CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

BRENO SALVADOR DE FREITAS

RELATÓRIO DE ESTÁGIO
LABORATÓRIO DE INTERFACE HOMEM-MÁQUINA – LIHM

Campina Grande
2021

BRENO SALVADOR DE FREITAS

RELATÓRIO DE ESTÁGIO
LABORATÓRIO DE INTERFACE HOMEM-MÁQUINA

Relatório de Estágio submetido à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Controle e Automação

Orientador: Prof. Danilo Freire de Souza Santos, D.Sc.

Campina Grande
2021

BRENO SALVADOR DE FREITAS

RELATÓRIO DE ESTÁGIO
LABORATÓRIO DE INTERFACE HOMEM-MÁQUINA

Relatório de Estágio submetido à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Controle e Automação

Aprovado em: / /

Professor Danilo Freire de Souza Santos, D.Sc.
Universidade Federal de Campina Grande
Orientador

Professor Jaidilson Jó da Silva, D.Sc.
Universidade Federal de Campina Grande
Convidado

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

UFCG	Universidade Federal de Campina Grande
RA	Realidade Aumentada
AR	*Augmented Reality*
VR	*Virtual Reality* (Realidade Virtual)
RV	Realidade Virtual
API	*Application Programming Interface* (Interface de programação de aplicações)
LIHM	Laboratório de Interface Homem-Máquina
UAEE	Unidade Acadêmica de Engenharia Elétrica
RAM	*Random Access Memory* (Memória de Acesso Aleatório)
MDN	*Mozilla Developer Network* (Rede do Desenvolvedor Mozilla)
WebGL	*Web Graphics Library* (Biblioteca Gráfica da Web)
HTML	*HyperText Markup Language* (Linguagem de Marcação de Hiper-Texto)
CSS	*Cascading Style Sheets* (Folhas de Estilo em Cascata)
3D	3 Dimensões
ID	Identificação/Identificador

SUMÁRIO

1 INTRODUÇÃO

É notável que algumas tecnologias antes vistas apenas no cinema em filmes de ficção científica, estão cada vez mais se tornando realidade. Tecnologias como o toque de tela, conversas por vídeo através de webcam e a biometria digital, hoje em dia encontram-se na palma das mãos implantados em dispositivos móveis inteligentes. Esse avanço tecnológico está possibilitando cada vez mais a evolução de tecnologias como a Realidade Aumentada (RA).

Para DE PACE et al. (2018), apesar de alguns anos atrás a falta de dispositivos de custo acessível ter sido a principal barreira para uma ampla adoção de aplicativos de RA, a ampla adoção de dispositivos móveis removeu essa limitação, já que eles apresentam todos os sensores e unidades de processamento necessários para desenvolver e implantar esses sistemas.

1.1 MOTIVAÇÃO

A RA é uma tecnologia que vem sendo utilizada não só para entretenimento como também em cenários da indústria. Essa tecnologia abre espaço para o desenvolvimento de sistemas que solucionam algumas dores presentes na indústria, seja em relação aos custos, seja em relação à produtividade. Esses sistemas focam em aumentar a eficiência e o rendimento na produção, assim como diminuir custos advindos da manutenção de máquinas feita na modalidade presencial.

Para fazer a manutenção presencial de uma máquina na indústria, muitas vezes um técnico de chão de fábrica precisa da assistência de um engenheiro especialista. Considerando que o engenheiro estará significativamente distante enquanto o técnico está diariamente operando a máquina, levaria tempo e dinheiro para que o engenheiro fosse efetuar a manutenção ou reparo da máquina pessoalmente, o que acarretaria um custo para a fábrica. Dessa forma, a manutenção remota pode mitigar esses custos consideravelmente.

Motivado pelas oportunidades de melhoria presentes na indústria e pelo fato de o estágio ter acontecido durante a pandemia do Covid-19, período em que o trabalho remoto

foi adotado significativamente, surgiu a ideia de desenvolver um sistema de RA para manutenção colaborativa remota de máquinas na indústria.

1.2 OBJETIVOS

O objetivo no estágio foi capacitar o aluno no desenvolvimento de software no âmbito da RA, com imersão numa metodologia ágil. Dessa forma, este relatório apresenta as atividades desenvolvidas pelo aluno Breno Salvador de Freitas durante o seu estágio no LIHM/UAEE (Laboratório de Interface Homem-Máquina) da UFCG, do dia 01/03/2021 ao dia 15/05/2021, totalizando uma carga horária de 184 horas. As atividades que foram planejadas para serem realizadas durante o estágio foram:

A. Imersão em atividades de gestão ágil de produtos e projetos;
B. Estudo dirigido em tecnologias de RA e dispositivos móveis;
C. Experimentação de ferramentas de desenvolvimento de interface com o usuário para realidade aumentada;
D. Desenvolvimento de protótipos de software;
E. Validação e experimentação de resultados.

As atividades planejadas que chegaram a ser, de fato, realizadas durante o estágio foram fundamentais para o desenvolvimento do protótipo do sistema de RA intitulado ARMain.

O nome ARMain advém da junção das palavras *Augmented Reality* e *Maintenance*, o que remete manutenção com uso de RA. Em resumo, esse sistema cria sessões remotas em que um operador local técnico consegue ser auxiliado por um operador remoto especialista durante uma manutenção de uma máquina num cenário da indústria através de RA, se comunicando à distância tanto através de conteúdo aumentado quanto por voz e utilizando apenas um dispositivo móvel como um *smartphone* ou *tablet*.

Como o estágio foi realizado em sua maioria de forma remota durante a pandemia do Covid-19, o documento também apresenta os prós e contras dessa modalidade em um período letivo tão curto. Além disso, é feita uma introdução às tecnologias utilizadas na implementação do sistema. E por fim, é feita uma discussão e uma análise dos resultados obtidos durante o período de estágio e uma conclusão considerando o que foi feito e o que pode ser feito em atividades futuras.

1.3 ESTRUTURA DO TRABALHO

O relatório conta com alguns capítulos posteriores à introdução. No Capítulo 2, é apresentado o laboratório do estágio e os prós e contras do estágio ter acontecido na maior parte do tempo na modalidade remota durante a pandemia do Covid-19. Posteriormente, no Capítulo 3, é apresentada a fundamentação teórica sobre as tecnologias usadas para a implementação do ARMain. Em seguida, no Capítulo 4, são detalhadas as atividades desenvolvidas durante o estágio. Já no Capítulo 5, é feita uma análise e uma discussão dos resultados obtidos durante o estágio. O relatório é finalizado no Capítulo 6, onde é feito o encaminhamento das conclusões e proposta de refinamentos para a continuação do desenvolvimento do ARMain.

2 O ESTÁGIO REMOTO NO LIHM

Localizado na Universidade Federal de Campina Grande – UFCG, no campus de Campina Grande, o LIHM é o Laboratório de Interface Homem-Máquina da Unidade Acadêmica de Engenharia Elétrica – UAEE.

O LIHM é um ambiente onde são realizadas pesquisas e desenvolvimentos pelos discentes da graduação e da pós-graduação, além dos experimentos relacionados às disciplinas que lá são ministradas. As atividades lá desenvolvidas são voltadas para sistemas de hardware e software que fazem a interface do usuário com sistemas utilizados em cenários de automação industrial.

O laboratório é composto por um espaço para avaliação de sistemas, sendo uma sala de testes e outra para observação e coleta de dados. Porém, como já mencionado anteriormente, o estágio foi realizado remotamente, visto que o período letivo ocorreu durante a pandemia do Covid-19. Pode-se dizer que tanto a pandemia quanto o trabalho remoto trouxeram algumas facilidades por um lado, mas dificuldades por outro.

A respeito das facilidades, o trabalho remoto:

- Reduziu as chances de contaminação do vírus já que o isolamento foi garantido em casa;
- Permitiu configurar uma agenda mais flexível.

Já quanto às dificuldades do trabalho remoto durante a pandemia, foram as seguintes:

- A falta de acesso aos computadores do laboratório para emulação de dispositivos móveis, o que fez com que o laptop do estagiário apresentasse travamentos algumas vezes ao rodar os emuladores que consomem muita memória RAM;
- A falta de um dispositivo móvel que suportasse RA para fazer testes do ARMain;
- O estagiário e sua família contraíram o vírus em tempos diferentes, dificultando a realização das atividades no tempo planejado.

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentadas as tecnologias que foram estudadas e utilizadas no desenvolvimento do ARMain para a web com funcionalidades de RA. O uso dessas tecnologias facilitou a implementação do código com a linguagem que o aluno está mais familiarizado (Javascript), além do versionamento do software durante as etapas concluídas no decorrer do estágio.

3.1 WEBXR

O WebXR[1] é uma API do *Immersive Web Community Group*, que tem colaboradores do Google, Microsoft, Mozilla e outros (Google Developers, 2021). Essa tecnologia fornece a funcionalidade necessária para trazer tanto a RA quanto a Realidade Virtual (RV) para a web. O "XR" do WebXR vem justamente da Realidade Cruzada, a qual representa a junção de AR e VR que do inglês são escritos *Augmented Reality* e *Vitual Reality*.

Como o estágio foi totalmente focado na RA, o único módulo do WebXR que é considerado neste documento é o de RA. Com esse módulo, é possível criar aplicações de RA para o uso em navegadores web compatíveis, permitindo que o conteúdo virtual seja alinhado com o ambiente real antes de ser exibido aos usuários da aplicação.

De acordo com o Mozilla, MDN (*Mozilla Developer Network*) (2021), o módulo de RA do WebXR ainda está em um estado de desenvolvimento inicial e ainda não é estável o suficiente para uso regular. Inclusive, ainda não foi incluída a sua documentação no MDN, mas ela será documentada assim que a especificação do módulo for estabelecida.

Para que o usuário final execute um sistema de RA desenvolvido com WebXR, é necessário não apenas usar um dispositivo móvel que suporta RA, mas que também tenha instalado um navegador compatível com a tecnologia. Já para que o desenvolvedor do mesmo sistema execute, ele pode tanto fazer isso em um dispositivo e navegador padrão

[1] Disponível em: <https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API>

para validação e experimentação do sistema, quanto ele pode fazer testes durante a implementação do código usando a extensão de emulador da API WebXR.

A equipe do Mozilla criou essa extensão compatível com os navegadores Firefox e Chrome, a qual emula o WebXR, simulando uma variedade de dispositivos compatíveis, como o Oculus Quest, Oculus Go, Samsung Gear e o Google Cardboard. Com a extensão instalada no navegador, o desenvolvedor pode usá-la a partir do painel de ferramentas do desenvolvedor, permitindo que ele controle a posição e orientação de um dispositivo móvel virtual que se encontra em uma cena virtual, os dois representando uma um dispositivo móvel real em um ambiente aumentado real, respectivamente. Dessa forma, o desenvolvedor consegue simular como a aplicação se comportaria em um ambiente real aumentado através de um dispositivo móvel real, ou de um dos dispositivos citados, de forma simples diretamente no computador onde o desenvolvedor programa.

Ainda de acordo com o MDN (2021), o WebXR não é uma tecnologia de renderização e nem fornece recursos para gerenciar dados 3D. Embora essa tecnologia gerencie o tempo, o agendamento e os vários pontos de vista relevantes ao desenhar a cena, ela não sabe como carregar, gerenciar, renderizar nem texturizar conteúdo de RA. Felizmente, o WebGL[2] e as várias estruturas e bibliotecas baseadas em WebGL estão disponíveis para tornar tudo isso possível.

3.2 THREE.JS

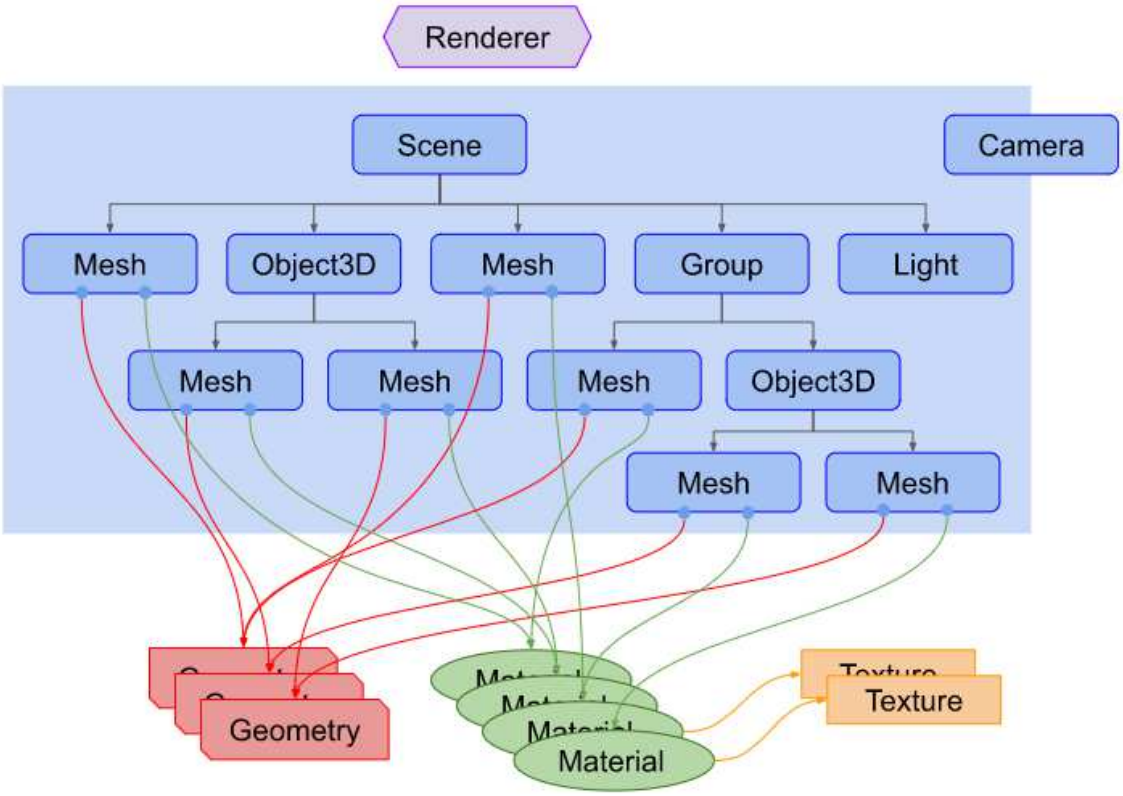
Uma das bibliotecas que usam WebGL é a Three.js. Ela foi criada por Ricardo Cabello e lançada no Github em abril de 2010. De acordo com a própria documentação do Three.js[3], o *Three.js Fundamentals* (2021), ela nada mais é que uma biblioteca Javascript e API que tenta facilitar ao máximo a obtenção de conteúdo 3D em uma página da web, criando computação gráfica 3D usando WebGL e exibindo-as.

A Three.js complementa muito bem o WebXR, pois ela lida com cenas, luzes, sombras, materiais, texturas, matemática 3D e todas as coisas que teriam que ser programadas se fosse usar WebGL diretamente.

Para entender melhor como funciona o desenvolvimento com a biblioteca, recomenda-se observar a Figura 1.

[2] Disponível em: <https://www.khronos.org/webgl/>
[3] Disponível em: <https://threejs.org/>

Figura 1. Representação de uma pequena aplicação com Three.js.



Fonte: (Three.js Fundamentals, 2021).

A Figura 1 representa um diagrama de uma estrutura de uma pequena aplicação feita com Three.js. Para desenvolver uma aplicação com essa ferramenta, é necessário criar uma renderizador que é o principal objeto. Além dele, também é necessário criar uma câmera e uma cena que são passados para o renderizador para que ele renderize a parte da cena 3D que está dentro do tronco da câmera como uma imagem 2D para uma tela chamada canvas.

Dentro da cena, são instanciados alguns objetos de classes como:

- *Mesh*: classe que representa objetos baseados em malha poligonal triangular;
- Object3D: classe base para a maioria dos objetos 3D. Fornece um conjunto de propriedades e métodos para manipular objetos no espaço 3D;
- *Group*: são quase idênticos aos Object3D;
- *Light*: classe base abstrata para luzes. Em resumo, representa a luz dentro da cena;
- *Camera*: classe base abstrata para câmeras. Em resumo, é a câmera que captura todos os objetos que se encontram dentro da cena.

Como pode ser visto na Figura 1, esses objetos são filhos do objeto *Scene* que permite a configuração de o que e onde é renderizado pelo Three.js, como os objetos

listados acima. Isto posto, pode-se observar que os últimos elementos são da classe Mesh. Cada objeto *Mesh* tem uma geometria como esfera, cubo, plano ou outra, e um material que representa sua cor, brilho e textura.

Todo esse contexto da biblioteca Three.js acoplado ao WebXR permite uma imersão bastante interessante com RA, onde a cena renderizada é o ambiente real aumentado, a câmera é um recurso do dispositivo móvel do usuário e os objetos virtuais mencionados são posicionados pelo usuário no ambiente aumentado de interesse.

3.3 GIT

Git[4] é um sistema de controle de versões distribuído que foi inicialmente projetado por Linus Torvalds em 2005 para o desenvolvimento do *kelnel* do sistema operacional Linux. Posteriormente, ele foi usado no desenvolvimento de outros softwares e também para registro de histórico de edições de arquivos em geral.

Atualmente, o Git é usado para desenvolver tanto projetos pequenos como muito grandes, com rapidez e eficiência. Desde pequenos até grandes times de desenvolvedores necessitam de uma ferramenta de versionamento de projeto, para que eles consigam trabalhar no mesmo arquivo ou código simultaneamente sem que haja conflitos nas alterações. Certamente, pode-se afirmar que essa tecnologia é um requisito de conhecimento para a extrema maioria dos engenheiros e desenvolvedores de software.

Existem plataformas de hospedagem de código-fonte e arquivos com controle de versão usando o Git. O Github[5], o qual foi usado durante o estágio, é uma delas. Para hospedar os arquivos de um software no Github, basta criar um repositório remoto e armazená-los nele. Após isso, o time de desenvolvedores pode clonar estes arquivos e transferir o clone para repositórios locais em seus computadores, possibilitando que alterem as mesmas linhas de código do software simultaneamente sem que haja conflito um com o outro. O que impede o conflito entre essas alterações é justamente o versionamento com o recurso de ramificação do Git.

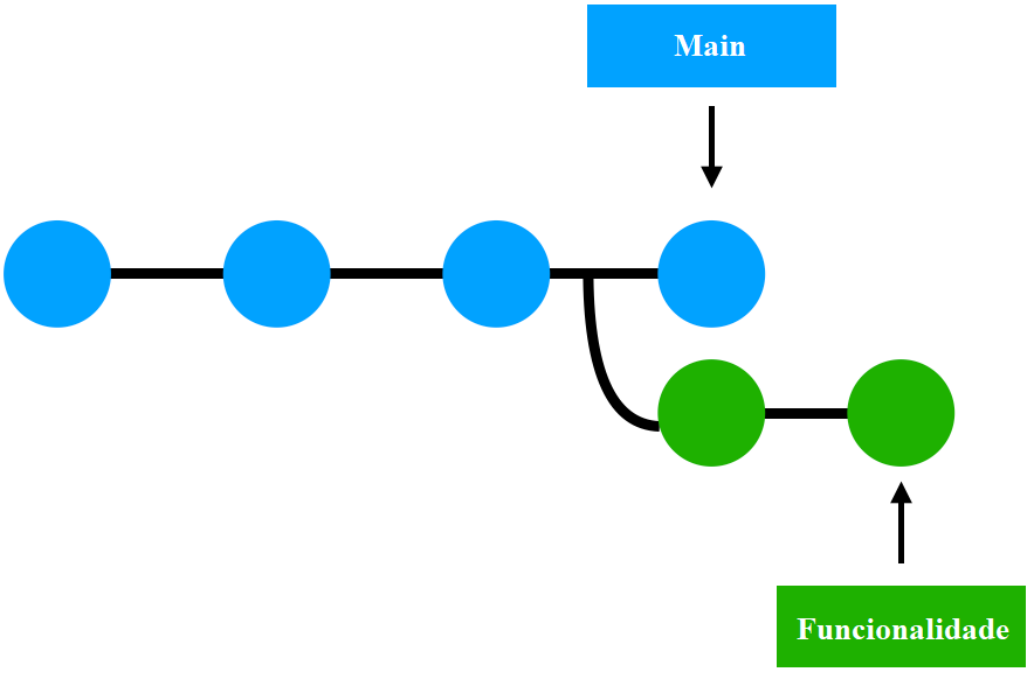
Esse recurso é composto por ramificações onde a principal é chamada de *Main*. Nessa ramificação é onde fica o código principal do software. Para fazer alterações no código do software, é recomendado que os desenvolvedores criem outras ramificações

[4] Disponível em: <https://git-scm.com/>
[5] Disponível em: <https://github.com/Brenosalv/ARMain>

originadas da *Main* justamente para proteger o código principal. Assim, eles podem implementar novas funcionalidades com o código-fonte sem que corra o risco de quebrar a integridade do original, o qual está funcionando perfeitamente.

Como pode ser visto na Figura 2, as estruturas azuis compõem a ramificação *Main* e a verde é uma ramificação chamada "Funcionalidade" que um desenvolvedor criou para implementar uma nova funcionalidade do software. Caso o código dessa funcionalidade esteja bem implementado e revisado, a ramificação "Funcionalidade" pode ser mesclada com a *Main*, fazendo com que se torne apenas um código e resulte em uma nova versão do software. Dessa forma, versões do software vão sendo criadas e gerenciadas, o que configura a razão pela qual o Git é tão usado.

Figura 2. Diagrama de recurso de ramificações do Git.



Fonte: Adaptado de (dyclassroom.com, 2021).

Esse processo descrito no parágrafo anterior representa a principal motivação de usar o Git no desenvolvimento de software, porém há muitas outras ações que podem ser tomadas no versionamento e em um repositório de um projeto de software. O Git possui diversos comandos para que todas as ações relativas ao versionamento sejam executadas como o "git push" para enviar conteúdo do repositório local para o repositório remoto ou o "git merge" para mesclar ramificações. Esses comandos e os outros podem ser executados no terminal do sistema operacional e são de fácil entendimento.

4 ATIVIDADES DESENVOLVIDAS

Cada atividade desenvolvida durante o estágio teve como finalidade a implementação do software ARMain adotando uma metodologia ágil, a qual é apresentada a seguir. Posteriormente, são detalhadas as atividades que foram realizadas pelo aluno.

4.1 METODOLOGIAS ÁGEIS E SCRUM

Sabe-se que para executar um projeto de software, é necessário adotar uma metodologia do início ao fim e por isso não seria diferente para o desenvolvimento do software ARMain. A metodologia adotada durante o estágio foi uma metodologia ágil, a qual tem algumas vantagens em relação às metodologias tradicionais.

A diferença das metodologias ágeis para as tradicionais é o enfoque nas pessoas e não em processos ou algoritmos, fazendo-as gastar menos tempo com documentação e mais tempo com a implementação do software (SOARES, 2004). Além disso, as metodologias ágeis promovem maior flexibilidade e simplicidade na execução das tarefas. Dessa forma, a adoção de uma metodologia ágil promoveu a execução das atividades do estágio de forma, de fato, ágil.

A metodologia ágil escolhida para o projeto foi o Scrum, que é um framework de gerenciamento de projetos, usado na organização e no desenvolvimento ágil de projetos. É uma metodologia amplamente utilizada por equipes de desenvolvimento de software atualmente e tem feedback bastante positivo de seus usuários.

Para fazer uso do Scrum, foi utilizado o Taiga[6], o qual é um sistema de gerenciamento de projetos gratuito. Sua interface gráfica promove uma boa usabilidade, o que diminui a curva de aprendizado ao gerenciar o projeto.

No Scrum, há três papeis que podem ser assumidos: *product owner*, *scrum master* e desenvolvedor. Como havia apenas dois colaboradores, o orientador ficou com o papel tanto de *product owner* como o de *scrum master*, enquanto o estagiário ficou com o papel de desenvolvedor. Dessa forma, o orientador ficou responsável por criar e gerenciar as

[6] Disponível em: <https://tree.taiga.io/project/danilosantos-ar-maintenance>

atividades durante todo o projeto do ARMain, enquanto o estagiário ficou responsável por desenvolver o ARMain até o fim do estágio.

Para executar as atividades durante o estágio, o Scrum tem como base 4 reuniões. No estágio, foram adotados apenas três tipos de reunião: A de planejamento de *sprint*, a de retrospectiva de sprint e a de revisão de sprint. Isso aconteceu porque havia apenas duas pessoas no projeto e por isso foi possível simplificar não fazendo as reuniões diárias.

A primeira reunião foi a de planejamento de *sprint*, onde foi configurado o *backlog*, que é uma lista priorizada com tudo que deve ser desenvolvido pelo estagiário durante todo o estágio.

Após a primeira reunião para o planejamento de *sprint*, há cada 15 dias aconteceram as três reuniões supracitadas em um mesmo horário: A de retrospectiva de *sprint,* onde é feita uma retrospecção do que foi feito durante a sprint passada e é verificado se há pontos do processo que podem ser melhorados; logo após a retrospectiva, já era feita a reunião de revisão da *sprint* passada, a qual serve para revisar o que foi feito durante a *sprint*, verificar se está de acordo com os requisitos e se necessário atualizar o *backlog*; e por fim era realizada a reunião de planejamento da próxima *sprint*.

A reunião que não foi adotada foi a *daily scrum*, que acontecem diariamente para dar *feedback* do que foi feito no dia anterior e planejar o que vai ser feito no dia presente de acordo com a sprint atual. Como o trabalho realizado foi feito individualmente, dentro do arcabouço do scrum não faz sentido realizar esse tipo de reunião.

Além das reuniões, há cada 7 dias o estagiário teve que dar feedback do andamento das tarefas que estavam sendo desenvolvidas na presente *sprint*. Dessa forma, as atividades foram sendo desenvolvidas com a flexibilidade para possíveis mudanças nas tarefas a serem executadas para concluir as atividades planejadas, a cada 15 dias até o fim do estágio, se adequando às condições de cada ciclo.

4.2 REALIZAÇÃO DAS ATIVIDADES

Relembrando quais atividades foram planejadas para serem realizadas durante o estágio, são elas:

A. Imersão em atividades de gestão ágil de produtos e projetos;
B. Estudo dirigido em tecnologias de RA e dispositivos móveis;

C. Experimentação de ferramentas de desenvolvimento de interface com o usuário para realidade aumentada;
D. Desenvolvimento de protótipos de software;
E. Validação e experimentação de resultados.

A atividade A começou a ser realizada a partir do momento que se iniciou o desenvolvimento do ARMain com o Scrum, já que o objetivo era imergir na gestão ágil de um projeto.

Com isso, deu-se início à atividade B, realizando um estudo dirigido em tecnologias de RA e dispositivos móveis. No início, buscou-se uma ferramenta útil para desenvolver o ARMain em formato de aplicativo nativo para o sistema operacional Android e foi encontrado o ARCore.

O ARCore é um kit de desenvolvimento de software desenvolvido pelo Google usado para desenvolvimento de aplicativos nativos que usam RA em suas funcionalidades. Para implementar o aplicativo com o ARCore, utiliza-se Java ou Kotlin como linguagem de programação. De fato, parece muito conveniente com os objetivos do estágio, porém a não familiaridade com essas linguagens de programação apresentou uma maior curva de aprendizado do que o esperado. Como o estagiário tinha mais familiaridade com a linguagem de programação Javascript e desenvolvimento web, buscou-se uma tecnologia que permitisse o desenvolvimento de um sistema de RA de acordo com tal contexto.

Assim, foram estudadas as tecnologias WebXR e Three.js, as quais permitiram o desenvolvimento do ARMain para web e a implementação de objetos virtuais 3D animados em um ambiente aumentado, respectivamente. Dessa forma, a curva de aprendizado do estagiário diminuiu com a maior familiaridade com as tecnologias adotadas e a facilidade de acesso do usuário aumentou, já que basta acessar o software através de um navegador compatível em um dispositivo móvel.

Para implementar o *front-end* das páginas web que servem para entrar e criar uma sessão para manutenção remota, foi utilizada a biblioteca React.js que tem como base o uso das tecnologias HTML, CSS e Javascript para estruturar, estilizar e tratar a lógica da aplicação, respectivamente. A partir de então, iniciou-se a atividade C, ou seja, a fase de experimentação das ferramentas utilizadas para o desenvolvimento da interface com o usuário através de RA. A interface do ARMain foi implementada de forma que ao entrar em uma sessão, o usuário imerge em um ambiente aumentado com o auxílio do WebXR e do Three.js juntamente com a câmera do dispositivo móvel. Esse ambiente aumentado

fica visível na tela do dispositivo móvel dos operadores local e remoto, sendo a área real onde o operador local e a máquina se encontram na indústria, porém com o benefício de posicionamento de âncoras. Essas âncoras são objetos virtuais 3D que o operador remoto pode posicionar na máquina ou ao redor dela para auxiliar o operador local na manutenção ou reparo. Isto posto, a experimentação das tecnologias adotadas levou à conclusão que poderia ser desenvolvido um protótipo do software.

Foi então que a atividade D foi iniciada. Para desenvolver um protótipo, foi utilizada a ferramenta de versionamento de software Git para armazenar o código do protótipo em um repositório localizado no Github. A cada *sprint*, o código sofreu alterações e foi atualizado no repositório. Dessa forma, tanto o aluno quanto o orientador tinham acesso ao código atualizado do protótipo do ARMain.

Além disso, foi usada a extensão para navegador WebXR API *Emulator* para que pudesse fazer testes no ARMain conforme fosse evoluindo no desenvolvimento sem que houvesse a necessidade de um dispositivo móvel. Certamente, um dispositivo móvel real seria mais preciso nos testes, mas a não disponibilidade de um fez com que o emulador ajudasse bastante no desenvolvimento do protótipo.

Por causa das dificuldades decorrentes da pandemia do Covid-19, a atividade E não chegou a ser realizada até o fim da vigência do estágio. Portanto, não foi feita uma validação e experimentação do ARMain em um ambiente real com um dispositivo móvel real para que fossem verificados os seus resultados.

5 DISCUSSÃO E ANÁLISE DOS RESULTADOS

Durante e após desenvolver as atividades, foram obtidos alguns resultados, tanto da implementação do código do protótipo do ARMain, quanto da metodologia ágil.

5.1 SCRUM

A prática com o Scrum durante todo o estágio trouxe como resultado um aprendizado muito importante para a formação de um engenheiro que pretende atuar no mercado de trabalho, como é o caso do estagiário. A razão disso é que o Scrum é uma das metodologias ágeis mais empregadas no mercado de desenvolvimento de software. Certamente, ter tido uma experiência bem-sucedida com a metodologia será um grande diferencial profissional.

Sendo assim, o Scrum foi praticado com êxito, completando cinco sprints no total. Cada uma das reuniões de retrospectiva e planejamento das sprints realizadas a cada 15 dias, foi fundamental para que as atividades fossem cumpridas com os devidos ajustes no decorrer do estágio.

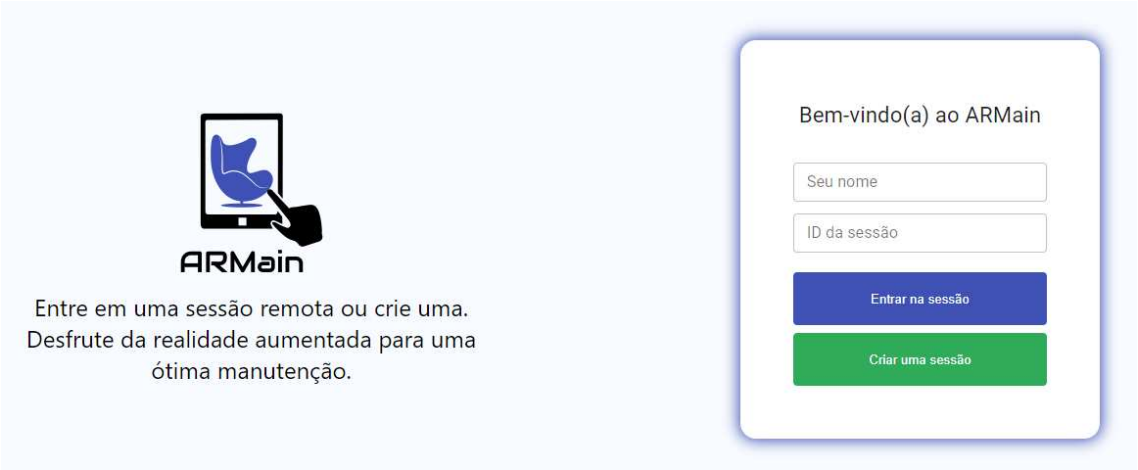
Quanto ao Taiga, pode-se afirmar que ele foi bastante útil para auxiliar na aplicação do Scrum, sendo possível observar as tarefas concluídas que constam tanto no backlog quanto em cada sprint finalizada.

Foi notado que a metodologia ágil é de fácil compreensão, não acarretou custos já que a equipe era pequena e motivou constantemente o aluno, pois a divisão do projeto em pequenas tarefas fez com que a produtividade e o comprometimento fossem intensificados.

5.2 ARMAIN

Com o *front-end* do protótipo do ARMain sendo implementado com a biblioteca React.js, foram desenvolvidas as páginas usadas para criar uma sessão de RA e para entrar em uma sessão já criada previamente. Nas Figuras 3 e 4, pode-se ver as duas interfaces prontas.

Figura 3. Representação da página para entrar em uma sessão de RA.



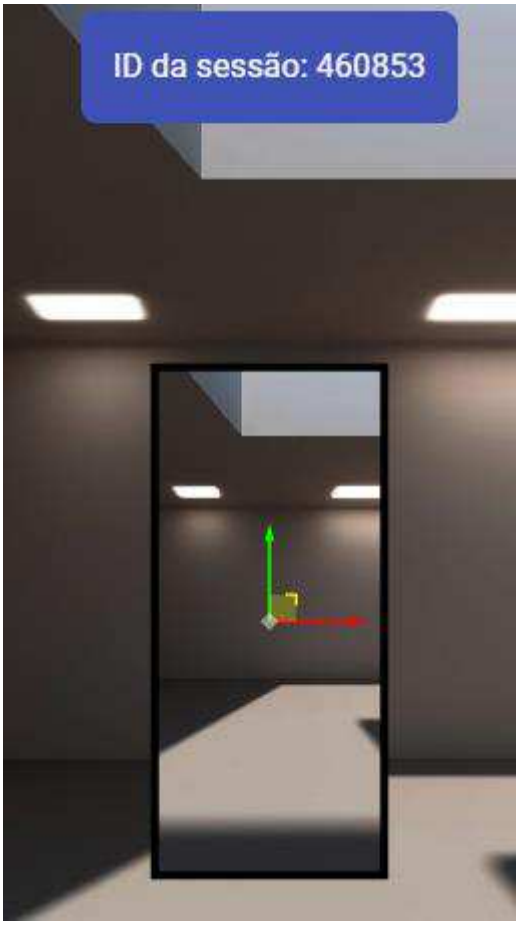
Fonte: Autoria própria.

Figura 4. Representação da página para criar uma sessão de RA.

Fonte: Autoria própria.

Apesar do protótipo não ter sido testado em um ambiente real por causa da falta de um dispositivo móvel que suportasse RA, alguns testes foram feitos usando o emulador WebXR API. Com ele, foi possível simular uma cena de um ambiente aumentado e um dispositivo móvel usando as funcionalidades do WebXR e do Three.js. A Figura 5 mostra como fica a simulação da cena quando um usuário está dentro de uma sessão durante uma manutenção remota.

Figura 5. Representação da sessão de RA.



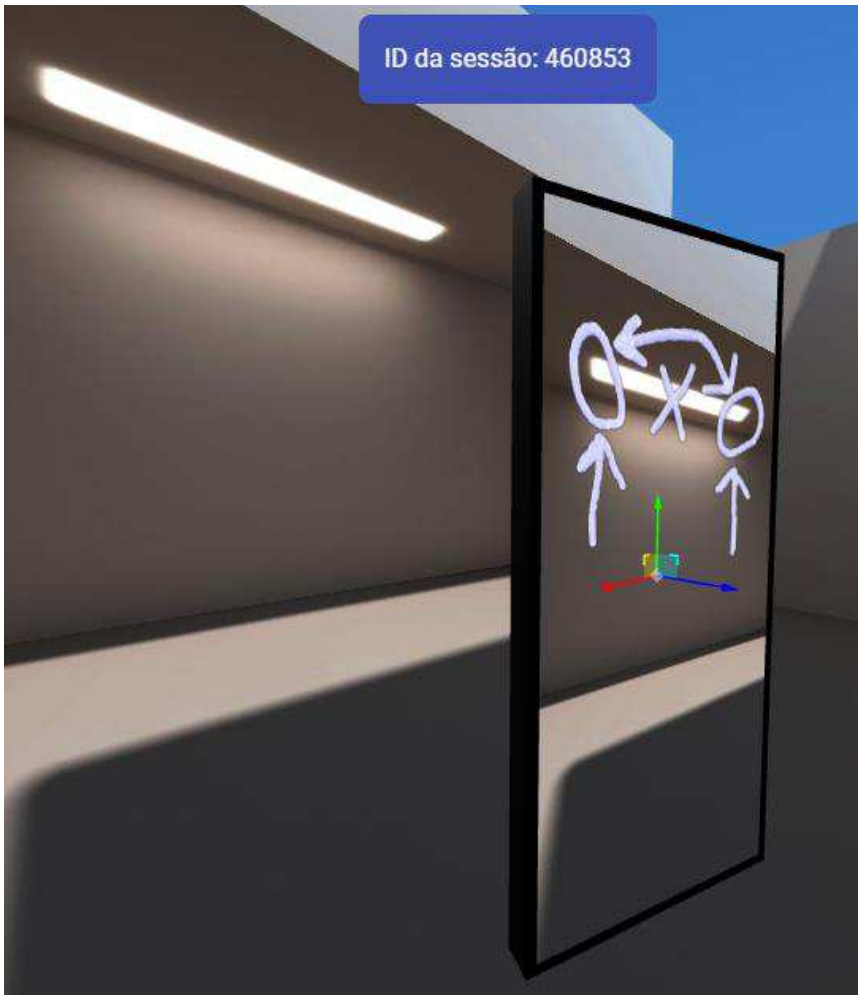
Fonte: Autoria própria.

Ao entrar numa sessão ou criar uma, é gerado um ID de seis dígitos para identificar a sessão, o qual fica à mostra nas telas dos dispositivos móveis dos operadores local e remoto. No emulador, o ID fica fora da tela do dispositivo móvel virtual, o qual se encontra no centro da Figura 5. Porém, foi verificado que o conteúdo do ID fica dentro da tela quando em um dispositivo móvel real, permitindo que o operador remoto que já está dentro da sessão compartilhe o ID com o operador local para que ele também entre na mesma sessão. Já as setas vermelha e verde, elas são interativas para que, durante testes, o desenvolvedor possa deslocar o dispositivo móvel virtual para verificar de diferentes ângulos se o conteúdo aumentado foi posicionado no local desejado ou simplesmente para simular o operador local movimentando a câmera do dispositivo móvel durante testes.

Durante a sessão, o operador remoto deve orientar o operador local a posicionar a câmera do dispositivo móvel a aproximadamente 1m de distância da máquina para que ele consiga desenhar qualquer coisa em cima da máquina ou ao redor dela, posicionando

uma âncora no formato de um tubo flexível criado com o Three.js. A Figura 6 mostra um exemplo de uso do desenho livre com o ARMain. Como é apenas um simulador de um ambiente real aumentado, foi usado uma luminária virtual no lugar de uma máquina real somente para exemplificar. É visto que o operador remoto tem total liberdade para desenhar setas, círculos e outros formatos para indicar o que ele deseja enquanto auxilia o operador local. Após posicionar a âncora, o operador local pode se movimentar livremente já que o desenho já foi posicionado e fica fixo onde foi posicionado.

Figura 6. Exemplo de interação com desenho livre.



Fonte: Autoria própria.

O ideal é que esse processo seja feito juntamente com comunicação por voz no próprio ARMain, porém essa funcionalidade não chegou a ser implementada e, por hora, foi considerada a necessidade de outra tecnologia para estabelecer uma comunicação por voz durante a manutenção. Não apenas essa funcionalidade como outras ficaram faltando, mas podem ser agregadas em possíveis desenvolvimentos futuros do protótipo.

6 CONSIDERAÇÕES FINAIS

A necessidade de realizar o estágio de forma remota, o advento da pandemia do Covid-19 e o período letivo tão curto acarretaram algumas dificuldades que impossibilitaram um desenvolvimento mais profundo do ARMain. Portanto, pode-se afirmar que o protótipo do software ainda precisa de uma funcionalidade para comunicação por voz, mais opções de âncoras e da implementação da transmissão de dados entre os operadores. Essas sugestões abrem caminho para trabalhos futuros com foco na melhoria do protótipo e na sua validação com testes de dois operadores durante uma manutenção colaborativa remota de uma máquina em um cenário industrial real utilizando dispositivos móveis reais.

No que diz respeito ao que foi, de fato, realizado durante o estágio, pode-se dizer que ele foi fundamental para a formação e o crescimento profissional do estagiário como engenheiro eletricista, tendo em vista a imersão em atividades práticas que levaram em consideração conhecimentos que foram adquiridos durante a graduação em Engenharia Elétrica, principalmente em relação ao desenvolvimento de software e ao gerenciamento de projeto.

Também pode-se afirmar que o desenvolvimento de software utilizando uma tecnologia ainda tão nova como a RA foi desafiador para o estagiário que não tinha familiaridade antes do estágio. Porém, certamente, saber lidar com novos desafios é algo que se espera de um engenheiro e por isso essa experiência foi enriquecedora.

Por fim, a necessidade de cumprimento de prazos e reuniões, as dificuldades decorrentes do contexto do estágio e as dificuldades encontradas nas atividades em si, acabaram estimulando o desenvolvimento de competências e maturidade necessárias para a inserção no mercado de trabalho, que é o principal foco do aluno que está prestes a concluir a graduação.

REFERÊNCIAS

DE PACE, F. et al. (2018). Augmented Reality in Industry 4.0. American Journal of Computer Science and Information Technology, Vol.6 No.1:17.

SOARES, M. S. Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software. Universidade Presidente Antônio Carlos, BR 482 Km 3, Gigante, Conselheiro Lafaiete, MG, Brasil, (2004).

Fundamentals of WebXR. **MDN Web Docs**, Mozilla, 2021. Disponível em: https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API/Fundamentals. Acesso em: 11 de maio de 2021.

WebXR. **Google Developers**, Google, 2021. Disponível em: https://developers.google.com/ar/develop/webxr. Acesso em: 11 de maio de 2021.