dinheiro no ato da entrega. No entanto, você também observa que essa funcionalidade pode aumentar os riscos de segurança e atrasos nas entregas. Nesse caso, você precisa analisar os prós e contras, considerar as restrições operacionais e decidir se deve incluir ou não esse requisito no sistema.

2.2 Projeto (Design)

2.2.1 Projeto Arquitetural:

Exemplo prático: Suponha que você esteja projetando um sistema de gerenciamento de tarefas colaborativo. Você define uma arquitetura baseada em microserviços, onde cada funcionalidade do sistema é implementada como um serviço independente. Você projeta um serviço de autenticação e autorização separado, um serviço de armazenamento de dados e um serviço de notificações. Esses serviços se comunicam entre si por meio de APIs bem definidas, formando a estrutura do sistema.

2.2.2 Projeto Detalhado:

Exemplo prático: Seguindo com o sistema de gerenciamento de tarefas, no projeto detalhado, você descreve em detalhes como cada componente será implementado. Por exemplo, você projeta a interface do usuário com um layout intuitivo e responsivo, detalha a lógica de negócios para criar, atribuir e concluir tarefas, e define como as notificações serão enviadas aos usuários por e-mail ou via aplicativo móvel.

2.3 Implementação (Codificação)

Exemplo prático: Agora, na fase de implementação, você começa a escrever o código-fonte do sistema. Utilizando uma linguagem de programação como Python, você cria classes e métodos para representar as diferentes entidades do sistema, como usuários, tarefas e notificações. Você implementa a lógica de negócios para manipular essas entidades, realiza a integração com bibliotecas de terceiros, e realiza testes unitários para garantir a correção do código.

Espero que esses exemplos mais criativos tenham sido úteis para ilustrar os procedimentos de definição de requisitos, projeto e implementação. Se precisar de mais exemplos ou tiver mais dúvidas, estou à disposição para ajudar!

3. **Telly Stories**

Os Telly Stories, ou "user stories", são uma técnica de desenvolvimento de software que originou-se no desenvolvimento ágil para ajudar as equipes a entenderem o produto sob a perspectiva do usuário. Aqui estão algumas histórias de usuários possíveis para o sistema SeeForMe:

“Como usuário com deficiência visual, quero poder criar uma conta fornecendo minhas informações básicas para que possa usar os serviços do aplicativo.”

“Como usuário com deficiência visual, quero solicitar assistência visual por meio de chamadas de vídeo em tempo real, para que os voluntários possam me fornecer orientações e descrições visuais”

"Como usuário com deficiência visual, quero ser capaz de compartilhar o que estou vendo através da câmera do meu smartphone ou tablet, para que o SeeForMe me auxilie com áudio descrição dos objetos."

"Como voluntário, quero ser capaz de me inscrever para ajudar, fornecendo minhas informações básicas e especificando minhas áreas de especialização e horários disponíveis."

"Como voluntário, quero receber notificações quando um usuário com deficiência visual solicitar ajuda durante meus horários disponíveis, para que possa atender à chamada e prestar assistência."

"Como usuário com deficiência visual, quero ser capaz de fornecer feedback após cada chamada, para ajudar a melhorar a qualidade do serviço."

4. Regras de Negócio

A criação de um software é um processo complexo que envolve diversas etapas, desde a concepção da ideia até o lançamento do produto final. Uma das partes mais cruciais nesse processo é a definição das regras de negócio, que estabelecem os critérios e diretrizes que orientam a operação do software.

As regras de negócio são os pilares que sustentam a funcionalidade e a operação de um sistema de software. Elas são definidas pela organização e refletem a maneira como ela conduz seus negócios (Davenport, 1993). As regras de negócio são essenciais para garantir que o software atenda aos requisitos da organização e fornecem um framework para o desenvolvimento de software.

O desenvolvimento de regras de negócio deve levar em consideração uma variedade de fatores, incluindo a estratégia de negócios da organização, as necessidades e preferências dos usuários, os regulamentos legais e a tecnologia disponível (Porter, 1980). Por exemplo, no caso de um software de comércio eletrônico, as regras de negócio podem incluir políticas de preços, opções de entrega, políticas de retorno e procedimentos de pagamento.

A implementação eficaz de regras de negócio em software requer uma

abordagem sistemática que inclui a identificação, a documentação, a implementação e a manutenção das regras (Jacobson, Booch, & Rumbaugh, 1999). A identificação das regras de negócio envolve a análise dos processos de negócios da organização e a determinação dos critérios que devem orientar a operação do software.

A documentação das regras de negócio é essencial para garantir a consistência e a clareza na implementação do software. Ela deve ser precisa, completa e de fácil compreensão para todos os envolvidos no projeto (Sommerville, 2011).

A implementação das regras de negócio envolve a tradução dessas regras em código de software. As regras de negócio devem ser implementadas de uma forma que permita sua fácil alteração e atualização, a fim de acomodar mudanças nos processos de negócios ou regulamentos legais (Pressman, 2010).

Por fim, a manutenção das regras de negócio é uma parte crucial do ciclo de vida do software. Isso envolve a atualização das regras conforme necessário e a garantia de que continuem a orientar a operação do software de forma eficaz (Pfleeger & Atlee, 2010).

Em resumo, as regras de negócio desempenham um papel fundamental na criação de software. Elas orientam o desenvolvimento e a operação do software e garantem que ele atenda às necessidades da organização e dos usuários.
No contexto do projeto SeeForMe, as regras de negócio podem incluir:

ID	**NOME**	**DESCRIÇÃO**
RN01	Registro de Usuários	Todos os usuários, sejam eles voluntários ou pessoas com deficiência visual, devem se cadastrar no aplicativo, fornecendo informações básicas para identificação e segurança.
RN02	Verificação de Voluntários	Antes de um voluntário poder começar a ajudar, ele deve passar por um processo de verificação para confirmar sua identidade e garantir que está apto a prestar assistência visual.
RN03	Solicitação de Assistência	Usuários com deficiência visual podem solicitar assistência visual sempre que precisarem. Os pedidos são encaminhados para voluntários disponíveis no momento.

RN04	Privacidade dos Usuários	A privacidade dos usuários é de suma importância. Os dados dos usuários são protegidos com medidas robustas de segurança, e a transparência é mantida em relação às políticas de privacidade.
RN05	Conexão em Tempo Real	A plataforma deve permitir a transmissão em tempo real de vídeo entre o usuário com deficiência visual e o voluntário, para que a assistência possa ser fornecida efetivamente.
RN06	Assistência Multilingue	A plataforma deve permitir que pessoas com deficiência visual se conectem com voluntários que falem a mesma língua, ampliando as possibilidades de assistência.
RN07	Feedback e Avaliações	Após cada sessão de assistência, tanto o usuário com deficiência visual quanto o voluntário devem poder deixar feedback e avaliações sobre a experiência. Isso pode ajudar a manter a qualidade do serviço e identificar áreas de melhoria.
RN08	Assistência Não Profissional	Os voluntários fornecem assistência não profissional e não substituem a assistência profissional (por exemplo, a de um médico oftalmologista).
RN09	Não Substituição de Tecnologias Assistivas	O aplicativo não substitui outras tecnologias assistivas, mas serve como uma ferramenta adicional de suporte para pessoas com deficiência visual.
RN10	Identificar objeto em tempo real	Usuários com deficiência visual podem utilizar a câmera do celular para detectar objetos em tempo real, sem a necessidade de um voluntário.
RN11	Identificar objeto por foto	Usuários com deficiência visual podem realizar o envio de imagens para a identificação de um objeto.
RN12	Cadastrar objeto não identificado	Caso o objeto não for detectado automaticamente pelo sistema, o usuário com deficiência visual poderá enviar a imagem para os voluntários descreverem o objeto.

RN13	Leitura de rótulos	Usuários com deficiência visual podem solicitar leitura de rótulos, para obterem informações presentes em embalagens como: modo de uso, ingredientes, etc.
RN14	Notificação para Deficientes VIsuais	Quando a imagem enviada pelo usuário com deficiência visual for identificada por um usuário voluntário, uma notificação será enviada.
RN15	Cadastrar novos objetos	Os usuários voluntários podem enviar imagens de objetos identificados para acrescentar ao banco de dados.
RN16	Visualizar lista de imagens	Os usuários voluntários podem visualizar uma lista de imagens que ainda não foram identificadas.
RN17	Avaliar imagens cadastradas	Os usuários com permissão devem avaliar as imagens enviadas pelos voluntários para permitir que sejam introduzidas ao banco de dados.
RN18	Identificar objetos	Os usuários voluntários podem identificar objetos não identificados em imagens enviadas por deficientes visuais.
RN19	Notificação para Voluntários	Quando um novo objeto não identificado for enviado por um usuário com deficiência visual, uma notificação será enviada ao voluntário.

5. Requisitos Funcionais

Os requisitos funcionais de um software são uma parte integral do desenvolvimento do sistema. Eles descrevem as funcionalidades que o software deve ter para atender às necessidades dos usuários e aos objetivos de negócio da organização.

Kotonya e Sommerville (1998) definem os requisitos funcionais como declarações de serviços que um sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações. De maneira simplificada, os requisitos funcionais definem o "o que" o sistema deve fazer. Os requisitos funcionais de um sistema descrevem as

funcionalidades que o sistema deve fornecer, os serviços que deve executar, as tarefas que deve realizar e as operações que deve suportar. Eles são especificados no nível de detalhe que os desenvolvedores do sistema precisam para projetar, construir e testar o sistema.

Com base no sistema SeeForMe descrito anteriormente, aqui estão alguns possíveis requisitos funcionais:

ID	NOME	DESCRIÇÃO
RF01	Registro de usuário:	O sistema deve fornecer uma interface onde os usuários (pessoas com deficiência visual e voluntários) possam se cadastrar, inserindo informações básicas para a identificação.
RF02	Verificação de Voluntários	O sistema deve ser capaz de verificar as credenciais dos voluntários durante o processo de registro, para garantir que estão aptos a fornecer assistência visual.
RF03	Solicitação de Assistência:	O sistema deve permitir que usuários com deficiência visual solicitem assistência visual, com a capacidade de detalhar o tipo de assistência necessária.
RF04	Matching de Usuários:	O sistema deve ser capaz de conectar usuários com deficiência visual a voluntários disponíveis, levando em consideração fatores como disponibilidade, habilidades, idioma e localização.
RF05	Transmissão em Tempo Real	O sistema deve permitir a transmissão de vídeo em tempo real entre o usuário com deficiência visual e o voluntário, usando a câmera do dispositivo do usuário.
RF06	Feedback e Avaliações:	O sistema deve fornecer um mecanismo onde usuários e voluntários possam fornecer feedback e avaliar a experiência após cada sessão.
RF07	Multilíngue:	O sistema deve suportar vários idiomas para acomodar usuários e voluntários de diferentes origens linguísticas.
RF08	Proteção de Dados:	O sistema deve implementar medidas de segurança robustas para proteger as informações pessoais e os dados dos usuários.

RF09	Suporte ao Usuário:	O sistema deve fornecer suporte ao usuário para resolver problemas ou responder a perguntas relacionadas ao uso do aplicativo.
RF10	Detectar objetos	O sistema deve ser capaz de detectar objetos por meio da câmera ou fotos enviadas.
RF11	Identificar objetos	O sistema deve ser capaz de identificar objetos detectados, utilizando inteligência artificial para reconhecê-los.
RF12	Descrever objetos	O sistema deve ser capaz de descrever os objetos identificados para os usuários com deficiência visual, por meio de áudio.
RF13	Filtrar envios	O sistema deve permitir a avaliação de imagens enviadas pelos voluntários, com a finalidade de filtrar conteúdos impróprios.
RF14	Enviar notificações	O sistema deve enviar notificações aos usuários com deficiência visual quando uma de suas imagens for identificada; O sistema também deverá notificar os usuários voluntários quando uma nova imagem não identificada for enviada.

6. Requisitos Não Funcionais

Os requisitos não funcionais, também conhecidos como propriedades de qualidade ou restrições do sistema, definem como um sistema deve operar, ao contrário dos requisitos funcionais que descrevem o que um sistema deve fazer. Eles estabelecem as condições ou serviços que o sistema deve fornecer, mas não o comportamento específico. Em outras palavras, os requisitos não funcionais estabelecem o "como" em vez do "o que".

Segundo Sommerville (2011), os requisitos não funcionais descrevem as características do sistema que têm a ver com a implementação e a operação do sistema, tais como confiabilidade, desempenho, segurança, manutenibilidade, entre outros. Eles são frequentemente mais críticos para o sucesso de um sistema de software do que os requisitos funcionais, pois um sistema pode funcionar de acordo com os requisitos funcionais, mas se falhar em cumprir os requisitos não funcionais, pode ser considerado inaceitável pelos usuários.

Para o aplicativo SeeForMe, podemos listar os seguintes requisitos não funcionais:

ID	NOME	DESCRIÇÃO
RNF01	Acessibilidade	Garantir que o aplicativo seja acessível para pessoas com deficiência visual, seguindo as diretrizes de acessibilidade, como suporte a leitores de tela e controle por gestos
RNF02	Segurança e privacidade	O aplicativo deverá proteger as informações pessoais dos usuários, garantindo a segurança e a privacidade dos dados transmitidos durante as chamadas de vídeo.
RNF03	Disponibilidade	Assegurar que o aplicativo esteja disponível e funcional na maioria dos dispositivos móveis e redes, para que os usuários possam acessá-lo quando necessário.
RNF04	Desempenho	Garantir que o aplicativo seja responsivo e tenha um desempenho adequado durante as chamadas de vídeo, evitando atrasos ou interrupções significativas.
RNF05	Multiplataforma	Permitir que o aplicativo seja executado em diferentes plataformas, como iOS e Android, para alcançar um maior número de usuários.
RNF06	Tecnologia do voluntário digital	O aplicativo usará a tecnologia do GPT-4 da OpenAI para descrever imagens em texto.

7. Modelagem BPMN

A modelagem de processos de negócio (Business Process Modeling Notation - BPMN) é uma metodologia padronizada para a representação gráfica de processos de negócios. Como ferramenta, o BPMN permite uma visão clara, precisa e compreensível do fluxo de atividades e informações dentro de uma organização (White, 2004). Os diagramas BPMN fornecem uma linguagem visual comum que facilita a comunicação entre os stakeholders, promovendo uma compreensão clara dos processos de negócios e suas interações.

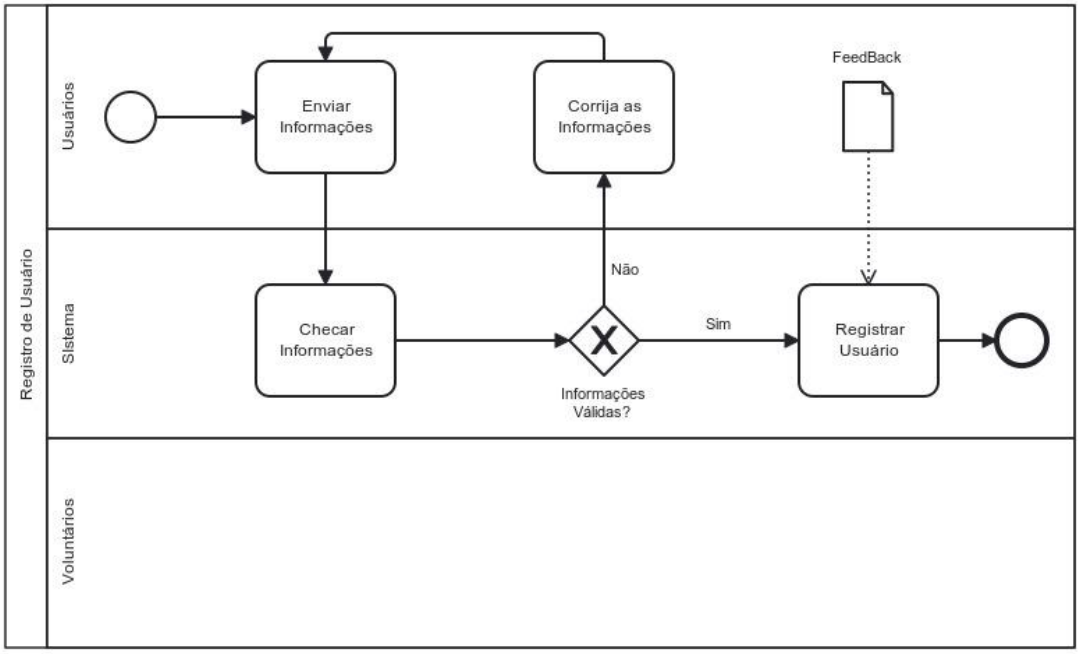
A aplicação da BPMN ao projeto do aplicativo SeeForMe envolveria a criação de um conjunto de diagramas para representar as várias funções e processos do sistema. Como exemplo simplificado, vamos considerar um processo central do SeeForMe: a solicitação de assistência por parte de um usuário com deficiência visual. Uma possível modelagem para o sistema seria:

ID	NOME	DESCRIÇÃO	ATOR
P01	Autenticar	Autenticar e autorizar utilizador da plataforma	Usuário
P02	Selecionar Imagem	Buscar imagens no dispositivo ou câmera	Usuário
P03	Enviar Imagem	Enviar imagem selecionada	Usuário
P04	Processar Imagem	Reconhecer e analisar imagem enviada	Sistema
P05	Retornar Análise	Enviar análise da imagem enviada	Sistema
P06	Solicitar Ajuda	Solicitar ajuda de um voluntário disponível	Usuário
P07	Buscar Voluntário	Buscar voluntários disponíveis	Sistema
P08	Atender Chamada	Atender solicitação para estabelecer ajuda	Voluntário
P09	Estabelecer Comunicação	Gerenciar comunicação entre usuário e voluntário	Sistema
P10	Coletar Estatísticas	Coletar estatísticas técnicas e de resultado da comunicação	Sistema

7.1 Registro de usuário

Um voluntário interessado em receber assistência visual no SeeForMe deve se registrar no aplicativo, fornecendo as informações necessárias.

Figura 1 - Modelagem BPMN Registro de usuário

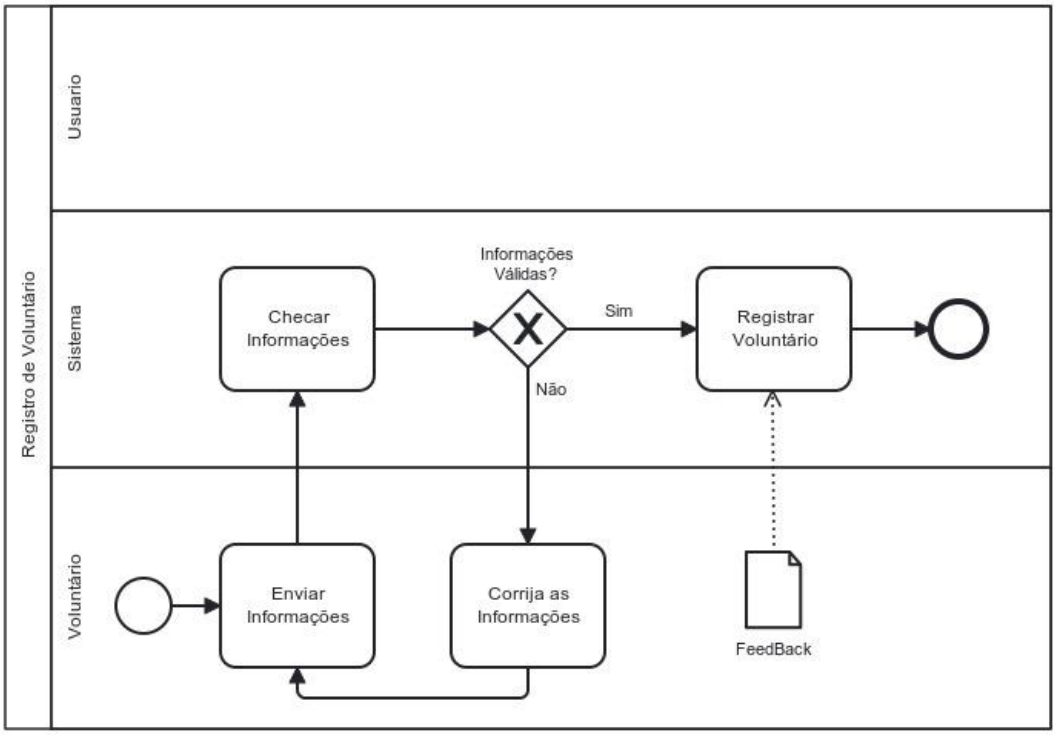


Fonte: Material do autor

7.2 Registro de Voluntários

Um voluntário interessado em fornecer assistência visual no SeeForMe deve se registrar no aplicativo, fornecendo as informações necessárias

Figura 2 - Modelagem BPMN Registro de Voluntários

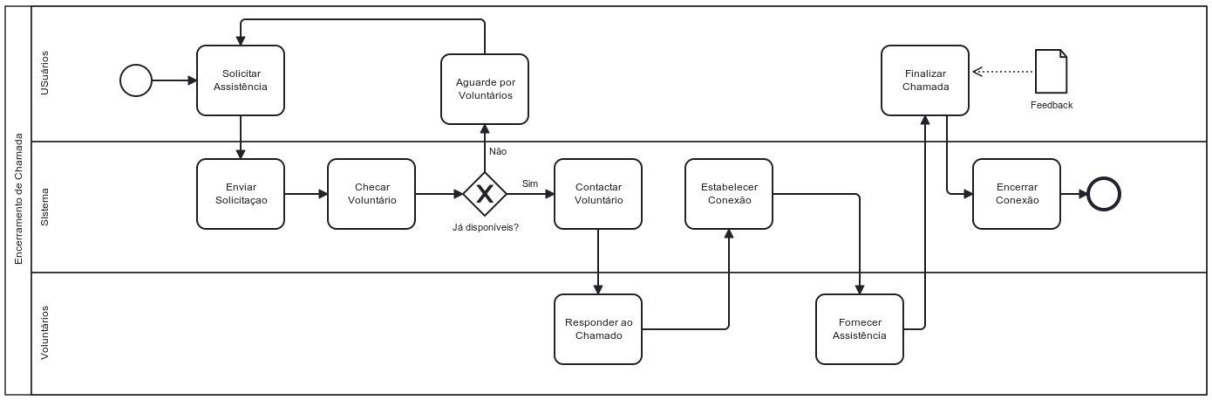


Fonte: Material do autor

7.3 Encerramento de chamada

Tanto o usuário quanto o voluntário têm a opção de encerrar a chamada quando a assistência visual é concluída ou se surgir algum problema.

Figura 3 - Modelagem BPMN Encerramento de chamada



Fonte: Material do autor

8. Diagrama de Caso de Uso

Os diagramas de casos de uso são uma representação gráfica que mostra como os usuários (chamados de atores) interagem com um sistema, ajudando a entender o que o sistema deve fazer. Desenvolvido como parte da Linguagem Unificada de Modelagem (UML), esse diagrama é uma ferramenta essencial durante a análise de requisitos de um software.

Os principais elementos de um diagrama de casos de uso são os atores (que podem ser usuários ou outros sistemas com os quais o sistema em questão se comunica) e os casos de uso (as ações que os atores podem executar no sistema). Os atores estão fora do sistema e interagem com ele através dos casos de uso.

Para construir um diagrama de casos de uso, os seguintes passos podem ser seguidos:

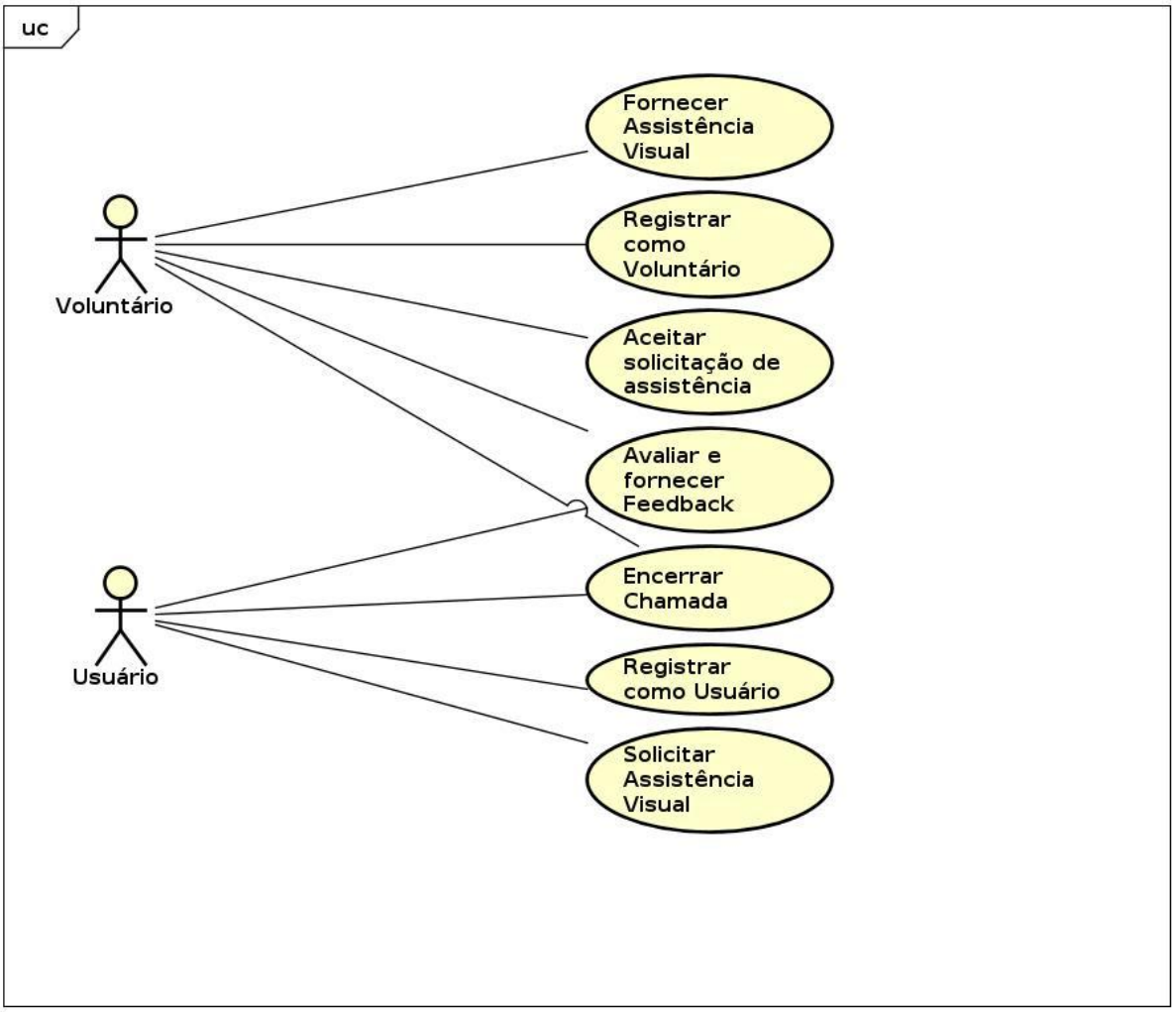
1. **Identificar os atores:** Comece listando todas as partes que interagem com o sistema. Eles podem ser usuários humanos, como clientes ou funcionários, ou outros sistemas.

2. **Identificar os casos de uso:** Após identificar os atores, liste todas as funções ou ações que esses atores podem realizar no sistema.

3. **Desenhar o diagrama:** Desenhe um retângulo para representar o sistema e coloque os casos de uso dentro do retângulo. Coloque os atores fora do retângulo e conecte os atores aos casos de uso com linhas para mostrar a interação entre eles.

4. **Detalhar os casos de uso:** Cada caso de uso pode ser detalhado descrevendo o fluxo de eventos, incluindo o fluxo básico (a sequência padrão de eventos) e os fluxos alternativos (eventos alternativos que podem ocorrer).

Figura 4 - Diagrama de Casos de Uso



Fonte: Material do Autor

9. Casos de uso textual

Caso de Uso: Fornecer Assistência Visual (UC-01)
Atores envolvidos: Voluntário
Fluxo principal: O voluntário recebe uma notificação de solicitação de assistência visual. O voluntário aceita a solicitação de assistência. O aplicativo estabelece uma chamada de vídeo em tempo real entre o voluntário e o usuário. O voluntário utiliza a câmera traseira do dispositivo para transmitir vídeo ao vivo. O usuário descreve a tarefa ou a necessidade de assistência visual ao voluntário. O voluntário fornece assistência visual, guiando o usuário e respondendo às suas perguntas. O voluntário conclui a tarefa ou auxilia o usuário até que a assistência seja completa. O voluntário encerra a chamada.
Conclusão: O voluntário forneceu a assistência visual solicitada pelo usuário. A chamada de vídeo em tempo real foi encerrada com sucesso.

Caso de Uso: Registrar como voluntário (UC-02)
Atores envolvidos: Voluntário
Fluxo principal: O voluntário abre o aplicativo SeeForMe. O voluntário seleciona a opção de registro como voluntário. O voluntário fornece as informações solicitadas, como nome, endereço de e-mail e outras informações relevantes. O voluntário concorda com os termos e condições do serviço. O voluntário conclui o registro.
Conclusão: O voluntário concluiu com sucesso o registro como voluntário.

Caso de Uso: Aceitar Solicitação de Assistência (UC-03)
Atores envolvidos: Voluntário
Fluxo principal: O voluntário recebe uma notificação de solicitação de assistência visual. O voluntário abre o aplicativo SeeForMe. O voluntário visualiza os detalhes da solicitação, incluindo informações básicas sobre o usuário e suas necessidades. O voluntário aceita a solicitação de assistência visual. O aplicativo estabelece uma chamada de vídeo em tempo real entre o voluntário e o usuário.
Conclusão: O voluntário aceitou com sucesso a solicitação de assistência e a chamada de vídeo em tempo real foi iniciada.

Caso de Uso: Avaliar e Fornecer Feedback (UC-04)
Atores envolvidos: Voluntário, Usuário
Fluxo principal: Após a conclusão da assistência visual, tanto o voluntário quanto o usuário têm a oportunidade de avaliar a qualidade do serviço. O voluntário ou o usuário abre o aplicativo SeeForMe. O voluntário ou o usuário seleciona a opção de avaliação e feedback. O voluntário ou o usuário atribui uma classificação e fornece comentários adicionais, se desejar. O voluntário ou o usuário envia a avaliação e o feedback.
Conclusão: O voluntário ou o usuário avaliou a qualidade do serviço fornecido durante a assistência visual. O feedback foi enviado com sucesso.

Caso de Uso: Encerrar a Chamada (UC-05)
Atores envolvidos: Voluntário, Usuário
Fluxo principal: Tanto o voluntário quanto o usuário têm a opção de encerrar a chamada a qualquer momento. O voluntário ou o usuário seleciona a opção de encerrar a chamada. A chamada de video em tempo real é encerrada.
Conclusão: A chamada foi encerrada com sucesso pelo voluntário ou pelo usuário.

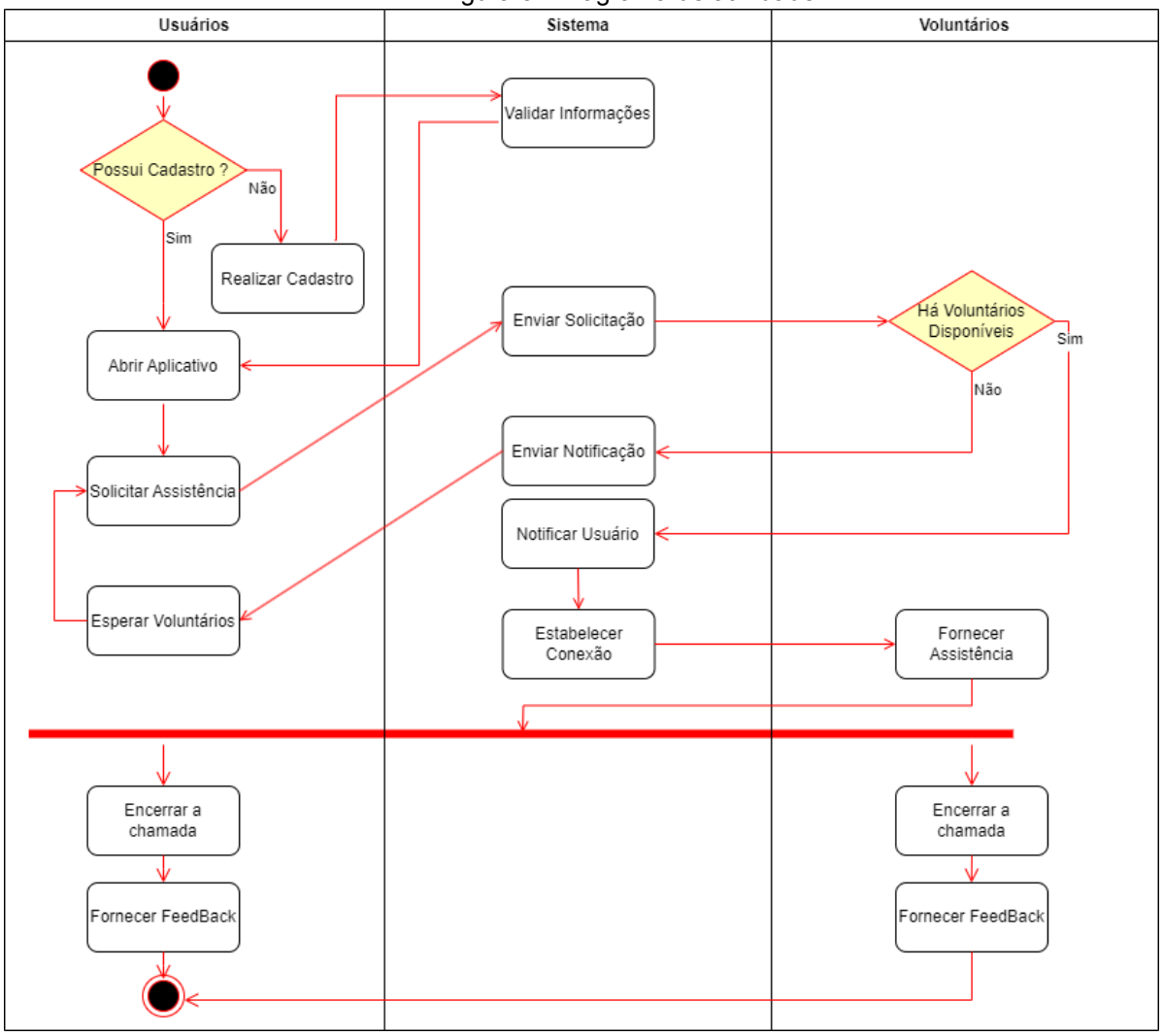
Caso de Uso: Registrar como Usuário (UC-06)
Atores envolvidos: Usuário
Fluxo principal: O usuário abre o aplicativo SeeForMe. O usuário seleciona a opção de registro como usuário. O usuário fornece as informações solicitadas, como nome, endereço de e-mail e outras informações relevantes. O usuário concorda com os termos e condições do serviço. O usuário concluir o registro.
Conclusão: O usuário concluiu com sucesso o registro como usuário no SeeForMe

Ministério da Saúde
FIOCRUZ
Fundação Oswaldo Cruz
Instituto Oswaldo Cruz
Laboratório de Comunicação Celular
Pós-Graduação *Stricto sensu* em Ensino em Biociências e Saúde

Caso de Uso: Solicitar Assistência Visual (UC-07)
Atores envolvidos: Usuário
Fluxo principal: O usuário abre o aplicativo SeeForMe O usuário seleciona a opção de solicitar assistência visual. O usuário descreve a tarefa ou a necessidade de assistência visual. O aplicativo busca por voluntários disponíveis e compatíveis com as necessidades do usuário. O usuário aguarda a conexão com um voluntário. O aplicativo estabelece uma chamada de vídeo em tempo real entre o usuário e o voluntário. O usuário e o voluntário se comunicam para que o voluntário possa fornecer a assistência visual necessária. Após receber a assistência necessária, o usuário pode encerrar a chamada.
Conclusão: O usuário solicitou com sucesso assistência visual através do SeeForMe e recebeu a ajuda necessária. A chamada de vídeo em tempo real foi encerrada com sucesso.

10. Diagrama de atividade

Figura 5 - Diagrama de atividade



Fonte: Material do autor

11. Diagrama de Classes

Os diagramas de classes são uma das estruturas mais importantes na orientação a objetos e na Linguagem Unificada de Modelagem (UML). Eles descrevem a estrutura estática de um sistema mostrando suas classes, seus atributos e os relacionamentos entre elas. As classes são representadas como retângulos divididos em três partes:

11.1 Partes do Diagrama de Classe

Nome da Classe:

A parte superior do retângulo contém o nome da classe.

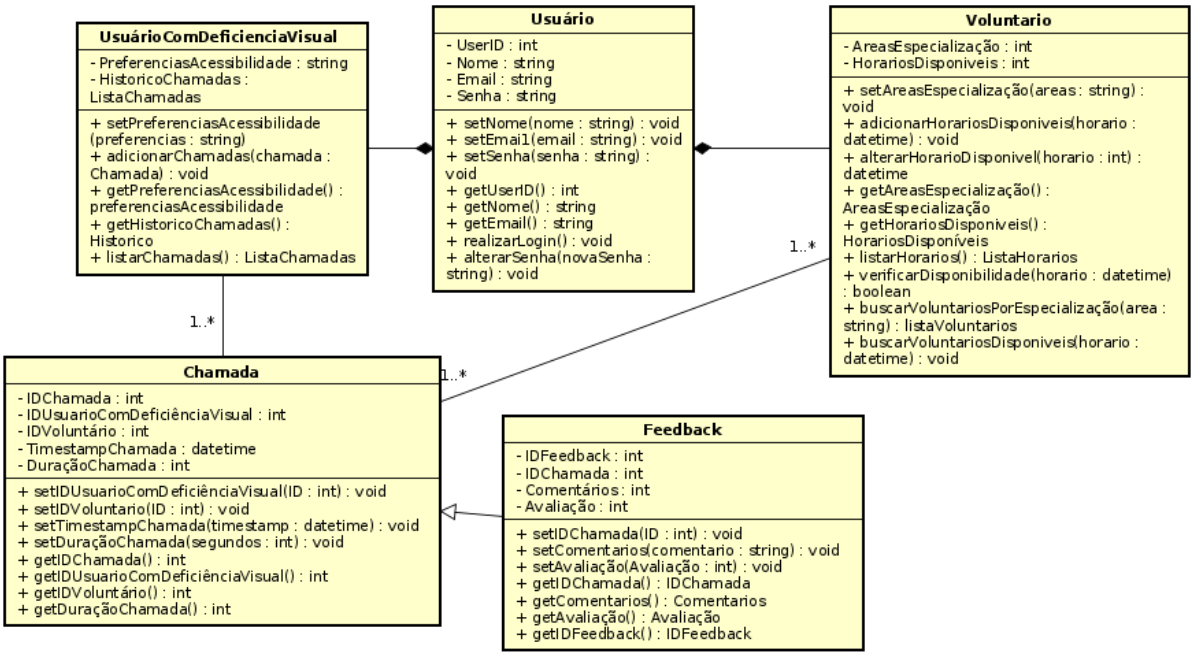
Atributos da Classe

A parte do meio do retângulo lista os atributos da classe. Os atributos são características ou propriedades que uma classe possui.

Métodos da Classe

A parte inferior do retângulo lista os métodos (ou operações) da classe. Os métodos são funções que definem o comportamento da classe.

Figura 6 - Diagrama de classes



Fonte: Material do Autor

11.2 Descrição do diagrama de classes

Classe Usuário: Esta seria a superclasse, com atributos como:

- UserID (único para cada usuário)
- Nome
- Email
- Senha

Classe UsuárioComDeficienciaVisual: Subclasse de Usuário, com atributos adicionais como:

- PreferenciasAcessibilidade (como ajustes de tamanho de texto, contraste de cores, etc.)

- HistoricoChamadas (uma lista de chamadas realizadas)

Classe Voluntario: Outra subclasse de Usuário, com atributos adicionais:

- AreasEspecialização (áreas em que o voluntário pode fornecer ajuda, como leitura, navegação, etc.)
- HorariosDisponiveis (os tempos em que o voluntário está disponível para ajudar)

Classe Chamada: Representa uma sessão de chamada única entre um UsuárioComDeficienciaVisual e um Voluntario. Atributos incluiriam:

- IDChamada (único para cada chamada)
- IDUsuarioComDeficienciaVisual
- IDVoluntario
- TimestampChamada (quando a chamada foi realizada)
- DuraçãoChamada (quanto tempo a chamada durou)

Classe Feedback: Representa o feedback dado após uma chamada. Atributos incluiriam:

- IDFeedback (único para cada feedback)
- IDChamada (associado a uma chamada específica)
- Comentários (textuais)
- Avaliacao (pontuação dada pelo usuário à chamada)

Quanto à relação entre as classes:

- ❖ UsuárioComDeficienciaVisual e Voluntário são subclasses de Usuário, o que significa que eles herdam todos os atributos e funcionalidades da classe Usuário, mas também possuem seus próprios atributos e funcionalidades específicos.
- ❖ UsuárioComDeficienciaVisual e Chamada têm uma relação de um para muitos, onde um UsuárioComDeficienciaVisual pode iniciar várias Chamadas.
- ❖ Voluntário e Chamada também têm uma relação de um para muitos, onde um Voluntário pode participar de várias Chamadas.

- Chamada e Feedback têm uma relação de um para um, onde cada Chamada pode ter um Feedback associado a ela.

12. O modelo de entidade-relacionamento (ER)

O modelo de entidade-relacionamento (ER) para o SeeForMe, refletindo as classes e suas interações no sistema, assim como a descrição que fizemos para o diagrama de classes. Aqui está uma descrição textual:

Entidade Usuário

A entidade Usuário seria a entidade principal que possui atributos como ID do Usuário, Nome, Email e Senha.

Entidade UsuárioComDeficienciaVisual

Esta entidade seria uma especialização da entidade Usuário, acrescentando atributos como PreferenciasAcessibilidade e HistoricoChamadas.

Entidade Voluntário

Esta entidade também seria uma especialização da entidade Usuário, acrescentando atributos como AreasEspecialização e HorariosDisponiveis.

Entidade Chamada

Esta é uma entidade separada que possui atributos como IDChamada, IDUsuarioComDeficienciaVisual, IDVoluntario, TimestampChamada e DuraçãoChamada. Ela estabelece uma relação "realiza" com UsuárioComDeficienciaVisual e uma relação "atende" com Voluntário.

Entidade Feedback

Esta é outra entidade separada com atributos como IDFeedback, IDChamada, Comentários e Avaliação. Ela estabelece uma relação "é associado a" com a entidade Chamada.

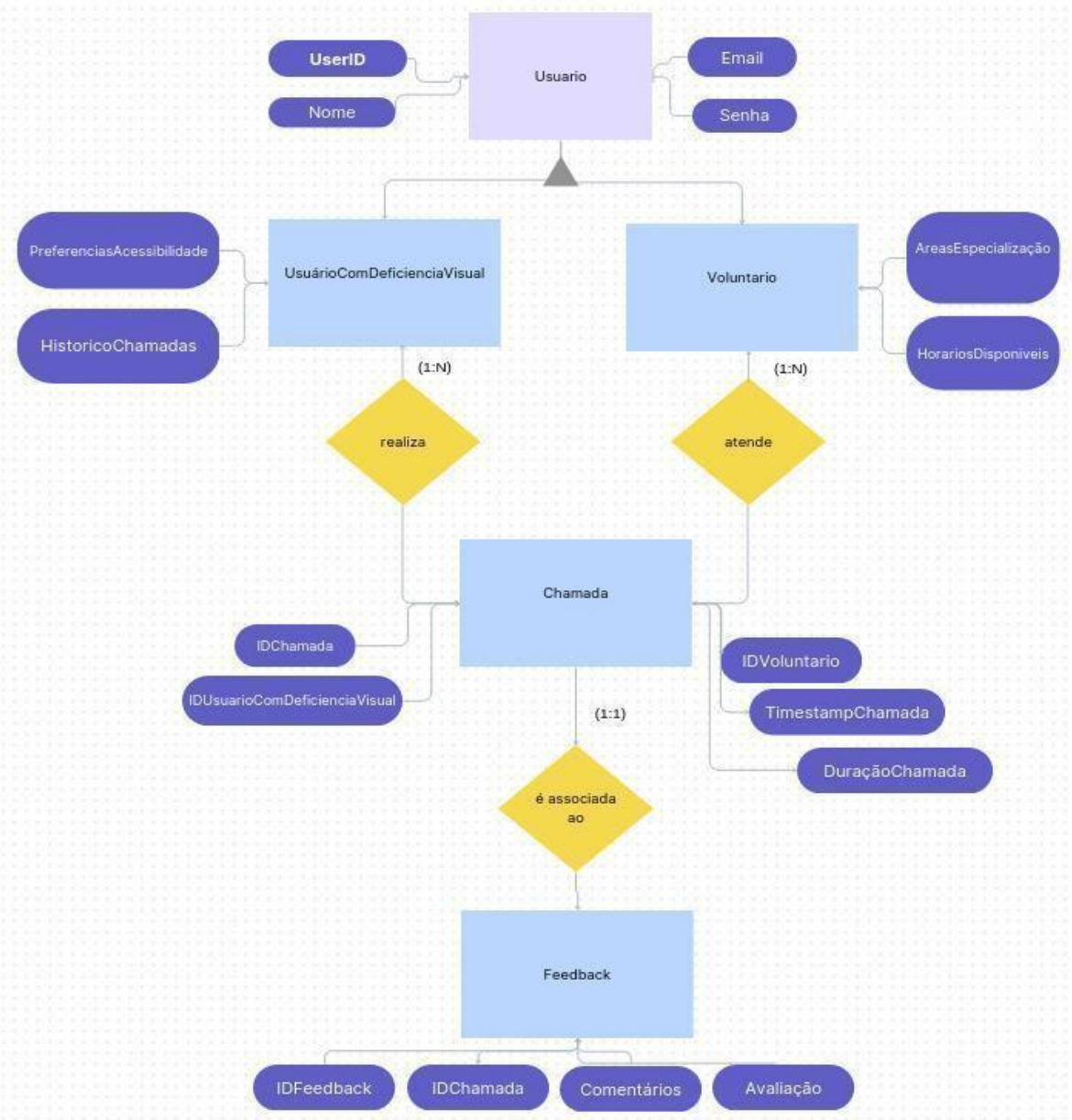
Nas relações, devemos também especificar a cardinalidade:

- A relação entre UsuárioComDeficienciaVisual e Chamada é de um para muitos

(1:N), pois um único usuário com deficiência visual pode realizar várias chamadas.

- A relação entre Voluntário e Chamada também é de um para muitos (1:N), já que um voluntário pode atender várias chamadas.
- A relação entre Chamada e Feedback é de um para um (1:1), já que cada chamada pode ter um feedback associado a ela.

Figura 7 - Modelo ER(Entidade Relacionamento)

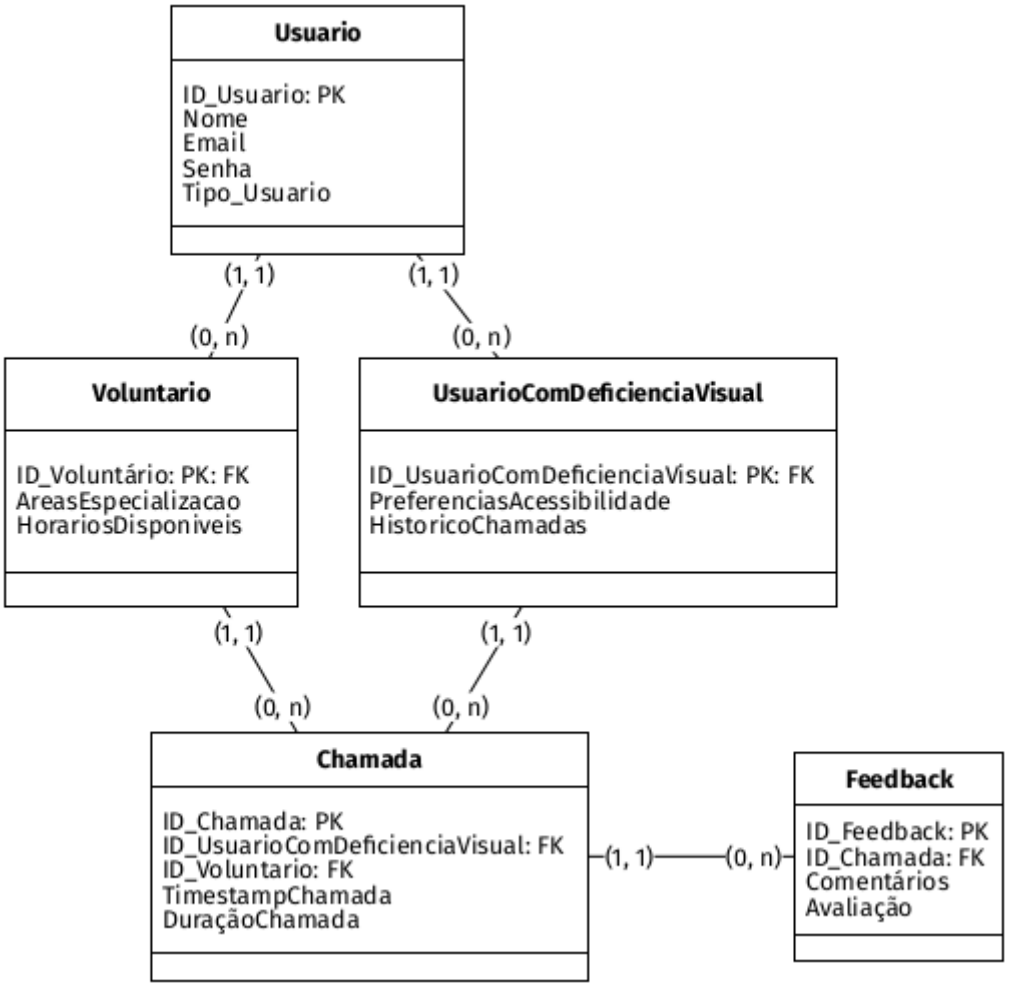


Fonte: Material do Autor

13. Estrutura do banco de dados

Para o sistema SeeForMe, a estrutura do banco de dados poderia refletir o Modelo de Entidade-Relacionamento que discutimos anteriormente. Aqui está um exemplo de como as tabelas no banco de dados poderiam ser estruturadas:

Figura 8 - Modelo Lógico



Fonte: Material do Autor

Tabela Usuário

- ID_Usuario (Chave primária)
- Nome
- Email
- Senha
- Tipo_Usuario (Este campo poderia indicar se o usuário é um "UsuárioComDeficienciaVisual" ou um "Voluntário")

Tabela UsuárioComDeficienciaVisual

- ID_UsuarioComDeficienciaVisual (Chave primária, estrangeira referenciando ID_Usuario)
- PreferenciasAcessibilidade
- HistoricoChamadas

Tabela Voluntário

- ID_Voluntario (Chave primária, estrangeira referenciando ID_Usuario)
- AreasEspecializacao
- HorariosDisponiveis

Tabela Chamada

- ID_Chamada (Chave primária)
- ID_UsuarioComDeficienciaVisual(Chave estrangeira referenciando ID_UsuarioComDeficienciaVisual)
- ID_Voluntario (Chave estrangeira referenciando ID_Voluntario)
- TimestampChamada
- DuraçãoChamada

Tabela Feedback

- ID_Feedback (Chave primária)
- ID_Chamada (Chave estrangeira referenciando ID_Chamada)
- Comentários
- Avaliação

14. Dicionário de Dados

Um dicionário de dados é uma documentação que lista todas as entidades do banco de dados, seus atributos (ou campos), e uma descrição detalhada de cada um. Veja como poderia ser um dicionário de dados para o sistema SeeForMe, considerando as tabelas descritas anteriormente:

Tabela Usuário

- ID_Usuario: Identificador único para cada usuário (Chave primária)
- Nome: Nome completo do usuário
- Email: Endereço de email do usuário
- Senha: Hash da senha do usuário
- Tipo_Usuario: Indica se o usuário é um "UsuárioComDeficienciaVisual" ou um "Voluntário"

Tabela UsuárioComDeficienciaVisual

- ID_UsuarioComDeficienciaVisual: Identificador do usuário com deficiência visual (Chave primária, estrangeira referenciando ID_Usuario)
- PreferenciasAcessibilidade: Preferências de acessibilidade do usuário com deficiência visual
- HistoricoChamadas: Registro de chamadas realizadas pelo usuário com deficiência visual

Tabela Voluntário

- ID_Voluntario: Identificador do voluntário (Chave primária, estrangeira referenciando ID_Usuario)
- AreasEspecializacao: Áreas de conhecimento que o voluntário pode ajudar
- HorariosDisponiveis: Horários em que o voluntário está disponível para atender chamadas

Tabela Chamada

- ID_Chamada: Identificador único para cada chamada (Chave primária)
- ID_UsuarioComDeficienciaVisual: Identificador do usuário com deficiência visual que iniciou a chamada (Chave estrangeira referenciando ID_UsuarioComDeficienciaVisual)
- ID_Voluntario: Identificador do voluntário que atendeu a chamada (Chave estrangeira referenciando ID_Voluntario)
- TimestampChamada: Horário em que a chamada foi iniciada
- DuraçãoChamada: Duração total da chamada

Tabela Feedback

- ID_Feedback: Identificador único para cada feedback (Chave primária)
- ID_Chamada: Identificador da chamada à qual o feedback está associado (Chave estrangeira referenciando ID_Chamada)
- Comentários: Comentários deixados pelo usuário com deficiência visual sobre a chamada
- Avaliação: Avaliação da chamada dada pelo usuário com deficiência visual.

15. GitHub

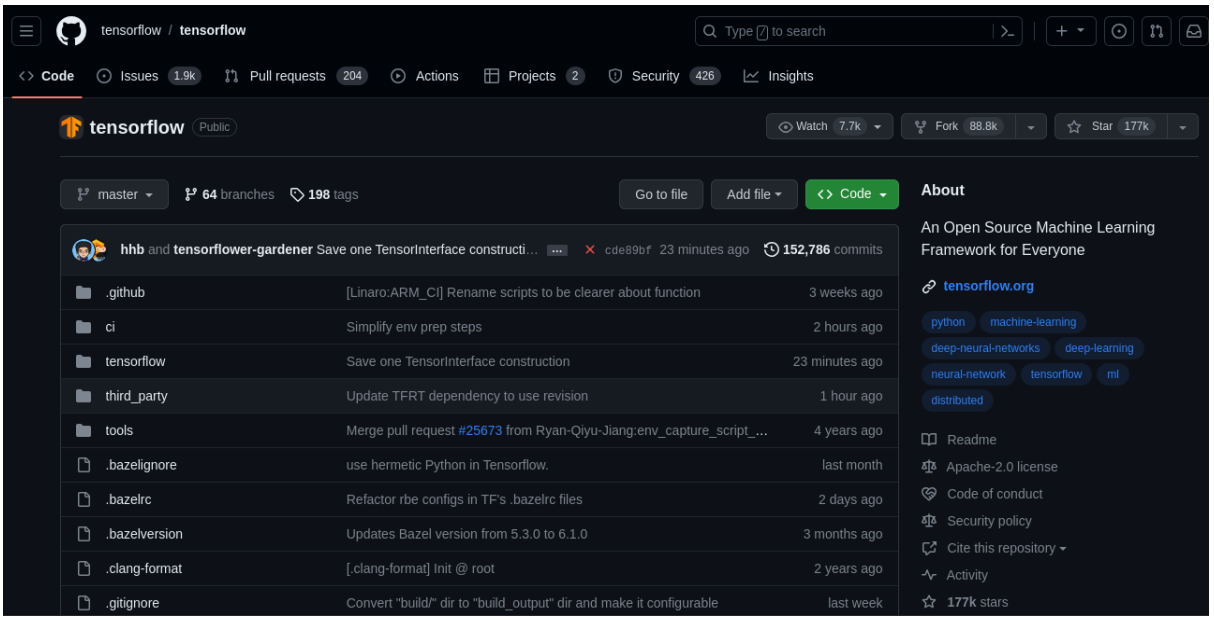
O GitHub é uma plataforma de hospedagem de código-fonte e um serviço de controle de versão baseado no Git. Foi fundado em 2008 por Chris Wanstrath, P. J. Hyett, Tom Preston-Werner e Scott Chacon. O GitHub oferece todas as funcionalidades distribuídas de controle de versão e gerenciamento de código-fonte (SCM) do Git, bem como seus próprios recursos, como um sistema de rastreamento de bugs, gerenciamento de tarefas, controle de acesso e funções de equipe para cada projeto. Confira abaixo como o GitHub pode ser usado em projetos de software:

15.1 Controle de Versão e Colaboração:

O GitHub utiliza o Git, um sistema de controle de versão distribuído, permitindo que várias pessoas trabalhem em um projeto simultaneamente. O GitHub facilita a colaboração ao permitir que os desenvolvedores "façam fork" em repositórios (criando sua própria cópia) e, depois, submetam "pull requests" (solicitações de integração) com suas alterações, para que possam ser revisadas e potencialmente mescladas no projeto original.

Abaixo um exemplo de repositório do github, o do TensorFlow:

Figura 9 - Projeto TensorFlow no GitHub

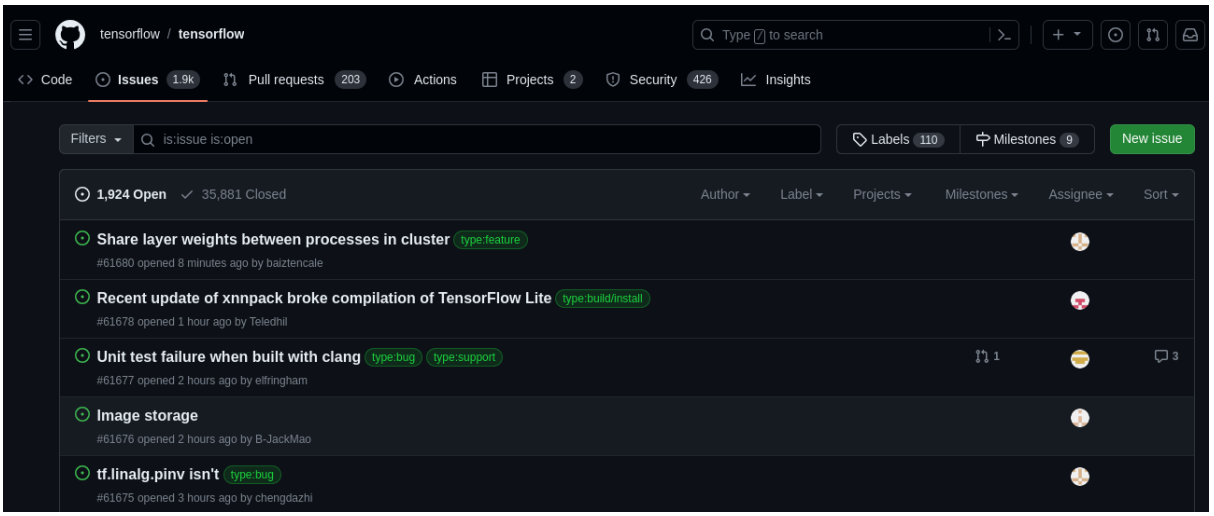


Fonte: Repositório do TensorFlow no GitHub

15.2 Rastreamento de Problemas:

O GitHub possui um sistema integrado de rastreamento de problemas que permite aos usuários e colaboradores criar problemas (issues) quando encontram um bug ou desejam sugerir um novo recurso.

Figura 10 - Aba Issues do projeto TensorFlow no GitHub

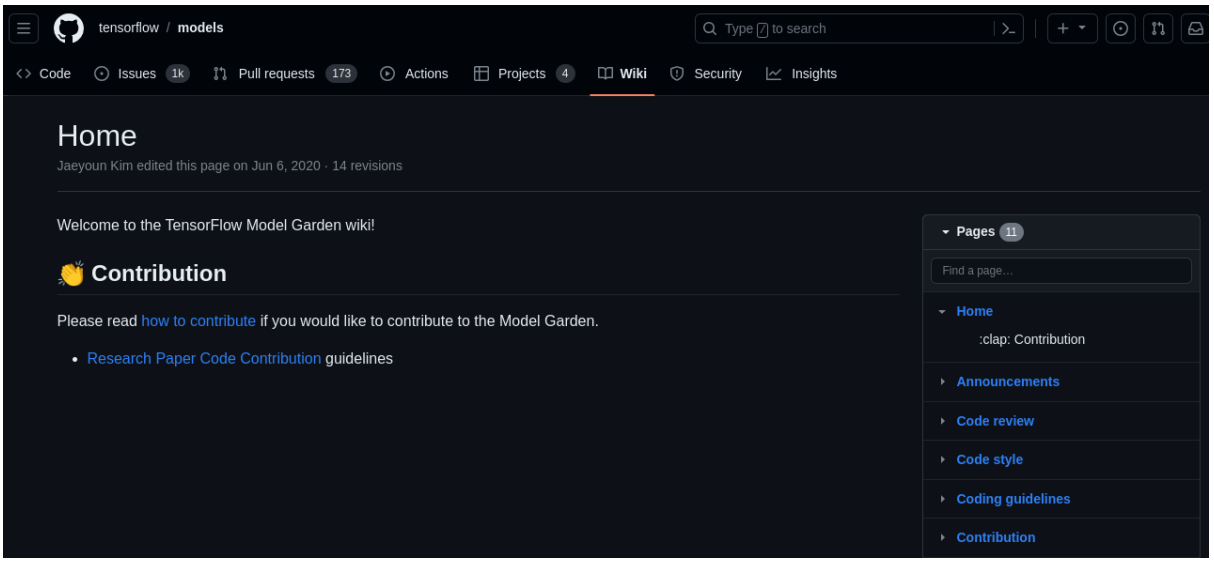


Fonte: Repositório do TensorFlow no GitHub

15.3 Documentação:

Usando a funcionalidade Wiki do GitHub, os projetos podem ter uma documentação abrangente e colaborativa. Além disso, o README do repositório é frequentemente usado para fornecer informações iniciais sobre o projeto.

Figura 11 - Aba Wiki do projeto TensorFlow no GitHub

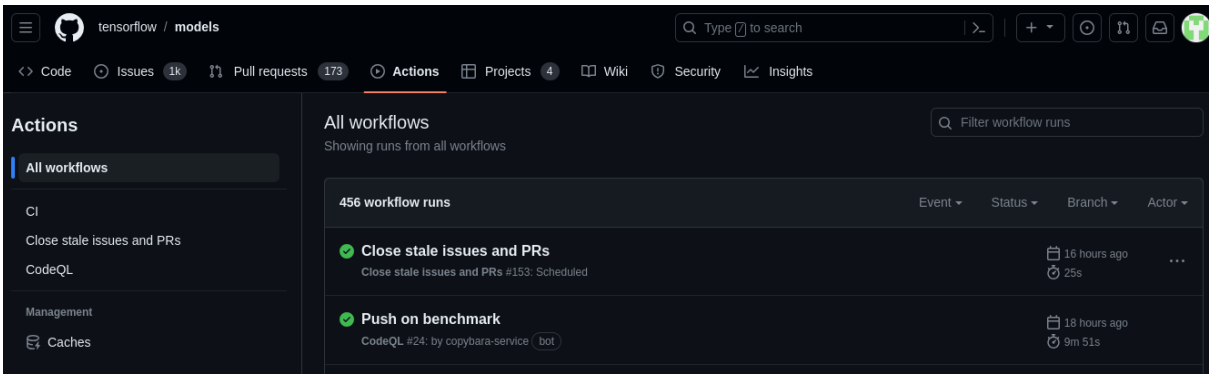


Fonte: Repositório do TensorFlow no GitHub

15.4 GitHub Actions:

Esta é uma funcionalidade para automação de CI/CD (Integração Contínua/Entrega Contínua). Permite que os desenvolvedores automatizem, personalizem e executem seus fluxos de trabalho de software no GitHub.

Figura 12 - Aba Actions do projeto TensorFlow no GitHub

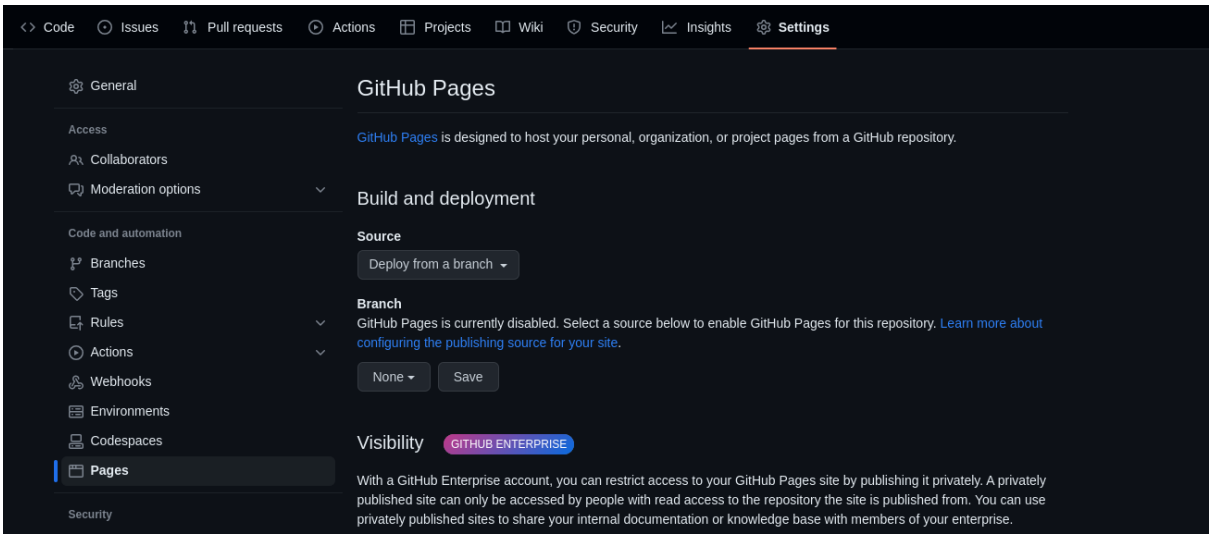


Fonte: Repositório do TensorFlow no GitHub

15.5 GitHub Pages:

É uma funcionalidade que permite aos usuários hospedar sites diretamente dos repositórios GitHub.

Figura 13 - Recurso GitHubPages

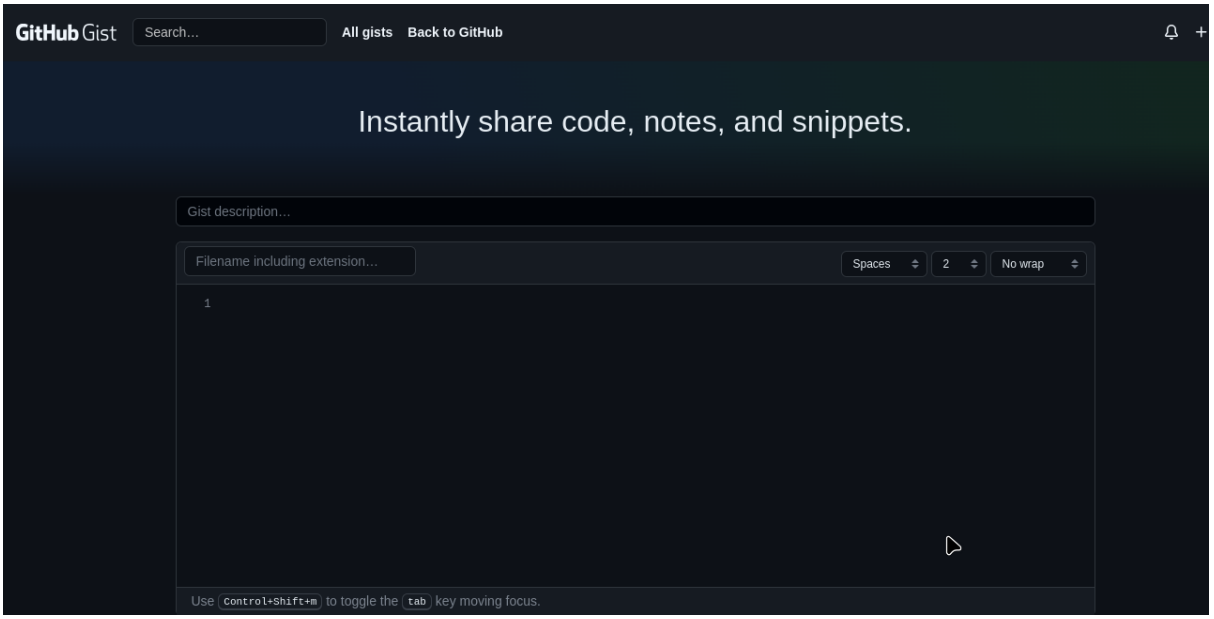


Fonte: Site do GitHub

15.6 Gists:

Para compartilhar trechos de código ou outros pequenos pedaços de informações, os usuários podem criar "gists".

Figura 14 - Recurso Gists do GitHub

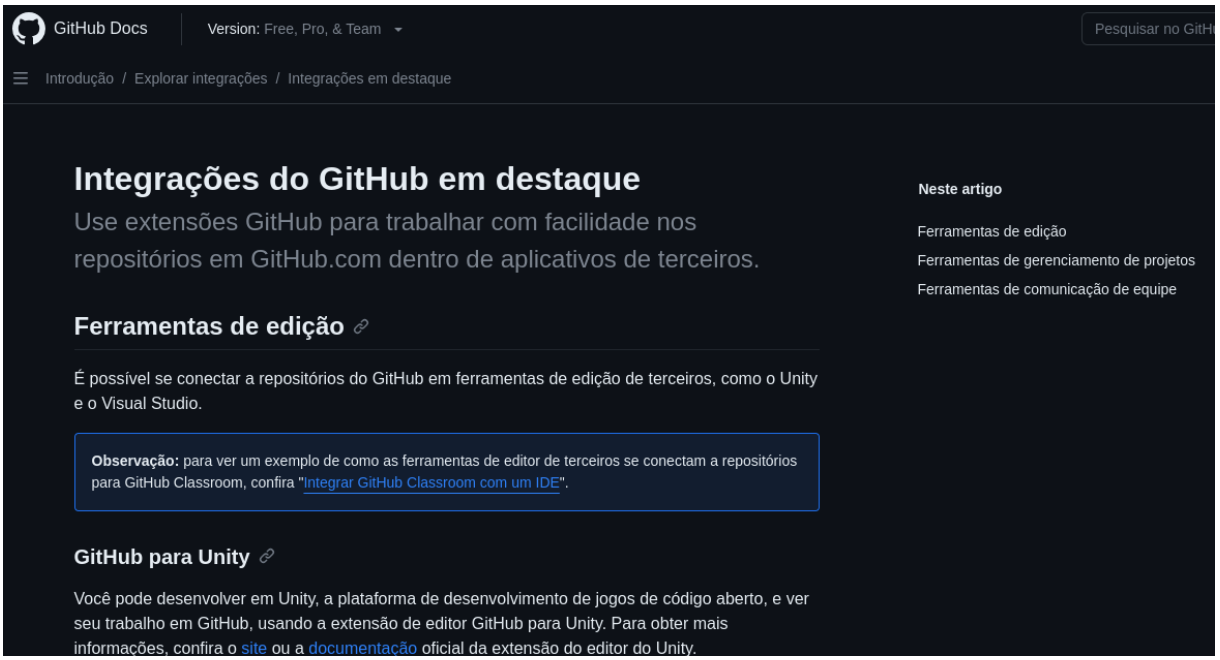


Fonte: Site do GitHub

15.7 Integração com Outras Ferramentas:

O GitHub pode ser integrado a uma variedade de ferramentas de desenvolvimento, como plataformas de CI, sistemas de gerenciamento de projetos e ferramentas de notificação.

Figura 15 - Recurso de integrações do GitHub

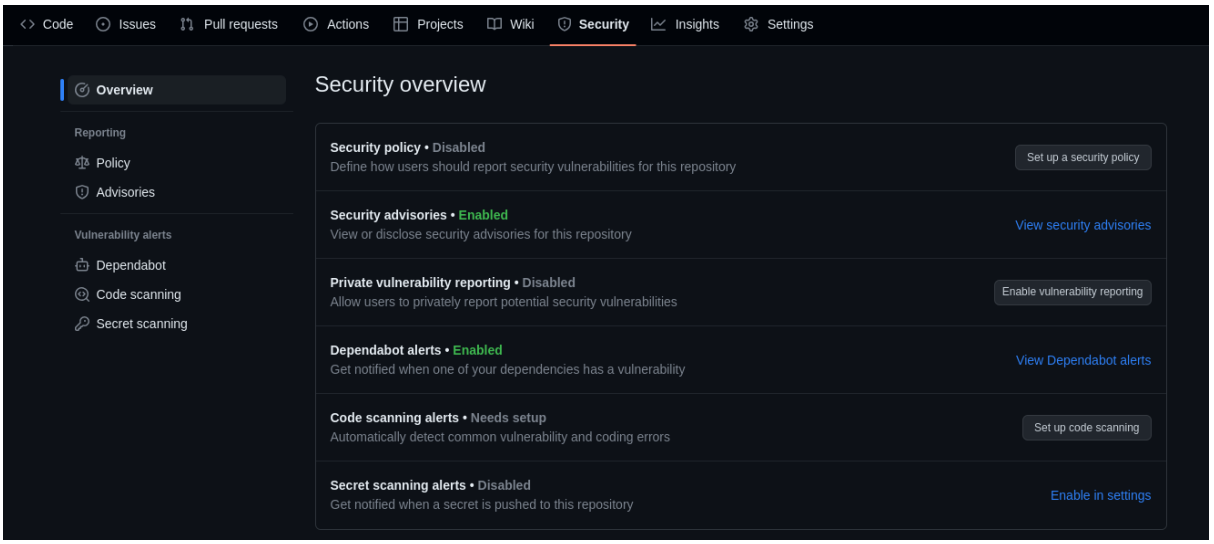


Fonte: Site do GitHub

15.8 Segurança:

O GitHub oferece várias funcionalidades de segurança, como autenticação de dois fatores, alertas de vulnerabilidade para dependências e a possibilidade de definir políticas de acesso ao repositório.

Figura 16 - Aba Security do projeto TensorFlow do GitHub

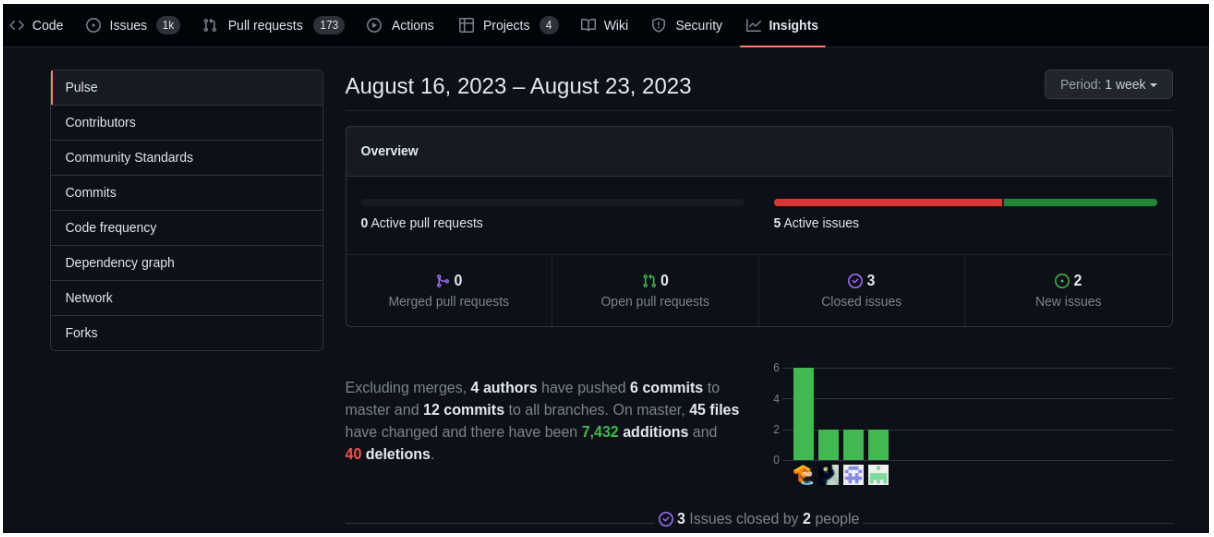


Fonte: Repositório do TensorFlow no GitHub

15.9 Insights e Análise:

Para ajudar a entender o desempenho e a atividade do repositório, o GitHub oferece insights, como análises de contribuição, atividade de pull request e muito mais.

Figura 17 - Aba Insights



Fonte: Repositório do TensorFlow no GitHub

16. Ferramentas e Tecnologias para Desenvolvimento de Software

Ferramentas e tecnologias comumente usadas em projetos de desenvolvimento de software:

16.1 Linguagens de Programação:

- Python
- Java
- JavaScript
- C#
- C++
- Ruby
- PHP
- Swift
- Kotlin
- TypeScript

16.2 Frameworks e Bibliotecas:

- Django (Python)
- Spring (Java)
- Node.js (JavaScript)
- React (JavaScript)
- Angular (JavaScript/TypeScript)
- Ruby on Rails (Ruby)
- Laravel (PHP)
- .NET (C#)
- Flask (Python)
- Express.js (JavaScript)

16.3 Controle de Versão:

- Git
- SVN (Subversion)
- Mercurial

16.4 Ambientes de Desenvolvimento Integrado (IDEs):

- Visual Studio Code
- IntelliJ IDEA
- Eclipse
- PyCharm
- Xcode
- Android Studio
- NetBeans
- Atom

16.5 Ferramentas de Gerenciamento de Projetos:

- Jira
- Trello
- Asana
- Microsoft Project

- Basecamp
- Monday.com

16.6 Bancos de Dados:

- Oracle
- MySQL
- PostgreSQL
- MongoDB
- Microsoft SQL Server
- SQLite

16.7 Testes e Qualidade de Software:

- Selenium
- JUnit
- Jest
- PyTest
- SonarQube
- Jira
- Jenkins
- Travis CI

16.8 Infraestrutura e Implantação:

- Docker
- Kubernetes
- Amazon Web Services (AWS)
- Microsoft Azure
- Google Cloud Platform (GCP)
- Heroku
- Apache Tomcat
- NGINX

17. Ferramentas e Tecnologias de Inteligência Artificial:

Ferramentas e tecnologias comumente usadas em projetos de desenvolvimento de

inteligência artificial:

17.1 Ferramentas de Inteligência Artificial:

- TensorFlow
- Apache SystemML
- Caffe
- Apache Mahout
- OpenNN
- Torch
- Neuroph
- Deeplearning4j

17.2 Software de Inteligência Artificial:

- Elephas
- Doxel
- Augmatrix
- Asana
- Writesonic
- Google Cloud
- ArcGIS
- Anyword

17.3 Tecnologias de desenvolvimento para Inteligência Artificial:

- Processamento de Linguagem Natural (PLN)
- Machine Learning
- Deep Learning
- Blockchain
- Tecnologia Imersiva

Essa lista é apenas uma amostra das muitas ferramentas e tecnologias disponíveis para desenvolvimento de software e inteligência artificial. A escolha das ferramentas e tecnologias depende dos requisitos do projeto, da preferência da equipe de desenvolvimento e da natureza da aplicação a ser desenvolvida. É importante lembrar que as tecnologias e ferramentas estão em constante evolução,

então é sempre bom estar atualizado e pesquisar as melhores opções para cada projeto específico.

18. Orientações

18.1 Defina claramente os objetivos do software:

Antes de iniciar o desenvolvimento, é essencial ter uma compreensão clara dos objetivos e requisitos do software. Isso ajudará a direcionar todas as etapas do processo de desenvolvimento.

18.2 Faça uma análise detalhada dos requisitos:

Realize uma análise completa e detalhada dos requisitos do software. Isso envolve entender as necessidades dos usuários, as funcionalidades necessárias e as restrições do sistema. Utilize técnicas como entrevistas, prototipagem e workshops para obter uma compreensão abrangente dos requisitos.

18.3 Planeje o desenvolvimento de forma iterativa:

Divida o projeto em iterações menores e desenvolva o software em etapas. Isso permite um desenvolvimento ágil, com feedback contínuo dos usuários e a possibilidade de ajustes ao longo do processo.

18.4 Utilize boas práticas de arquitetura de software:

Crie uma arquitetura sólida para o seu software, considerando a escalabilidade, a manutenibilidade e a reusabilidade. Utilize padrões de projeto, como MVC (Model-View-Controller), para separar as preocupações e facilitar a evolução do sistema.

18.5 Adote metodologias de desenvolvimento ágil:

Considere a utilização de metodologias ágeis, como Scrum ou Kanban, para gerenciar o desenvolvimento. Isso permite maior flexibilidade, colaboração eficiente e entrega incremental de funcionalidades.

18.6 Faça testes de forma automatizada e contínua:

Implemente uma estratégia de testes automatizados, incluindo testes unitários, integração e aceitação. Isso ajuda a garantir a qualidade do software e a identificar problemas precocemente.

18.7 Documente o software de forma clara e concisa:

Mantenha uma documentação atualizada, descrevendo a arquitetura, as decisões de design, os requisitos e as instruções de uso. Isso facilita a manutenção futura e o trabalho em equipe.

18.8 Priorize a usabilidade e a experiência do usuário:

Coloque o usuário no centro do desenvolvimento, projetando uma interface intuitiva e amigável. Realize testes de usabilidade para garantir que o software atenda às necessidades e expectativas dos usuários.

18.8 Mantenha-se atualizado com as tendências e tecnologias:

Esteja sempre atento às novas tendências e tecnologias na área de desenvolvimento de software. Mantenha-se atualizado e busque aprender e aplicar novas abordagens que possam melhorar o seu software.

18.9 Esteja aberto ao feedback e à evolução contínua:

Esteja preparado para receber feedback dos usuários e fazer melhorias constantes no software. A evolução contínua é essencial para atender às necessidades em constante mudança dos usuários e do mercado.

Lembre-se de que desenvolver um software de qualidade é um processo complexo e envolve diversos aspectos. Essas dicas podem ajudar a orientar o caminho, mas lembre-se de adaptá-las de acordo com as necessidades e particularidades do seu projeto.

19. Referências

BASS, L.; CLEMENTS, P.; KAZMAN, R. Software Architecture in Practice. 3. ed. Boston: Addison-Wesley, 2012.

BEIZER, B. Black-Box Testing: Techniques for Functional Testing of Software and Systems. New York: Wiley, 1995.

BROOKS, F. P. The Mythical Man-Month: Essays on Software Engineering. ed. comemorativa. Reading, Mass.: Addison-Wesley, 1995.

DUVALL, P.; MATYAS, S.; GLOVER, A. Continuous Integration: Improving Software Quality and Reducing Risk. Boston: Addison-Wesley Professional, 2007.

LEHMAN, M. M. Programs, Life Cycles, and Laws of Software Evolution. Proceedings of the IEEE, v. 68, n. 9, p. 1060-1076, 1980.

MYERS, G. J.; SANDLER, C.; BADGETT, T. The Art of Software Testing. 2. ed. Hoboken, NJ: Wiley, 2004.

PARNAS, D. L.; CLEMENTS, P. C. A Rational Design Process: How and Why to Fake It. IEEE Transactions on Software Engineering, SE-12, n. 2, p. 251-257, 1986.

PRESSMAN, R. S. Engenharia de Software: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2014.

SCHWABER, K.; SUTHERLAND, J. The Scrum Guide. [S.l.], 2017. Disponível em: <https://www.scrumguides.org>. Acesso em: [data de acesso].

SOMMERVILLE, I. Engenharia de Software. 9. ed. São Paulo: Pearson, 2011.

<Material do autor: 1,2,3,4,5,6,7 - Print da Modelagem feita no software Astah UML>

Figuras 8. 9, 10 e 11: Disponível em : <https://github.com/tensorflow/tensorflow>. Acesso em 23 Ago. 2023

Figura 12: Disponível em : <https://github.com>. Acesso em 23 Ago. 2023

Figura 13: Disponível em : <https://gist.github.com/>. Acesso em 23 Ago. 2023

Figura 14: Disponível em: <https://docs.github.com/pt/get-started/exploring-integrations/featured-github-integrations> Acesso em 23 Ago. 2023

Figura 15: Disponível em : <https://github.com/tensorflow/tensorflow>. Acesso em 23 Ago. 2023

Figura 16: Disponível em :<https://docs.github.com/pt/get-started/exploring-integrations/featured-github-integrations> Acesso em 24 Ago. 2023