UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Pedro Germano Cembranel Etges

Desenvolvimento de um Gêmeo Digital para a planta MPS 4.0 do Laboratório de Automação Industrial da UFSC

Florianópolis
2023

Pedro Germano Cembranel Etges

Desenvolvimento de um Gêmeo Digital para a planta MPS 4.0 do Laboratório de Automação Industrial da UFSC

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.
Orientador: Prof. Ricardo José Rabelo, Dr.
Supervisor: Prof. Ricardo José Rabelo, Dr.

Florianópolis
2023

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Cembranel Etges, Pedro Germano
Desenvolvimento de um Gêmeo Digital para a planta MPS 4.0 do Laboratório de Automação Industrial da UFSC / Pedro Germano Cembranel Etges ; orientador, Ricardo José Rabelo, coorientador, Felipe Gomes Cabral, 2024.
96 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2024.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Gêmeos Digitais. I. Rabelo, Ricardo José. II. Gomes Cabral, Felipe. III. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. IV. Título.

Pedro Germano Cembranel Etges

Desenvolvimento de um Gêmeo Digital para a planta MPS 4.0 do Laboratório de Automação Industrial da UFSC

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis,26 de fevereiro de 2024.

Prof. Marcelo De Lellis Costa de Oliveira, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Ricardo José Rabelo, Dr.
Orientador
UFSC/CTC/DAS

Prof. Ricardo José Rabelo, Dr.
Supervisor
LAI/DAS/UFSC

Prof. Rodrigo Castelan Carlson, Dr.
Avaliador
UFSC/CTC/DAS

Prof. Eduardo Camponogara, Dr.
Presidente da Banca
UFSC/CTC/DAS

Este trabalho é dedicado aos meus colegas de classe e
aos meus queridos pais.

AGRADECIMENTOS

Agradecimentos aos professores Ricardo Rabelo, Públio e Felipe Cabral pelo auxílio no desenvolvimento deste projeto.

“Gêmeos digitais são modelos computacionais que emulam em tempo real o comportamento de entidades físicas para fins de observação, simulação, análise e tomada de decisão. Eles são uma das tecnologias habilitadoras para a Indústria 4.0, que visa integrar e otimizar os sistemas ciberfísicos em cadeias de valor e modelos de negócio baseados na Internet das Coisas, Serviços e Pessoas. (RABELO, 2020)

DECLARAÇÃO DE PUBLICIDADE

Florianópolis, 26 de fevereiro de 2024

Na condição de representante da UFSC-LAI na qual o presente trabalho foi realizado, declaro não haver ressalvas quanto ao aspecto de sigilo ou propriedade intelectual sobre as informações contidas neste documento, que impeçam a sua publicação por parte da Universidade Federal de Santa Catarina (UFSC) para acesso pelo público em geral, incluindo a sua disponibilização *online* no Repositório Institucional da Biblioteca Universitária da UFSC. Além disso, declaro ciência de que o autor, na condição de estudante da UFSC, é obrigado a depositar este documento, por se tratar de um Trabalho de Conclusão de Curso, no referido Repositório Institucional, em atendimento à Resolução Normativa n° 126/2019/CUn.

Por estar de acordo com esses termos, subscrevo-me abaixo.

ICP-Edu

Documento assinado digitalmente
Ricardo Jose Rabelo
Data: 16/02/2024 06:32:30-0300
CPF: ***.802.119-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Ricardo José Rabelo, Dr.
LAI/DAS/UFSC

RESUMO

Com o impulsionamento da indústria pela Indústria 4.0, a demanda por digitalização e tratamento de dados tem se intensificado, visando aprimorar o suporte às tomadas de decisão. Como resultado desse contexto, tecnologias habilitadoras têm ganhado destaque, e entre elas se destaca o Gêmeo Digital (GD). Essa abordagem envolve um modelo computacional que emula em tempo real o comportamento de uma entidade física, permitindo realizar observações, simulações, análises e tomadas de decisão com base nessas informações.A conversão de um equipamento industrial real no chão de fábrica em um Sistema Ciber-Físico (SCF) e sua virtualização efetiva como Gêmeo Digital (GD) são processos complexos e desafiadores. Essa transformação exige a integração meticulosa de diferentes sistemas, a interoperação de diversos protocolos de comunicação e formatos, além da análise cuidadosa de protocolos capazes de transmitir grandes volumes de dados em tempo real. Este projeto tem como intuito implementar um gêmeo digital que possa ser usado em atividades de ensino, pesquisa e treinamento, contemplando as novas estações de trabalho e o robô móvel instalados no LAI (Laboratório de Automação e Informática Industrial), junto a uma metodologia pesquisa-ação para o desenvolvimento do projeto. Para implementar o GD, foi utilizado o software *PlantSimulation* para criar a virtualização e interação entre os sistemas digitais e o SCF, junto de uma API online que permite obter informações de operações em tempo real e interagir com o SCF, além de um banco de dados em nuvem para habilitar a integração entre os sistemas desenvolvidos. O projeto resultou na implementação de um GD capaz de interagir com SCF, gerar análises e disponibilizar informações de operação em tempo real para o usuário.

Palavras-chave: Gêmeo Digital. Sistemas Ciber-físicos. Indústria 4.0.

ABSTRACT

With the boost of the industry by Industry 4.0, the demand for digitization and data processing has intensified, aiming to improve the support for decision making. As a result of this context, enabling technologies have gained prominence, and among them the Digital Twin (DT) stands out. This approach involves a computational model that emulates in real time the behavior of a physical entity, allowing to perform observations, simulations, analyses and decision making based on this information. The conversion of a real industrial equipment on the factory floor into a Cyber-Physical System (CPS) and its effective virtualization as a Digital Twin (DT) are complex and challenging processes. This transformation requires the meticulous integration of different systems, the interoperability of various communication protocols and formats, as well as the careful analysis of protocols capable of transmitting large volumes of data in real time. This project aims to implement a digital twin that can be used in teaching, research and training activities, contemplating the new workstations and the mobile robot installed at LAI (Laboratory of Automation and Industrial Informatics), together with an action-research methodology for the project development. To implement the DT, the software PlantSimulation was used to create the virtualization and interaction between the digital systems and the CPS, along with an online API that allows to obtain real-time operation information and interact with the CPS, as well as a cloud database to enable the integration between the systems developed. The project resulted in the implementation of a DT capable of interacting with CPS, generating analyses and providing real-time operation information to the user.

Keywords: Digital Twin. Cyber-Physical System. Industry 4.0.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A Indústria 4.0, também conhecida como a Quarta Revolução Industrial, representa uma nova era na indústria, marcada pela digitalização, interconectividade, automação e inovação. Este termo foi originalmente introduzido na Feira de Hanôver em 2011 e refere-se ao uso de tecnologias avançadas, incluindo Internet das Coisas (IoT), inteligência artificial (IA), aprendizado de máquina, análise de big data, computação em nuvem e sistemas ciberfísicos, para melhorar a eficiência e a eficácia dos processos de produção (KAGERMANN; WAHLSTER; HELBIG, 2013a).

O avanço da Indústria 4.0 tem sido reportado por diversas fontes de notícias, como a (INDUSTRY. . . , 2019), que ressalta o crescente impacto da digitalização na indústria global, e a (THE. . . , 2020), que destaca o papel da IA e do aprendizado de máquina na otimização dos processos industriais. A Indústria 4.0 tem o potencial de revolucionar a forma como os produtos são fabricados e os serviços são entregues, trazendo benefícios como maior eficiência, flexibilidade, personalização e qualidade, ao mesmo tempo em que apresenta desafios relacionados à segurança, privacidade, formação de competências e adaptação à mudança (SCHWAB, 2017).

Um dos elementos fundamentais para a concretização do modelo da Indústria 4.0 são os Sistemas Ciber-Físicos (SCFs). Conforme definido por (COLOMBO; KARNOUSKOS; KAYNAK, 2017), um SCF é um ambiente integrado que combina capacidades computacionais e físicas, como sensoriamento, comunicação e atuação, com ciclos de realimentação onde processos físicos afetam os processos computacionais e vice-versa. Os SCFs podem apresentar diferentes níveis de autonomia, inteligência e capacidade de comunicação em tempo real, como apontado por (LEE; BAGHERI; KAO, 2015).

Com o impulsionamento da indústria pela Indústria 4.0, a demanda por digitalização e tratamento de dados tem se intensificado, visando aprimorar o suporte às tomadas de decisão. Como resultado desse contexto, tecnologias habilitadoras têm ganhado destaque, e entre elas se destaca o Gêmeo Digital (GD), do inglês *Digital Twin*. Essa abordagem envolve um modelo computacional que emula em tempo real o comportamento de uma entidade física, permitindo realizar observações, simulações, análises e tomadas de decisão com base nessas informações (ALAM; SADDIK, 2017).

A conversão de um equipamento industrial real no chão de fábrica em um Sistema Ciber-Físico (SCF) e sua virtualização efetiva como Gêmeo Digital (GD) são processos complexos e desafiadores. Essa transformação exige a integração meticulosa de diferentes sistemas, a interoperação de diversos protocolos de comunicação e formatos, além da análise cuidadosa de protocolos capazes de transmitir grandes volumes de dados em tempo real. Além disso, o procedimento envolve a construção de *wrappers*, a síntese de informações provenientes de instrumentação, Internet das

Coisas (IoT) e redes industriais, entre outros aspectos (SILLER; ROMERO; RABELO, R. J., 2018).

1.1 PROBLEMA

Em 2020, no âmbito de um trabalho de mestrado no Programa de PósAutomação e Informática, foi desenvolvido um GD para a planta atual do LAI (Laboratório de Automação Industrial do DAS). O projeto envolve sete estações de trabalho da FESTO e respectivos CLPs Siemens série 1200, modelados como SCFs, além de utilizarem o protocolo OPC-DA para comunicação.

Recentemente, foram instaladas mais três novas estações de trabalho e um robô móvel, também da FESTO, com CLPs Siemens da série 1500, e o protocolo OPC-UA para comunicação. Porém, tais SCFs não estão abrangidos no GD desenvolvido, e possuem diferenças nos aspectos de comunicação devido aos protocolos utilizados serem diferentes.

O protocolo OPC-DA apresenta uma série de limitações em relação ao OPC-UA, sendo as principais:

- **Plataforma:** OPC-DA só pode ser usado no Windows OS como servidor ou cliente, enquanto o OPC-UA pode ser usado em qualquer sistema operacional que suporte UDDI, como Linux, MacOS, Android, iOS, etc;
- **Funcionalidade:** OPC-DA permite acessar apenas os dados atuais dos dispositivos conectados ao servidor OPC, não sendo capaz de gerar alarmes ou eventos históricos. Já o OPC-UA suporta recursos como eventos históricos, múltiplas hierarquias e métodos, comandos, e também permite acessar dados passados dos dispositivos conectados ao servidor OPC;
- **Segurança:** OPC-DA usa um mecanismo simples de autenticação baseado em credenciais do usuário ou do sistema operacional, não oferecendo proteção contra ataques externos ou internos ao servidor opcional. O OPC-UA usa um mecanismo mais complexo de autenticação baseado em certificados digitais ou tokens criptográficos e também oferece níveis diferentes de segurança para os diferentes tipos de serviços opcional;
- **Compatibilidade:** OPC-DA tem uma interface padronizada que define quais propriedades e funções podem ser usadas pelos clientes, facilitando a integração entre diferentes produtos que implementam essa interface. No entanto, isso também implica que cada produto precisa fornecer um driver específico para se comunicar com o servidor usando esse protocolo, o que pode ocasionar limitação da portabilidade do produto para outros sistemas operacionais ou plataformas.

Já em relação aos CLPs Siemens série 1200 e série 1500, as principais diferenças estão presentes nos aspectos de desempenho e flexibilidade. O CLP Siemens série 1500 pode processar mais dados por segundo e lidar com mais entradas e saídas simultâneas em relação ao CLP Siemens série 1200. Pode também se adaptar a mais aplicações, usando métodos e programas para executar tarefas específicas além de também poder se comunicar com outros dispositivos inteligentes usando protocolos como OPC-UA ou MQTT.

Este trabalho visa desenvolver e implementar GDs para as novas bancadas do LAI, seguindo como base os modelos já desenvolvidos em trabalhos anteriores, adaptando o GD para as diferenças existentes entre as bancadas, como o protocolo de comunicação OPC-UA e os CLPs Siemens série 1500.

1.2 OBJETIVO GERAL

O objetivo geral deste PFC é implementar um GD para contemplar as novas estações do LAI que possa ser usado em atividades de ensino, pesquisa e treinamento.

1.3 OBJETIVOS ESPECÍFICOS

Para a realização deste objetivo geral, e seguindo uma metodologia criada no DAS para o desenvolvimento de GD, os seguintes objetivos específicos são previstos:

- Entendimento das funcionalidades das três novas estações de trabalho e do robô móvel “Robotino”;
- Definição da estratégia e arquitetura de integração com os CLPs;
- Definição dos modelos de dados para armazenar as informações coletadas dos CLPs que deverão ser emuladas no ambiente do GD;
- Definição da estratégia e arquitetura de integração entre as estações/robô e o GD;
- Modelagem gráfica do GD;
- Definição das funcionalidades do GD;
- Implementação e avaliação do GD.

1.4 ENQUADRAMENTO METODOLÓGICO

(CERVO; BERVIAN; SILVA, R. d., 2007) destaca que a pesquisa é uma investigação que visa à obtenção de novos conhecimentos e à resolução de problemas práticos. (GIL, 2017) define a pesquisa como um procedimento formal, com métodos

científicos, que permite obter informações sistemáticas e válidas para responder a questões de estudo.

Neste estudo, optou-se por empregar a metodologia de Pesquisa-ação. Essa abordagem se caracteriza pela sua base empírica e aplicação prática, voltada para a realização de intervenções, o desenvolvimento e a instigação de mudanças em grupos, organizações e comunidades (GIL, 2017). A Pesquisa-ação emprega técnicas de pesquisa consolidadas com o objetivo de conceber ações direcionadas à melhoria das práticas de grupos específicos (TRIPP, 2005).

A metodologia pesquisa-ação foi escolhida devido à natureza deste trabalho e suas semelhanças com a metodologia de desenvolvimento ágil. A metodologia de desenvolvimento ágil é uma abordagem altamente flexível e colaborativa amplamente empregada na criação de software. Esta abordagem se concentra na entrega contínua de valor aos clientes, priorizando a adaptação às mudanças e fomentando a colaboração ativa entre as equipes envolvidas no projeto e será melhor abordada na Seção 3.1.6.

A Figura 1 apresenta as etapas da pesquisa-ação. O ciclo de etapas se inicia na etapa de Planejar/Pesquisar os aspectos de melhoria e soluções possíveis para o problema abordado. Concluída a primeira etapa, segue-se para etapa de Implementar melhorias planejadas. Após a implementação das melhorias, inicia-se a etapa de Descrever mudanças, e por fim, os resultados e ações são avaliados na etapa Avaliar, dando início a um novo ciclo de planejamento e pesquisa.

Os passos metodológicos predefinidos para o desenvolvimento deste projeto estão estreitamente alinhados aos objetivos específicos explicitados na Seção 1.3, sendo esses passos:

- A Leitura de dois trabalhos realizados no mesmo ambiente: a dissertação de mestrado desenvolvida na PósAutomação, e um PFC de desenvolvimento de um Chatbot para a mesma planta;
- A refamiliarização com o ambiente computacional do TIA Portal da Siemens, dentro do qual está o SCADA, supervisório, entre outros módulos, de gestão geral da planta;
- Estudo e testes com o protocolo OPC-UA;
- Estudo dos equipamentos da nova planta, seus CLPs, suas programações, e identificação das tags geradas por estes;
- Estudo e testes de implementação inicial de aquisição de dados das tags para armazenamento em um repositório de dados digital;
- Seleção e testes do ambiente gráfico / ferramenta de modelagem do Gêmeo Digital (GD);

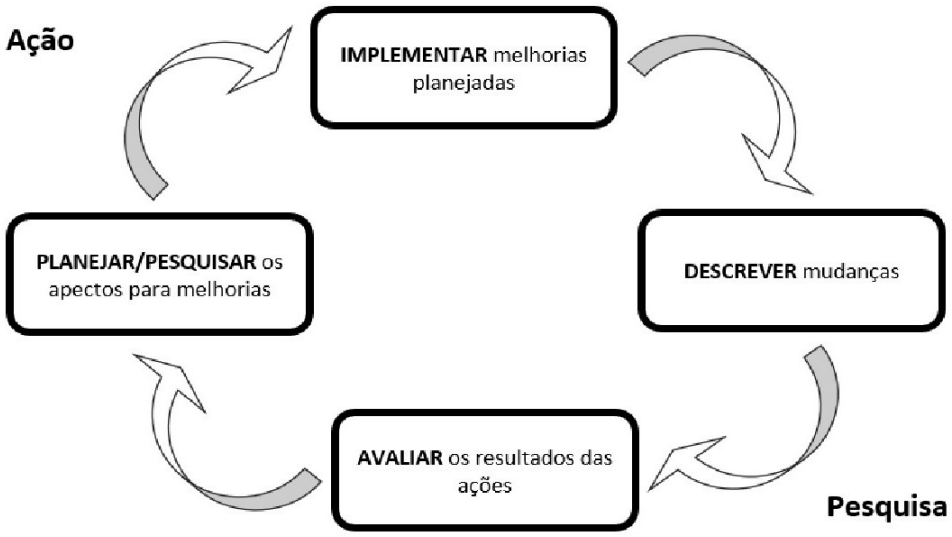


Figura 1 – Representação das 4 fases da pesquisa-ação.

Fonte: Adaptado de (TRIPP, 2005).

- Definição da interface gráfica, aspectos de animação, indicadores de desempenho a serem calculados e reportados;
- Definição das situações de interação Humano-Gêmeo, incluindo, se possível, de atuação no sistema físico a partir de decisões no nível do GD;
- Integração banco de dados do SCADA com o ambiente do GD;
- Testes e Implementação final do GD.

1.5 ESTRUTURA DO DOCUMENTO

Além da introdução apresentada, o restante do projeto está distribuído em mais 8 capítulos. O Capítulo 2 é referente ao âmbito onde este projeto foi desenvolvido. Já o Capítulo 3 serve como fundamentação teórica e ferramentas utilizadas para o desenvolvimento do projeto. No Capítulo 4 são apresentadas as especificações do projeto. No Capítulo 5 é mostrado o desenvolvimento e implementação de cada etapa deste projeto. Por fim, no Capítulo 6 há uma síntese do projeto, bem como possíveis aplicações futuras.

2 LABORATÓRIO DE AUTOMAÇÃO E INFORMÁTICA INDUSTRIAL - LAI

Neste capítulo é apresentado o laboratório onde este trabalho foi desenvolvido. Também são apresentados elementos envolvidos como as estações e o robotino utilizados para o desenvolvimento deste projeto.

2.1 INFRAESTRUTURA

O Laboratório de Automação e Informática Industrial é um laboratório de ensino e pesquisa que faz parte do Departamento de Automação e Sistemas (DAS) do Centro Tecnológico (CTC) da Universidade Federal de Santa Catarina. O LAI possui uma área de 60 m² e inclui diversas unidades automatizadas, 4 robôs manipuladores, 3 linhas de montagem em escala, 14 computadores tipo PC e 4 CLPs, além de um sistema modular de 7 estações da FESTO e respectivos CLPs (Siemens).

O laboratório em questão é utilizado para ensino e pesquisa em áreas relacionadas à automação e informática industrial, como controle de processos, instrumentação, redes industriais, robótica, sistemas embarcados, inteligência artificial e indústria 4.0. Ele também oferece suporte aos cursos de graduação e pós-graduação do DAS, bem como a projetos de pesquisa, de extensão e cooperação com outras instituições e empresas.

Contando com uma equipe qualificada de professores, técnicos e alunos, que desenvolvem atividades práticas e teóricas no laboratório, o LAI também possui equipamentos modernos e atualizados, que permitem a realização de experimentos e simulações em diferentes cenários industriais. O LAI busca constantemente a inovação e a excelência em suas atividades, contribuindo para a formação de profissionais competentes e para o avanço do conhecimento científico e tecnológico na área de automação industrial.

No escopo deste trabalho, das ferramentas disponibilizadas no LAI serão utilizadas 3 novas estações/bancadas da FESTO que já possuem o protocolo OPC-UA embutido, além de um robô móvel *AGV* (*Automated Guided Vehicle*) robotino também da FESTO.

2.2 ESTAÇÕES DE TRABALHO FESTO

Nesta seção são apresentadas as estações presentes no LAI utilizadas para o desenvolvimento deste trabalho e suas respectivas funcionalidades.

2.2.1 *MPS Distribution/Conveyor Station*

A bancada MPS Distribution/Conveyor Station da FESTO é um sistema de produção modular que permite realizar diversas tarefas de distribuição e transporte de

peças com tecnologia pneumática e elétrica. A bancada é composta por um módulo de separação que foi expandido com um módulo transportador e um nó de barramento. O módulo de separação usa um cilindro de dupla ação para empurrar as peças individualmente de um *buffer* de empilhamento. O módulo transportador usa um motor DC para mover a correia que transporta as peças para a direita ou para a esquerda. A correia pode ser parada para separar as peças.

A função da bancada é separar peças do fluxo de materiais, inserir peças no fluxo de materiais, distribuir peças com diferentes orientações. No caso da função de separação, sensores ópticos, sensores de reflexão difusa ou barreiras de luz detectam uma peça sobre a correia. A correia transporta a peça para o separador pneumático, que libera a peça completa (carcaça e inserto). No caso da função de inserção, o separador pneumático pode ser usado para inserir um inserto na carcaça da peça. No caso da função de distribuição, a correia pode ser parada em diferentes posições para distribuir as peças para outras estações ou dispositivos.

A bancada é controlada por um controlador lógico programável (PLC), que pode ser escolhido entre diferentes fabricantes e protocolos. O controlador recebe os sinais dos sensores e envia os comandos para as válvulas, os motores e os atuadores. O controlador também pode se comunicar com outras estações ou dispositivos através de uma rede industrial. O controlador pode ser programado com diferentes linguagens e ferramentas, dependendo das preferências e dos objetivos do usuário (DISTRIBUTION. . . , 2023).

A Figura 2 apresenta a estação *Distribution* utilizada no LAI.

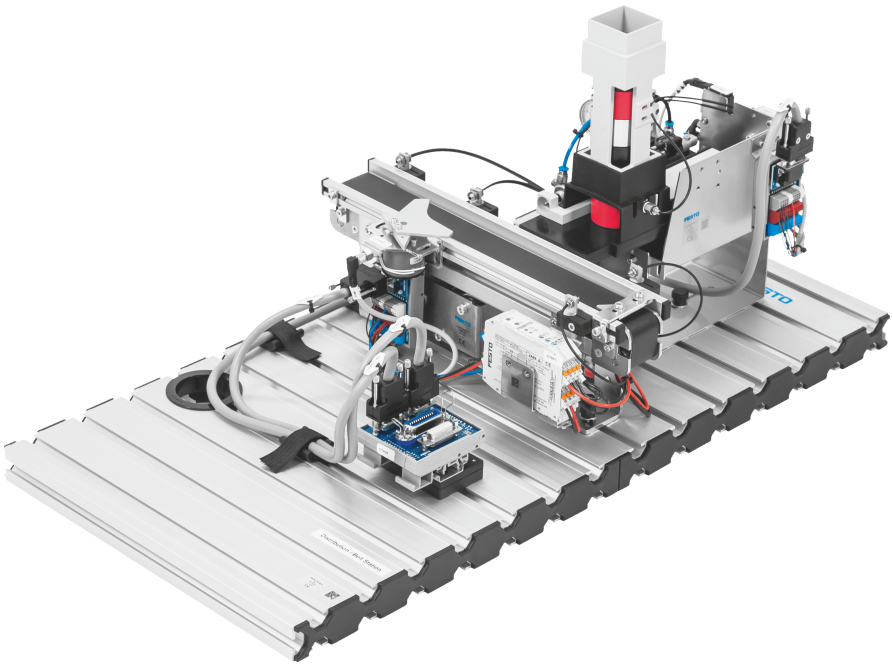


Figura 2 – *Distribution/Conveyor Station*.

Fonte: (DISTRIBUTION. . . , 2023).

2.2.2 *MPS Pick and Place Station*

A bancada MPS *Pick and Place* da FESTO é um sistema de produção modular que permite realizar diversas tarefas de manuseio com tecnologia pneumática e de vácuo. A bancada é composta por um módulo *Pick and Place* de dois eixos e um módulo transportador, que podem ser ajustados em diferentes posições e alturas. A bancada pode pegar, transportar e soltar peças de trabalho com uma ventosa ou uma garra paralela, dependendo da versão.

A função da bancada é: separar peças de trabalho do fluxo de materiais, inserir peças de trabalho no fluxo de materiais, montar peças de trabalho com diferentes orientações. No caso da função de separação, sensores ópticos, sensores de reflexão difusa ou barreiras de luz detectam uma peça de trabalho sobre a correia. A correia transporta a peça de trabalho para o separador elétrico, que libera a peça de trabalho completa (carcaça e inserto). No caso da função de inserção, o módulo *Pick and Place* pega um inserto da peça de trabalho da calha de fornecimento de material e coloca o inserto no alojamento da peça. No caso da função de montagem, o módulo *Pick and Place* pega uma tampa da peça de trabalho da calha de fornecimento de material e monta a tampa na base da peça.

A bancada é controlada por um controlador lógico programável (PLC), que pode ser escolhido entre diferentes fabricantes e protocolos. O controlador recebe os sinais dos sensores e envia os comandos para as válvulas, os motores e os atuadores. O controlador também pode se comunicar com outras estações ou dispositivos através de uma rede industrial. O controlador pode ser programado com diferentes linguagens e ferramentas, dependendo das preferências e dos objetivos do usuário (PICK. . . , 2023).

A Figura 3 apresenta a estação *Pick and Place* utilizada no LAI.

2.2.3 *MPS Sorting Station*

A bancada *Sorting Station* MPS da festo é uma estação de aprendizagem que simula e otimiza um processo de classificação de peças de acordo com suas propriedades, como material e cor. A bancada é composta por quatro módulos principais: o módulo de detecção, o módulo de transporte, o módulo de desvio e o módulo de armazenamento.

O módulo de detecção é responsável por identificar as características das peças, usando sensores ópticos e indutivos. O módulo possui uma barreira óptica de garfo, que detecta a inserção de uma peça na esteira transportadora, um sensor de reflexão difusa, que reconhece a cor da peça, um sensor óptico de fibra, que verifica se a peça é metálica ou não, e um sensor indutivo, que confirma se a peça é metálica ou não.

O módulo de transporte é responsável por movimentar as peças ao longo da bancada, usando uma esteira transportadora acionada por um motor elétrico. O módulo

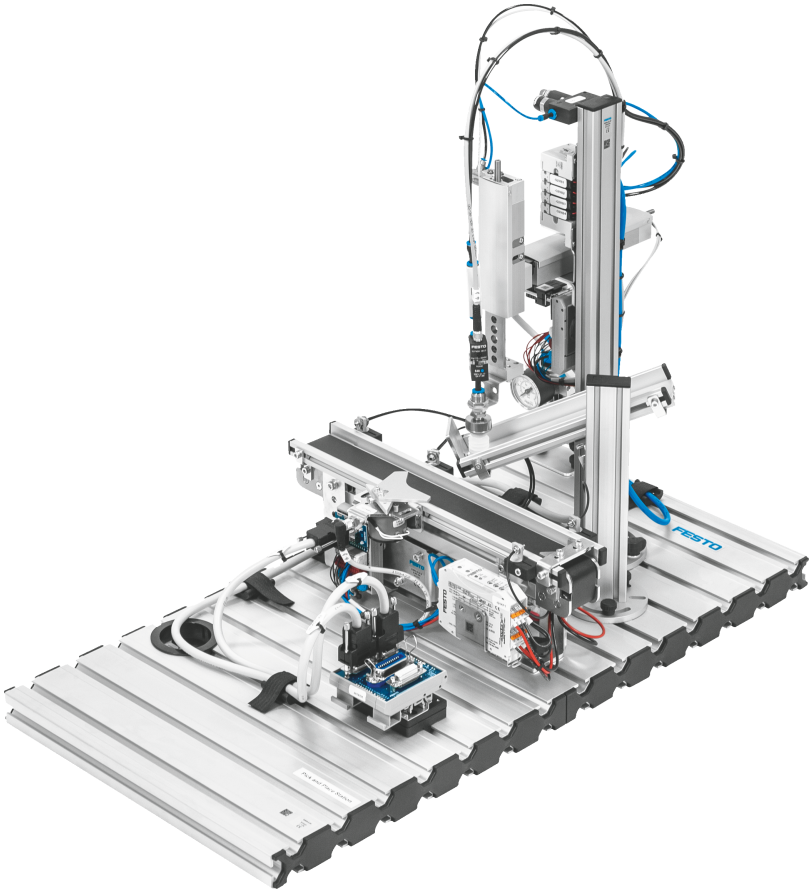


Figura 3 – *Pick and Place Station*.

Fonte: (PICK. . . , 2023).

possui um cilindro pneumático de bloqueio, que para as peças na posição de detecção, e um cilindro pneumático de liberação, que solta as peças para a classificação.

O módulo de desvio é responsável por classificar as peças em três calhas diferentes, de acordo com suas características. O módulo possui três chaves elétricas, que acionam os desvios pneumáticos das calhas. As chaves são controladas por uma lógica programável, que recebe os sinais dos sensores do módulo de detecção. As calhas são destinadas para as peças metálicas vermelhas, as peças metálicas pretas e as peças não metálicas.

O módulo de armazenamento é responsável por armazenar as peças classificadas nas calhas. O módulo possui três sensores ópticos de reflexão, que monitoram o nível de enchimento das calhas. Quando uma calha está cheia, o sensor envia um sinal para a lógica programável, que interrompe o processo de classificação até que a calha seja esvaziada (SORTING. . . , 2023).

A Figura 4 mostra a estação de trabalho de classificação e os módulos que a compõem.

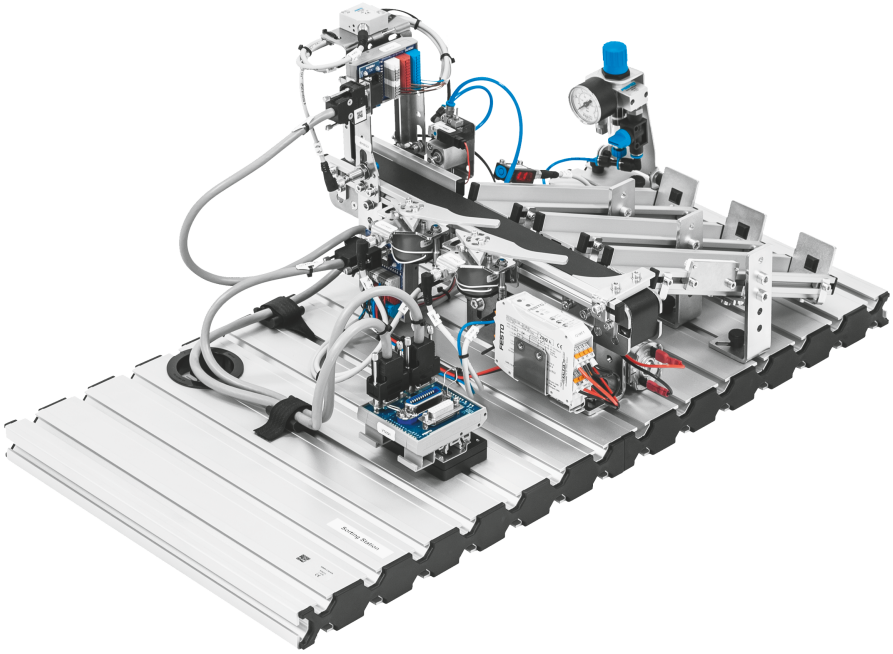


Figura 4 – *Sorting Station*.

Fonte: (SORTING. . . , 2023).

2.2.4 Robotino

O Robotino da festo é uma plataforma móvel de robótica para pesquisa e educação. Equipado com uma tração omnidirecional e com vários sensores, interfaces e extensões modulares, o Robotino pode ser usado de forma flexível para diferentes aplicações relacionadas à robótica. A programação do Robotino pode ser feita com diversos ambientes de programação, como Robotino View, Robotino SIM, MATLAB/Simulink, C/C++, Java, Python, etc.

O Robotino possui três rodas motrizes independentes, que permitem o movimento em qualquer direção e a rotação sobre o próprio eixo. Cada roda possui um encoder óptico para medir a velocidade e a posição. O Robotino também possui um sensor de distância a laser, que permite a detecção de obstáculos e a localização do robô no ambiente. Além disso, o Robotino possui uma câmera colorida, um sensor de inclinação, um sensor de bateria e um alto-falante.

O Robotino pode se comunicar com outros dispositivos por meio de interfaces como *Ethernet*, USB (*Universal Serial Bus*), CAN (*Controller Area Network*), RS-232 (*Recommended Standard 232*) e GPIO (*General Purpose Input/Output*). O Robotino também pode se conectar à internet por meio de uma rede sem fio integrada. O Robotino possui uma placa controladora baseada em Linux, que executa os programas do usuário e os serviços do sistema. O Robotino também possui uma placa de expansão opcional, que permite a conexão de módulos adicionais, como braços robóticos, garras, sensores ultrassônicos, entre outros.

O Robotino é uma ferramenta ideal para aprender e pesquisar sobre robó-

tica móvel, inteligência artificial, visão computacional, navegação autônoma, interação humano-robô e muito mais. O Robotino permite a realização de projetos desafiadores e criativos, que estimulam o pensamento lógico e a resolução de problemas. O Robotino também faz parte da rede de aprendizagem biônica da festo, que busca inspiração na natureza para desenvolver soluções inovadoras para a automação do futuro (ROBOTINO. . . , 2023).

No escopo deste trabalho, ele seria utilizado para a coleta, transporte e transferência de peças entre o conjunto das novas estações de trabalho com as estações de trabalho antigas. Porém, pelo fato de ser um equipamento novo e a falta de familiaridade tanto dos professores do LAI quanto do autor, não foi possível estabelecer a comunicação e interação do robotino com o sistema, sendo portanto postergado seu uso para trabalhos futuros. A Figura 5 apresenta o Robotino versão 4.

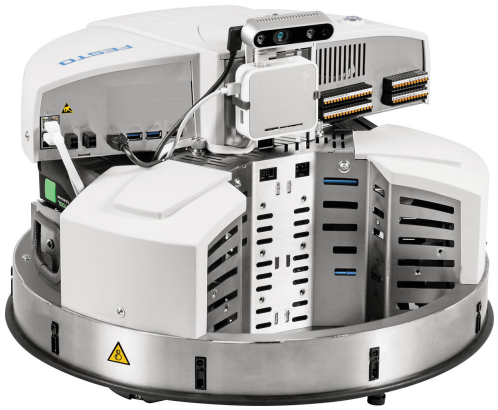


Figura 5 – Robotino.

Fonte: (ROBOTINO. . . , 2023).

3 FUNDAMENTAÇÃO TEÓRICA E FERRAMENTAS

O propósito deste capítulo é fornecer uma visão concisa do enquadramento metodológico empregado neste estudo, juntamente com as etapas e procedimentos definidos para alcançar os objetivos previamente estabelecidos, além da fundamentação teórica e das ferramentas utilizadas para o desenvolvimento do projeto.

3.1 FUNDAMENTAÇÃO TEÓRICA

Nesta seção são abordados os conceitos teóricos assim como os materiais de apoio e a metodologia utilizados para a fundamentação do desenvolvimento deste projeto.

3.1.1 Indústria 4.0

Nas primeiras três revoluções industriais, a ênfase estava voltada para os níveis do chão de fábrica e os processos de produção. A primeira revolução industrial teve início no final do século XVIII, marcando a transição do trabalho manual para a adoção de motores a vapor e água como fontes de energia, impulsionando o desenvolvimento da agricultura, transporte e fábricas. A Figura 6 mostra a linha do tempo, descrevendo as principais mudanças que identificam os inícios aproximados de cada revolução.

No início do século XX, a segunda revolução industrial foi caracterizada pela introdução da eletricidade e pela implementação da produção em massa concebida por Henry Ford. Na década de 1970, a terceira revolução industrial tornou-se parte integrante da indústria, destacando a mudança das tecnologias analógicas para as digitais e a introdução de computadores, também conhecida como a "Era da Automação"(REINER, 2014). A expectativa agora concentra-se no significativo avanço da quarta revolução industrial, com a introdução dos chamados Sistemas Ciberfísicos (apresentados mais aprofundadamente na Subseção 3.1.2).

A Indústria 4.0 (I4.0) tem origem na Alemanha, onde foi criado o termo em 2011, como parte de um projeto de estratégia de alta tecnologia do governo alemão, que visava promover a digitalização da manufatura. O conceito foi inspirado pelo projeto de pesquisa “Industrie 4.0”, que tinha como objetivo desenvolver uma arquitetura de referência para a integração de sistemas ciberfísicos na indústria. De acordo com (LIAO, Yongxin *et al.*, 2017) a I4.0 é parte da estratégia Alemã a longo prazo para fortalecer a competitividade no setor manufatureiro.

Não há uma definição padronizada para a Indústria 4.0, como indicado por (LIAO, Yongxin *et al.*, 2017). A literatura oferece diversas interpretações, conforme mencionado por (MOEUF *et al.*, 2018). Em termos gerais, conforme apontado por (RABELO, Ricardo José, 2021), a I4.0 caracteriza-se como um modelo de produção

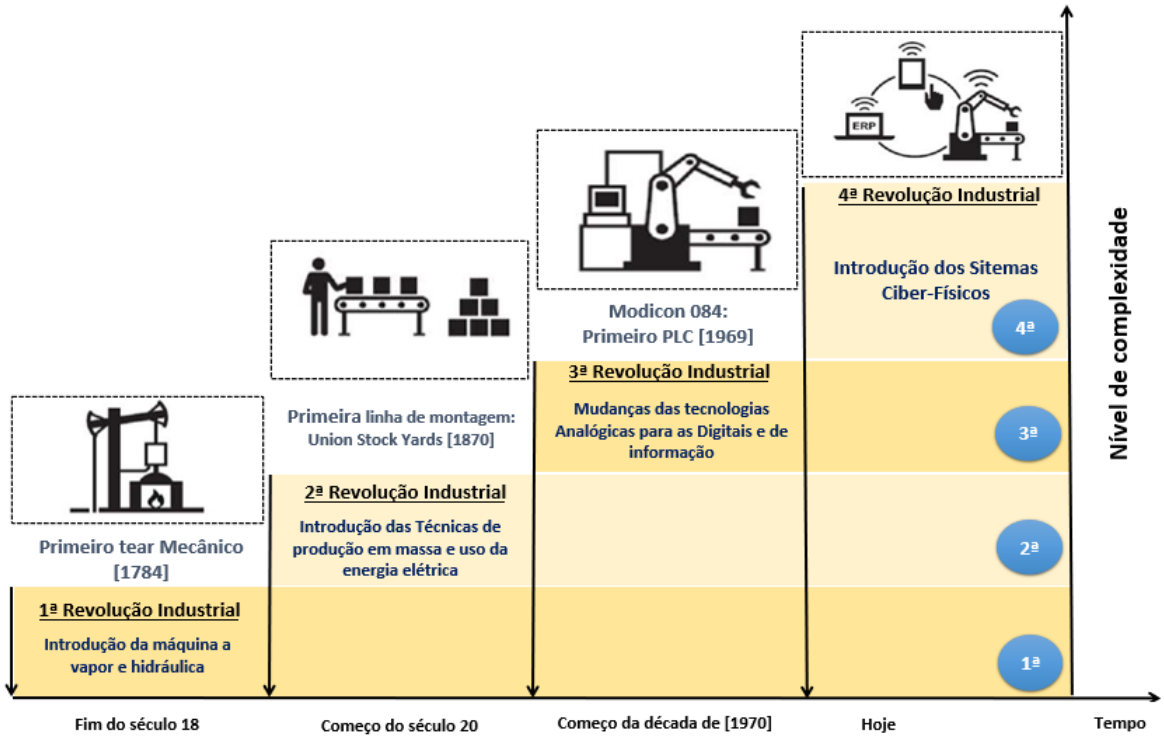


Figura 6 – História das revoluções industriais.

Fonte: Adaptado de (REINER, 2014).

que se destaca pela intensa digitalização e interconexão de tecnologias, sistemas de manufatura inteligentes, produtos e serviços, presentes em cadeias de valor múltiplas e dinâmicas, com modelos de negócios variáveis. Essa abordagem é fundamentada nos princípios da Internet das Coisas, dos Serviços e das Pessoas. A Indústria 4.0 é comumente descrita como um conjunto de princípios de design e tecnologias capacitadoras, fornecendo diretrizes para pesquisadores e profissionais na implementação de cenários I4.0 nas empresas (GHOBAKHLOO, 2018).

O grupo de trabalho alemão **Industrie 4.0** apresenta uma visão abrangente das tecnologias fundamentais e dos conceitos orientadores da I4.0. O relatório, divulgado em 2013, não apenas aborda a compreensão do tema, mas também fornece insights sobre como alcançá-lo. De acordo com (KAGERMANN; WAHLSTER; HELBIG, 2013b), para uma implementação bem-sucedida e o desenvolvimento eficaz do conceito I4.0, é essencial agir em oito áreas específicas:

- **Padronização e arquitetura de referência:** Considerando que a I4.0 envolverá networking e integração de várias empresas diferentes através de redes de valor, esta parceria colaborativa, apenas será possível se um único conjunto de padrões forem desenvolvidos;
- **Gerenciamento de sistemas complexos:** Como os produtos e sistemas de manufaturas estão se tornando cada vez mais complexos, o planejamento apro-

priado e os modelos explicativos devem fornecer uma base para gerenciar essa complexidade crescente. Os engenheiros devem, portanto, estar equipados com os métodos e ferramentas necessárias para desenvolver tais modelos;

- **Uma infraestrutura abrangente de banda larga para a indústria:** Redes de comunicação confiáveis, abrangentes e de alta qualidade, são requisitos fundamentais para a I4.0. A infraestrutura de banda larga da internet, portanto, precisa ser expandida em alta escalabilidade;
- **Segurança e proteção de dados:** Proteção e segurança são essenciais para o sucesso de sistemas de manufatura inteligente. É importante garantir que as instalações de produção e os próprios produtos não representem um perigo para as pessoas ou para o meio ambiente, enquanto, em paralelo, os dados e informações de produção e produtos, precisam ser protegidos contra o uso indevido e acesso não autorizado;
- **Organização de Trabalho:** Em fábricas inteligentes, o papel dos colaboradores mudará significativamente. O controle orientado cada vez mais em tempo real, transformará a operação de trabalho, os processos de trabalho e ambientes de trabalho. Ênfase na implementação de uma abordagem de sistemas sociotécnicos (STS - *Socio-Technical Systems*), para organizações oferecerem aos trabalhadores a oportunidade de ter maior responsabilidade e potencializar seu desenvolvimento pessoal;
- **Treinamento e desenvolvimento profissional contínuo:** Será necessário implementar estratégias de formação adequadas e organizar o trabalho de forma a fomentar a aprendizagem, possibilitando a aprendizagem ao longo da vida, onde várias técnicas de aprendizagem digital devem ser investigadas;
- **Framework regulamentado:** Embora novos processos de fabricação e redes de negócios envolvidos na I4.0 precisem estar em conformidade com a lei, a legislação vigente também precisará ser adaptada para levar em conta as inovações. Os desafios incluem a proteção de dados corporativos, questões de responsabilidade, tratamento de dados pessoais e restrições comerciais;
- **Eficiência de recursos:** O consumo de grandes quantidades de matérias primas e energia pela indústria, apresenta uma série de ameaças ao meio ambiente e à segurança no abastecimento. I4.0 proporcionará ganhos em produtividade e eficiência em recursos.

O relatório destaca que o percurso em direção à I4.0 será um processo em constante evolução. É crucial adaptar as tecnologias e experiências conforme os requisitos específicos da engenharia de manufatura, o que implica na exploração de soluções

inovadoras e na abertura para novos mercados (KAGERMANN; WAHLSTER; HELBIG, 2013b).

Conforme descrito por Vaidya et al. (2018), o desenvolvimento do conceito de Indústria 4.0 envolve nove pilares tecnológicos cruciais. Esses pilares são:

- big data e análise de dados;
- robôs autônomos;
- simulação;
- integração de sistemas;;
- Internet Industrial das Coisas (IIoT);
- segurança cibernética e Sistemas Ciber-Físicos (CPS);
- computação em nuvem;
- manufatura aditiva;
- realidade aumentada.

Embora essas tecnologias ou conceitos não sejam propriamente novos e não tenham sido criados exclusivamente para a Indústria 4.0 ou para a manufatura em geral, suas contribuições ao longo de suas respectivas evoluções e seu uso combinado têm desempenhado um papel significativo no avanço da Indústria 4.0 (RABELO, Ricardo José, 2021). O estudo de (LU, 2017) explora como o tema da quarta revolução industrial tem sido abordado na literatura, identificando conceitos, perspectivas, tecnologias-chave, arquiteturas básicas e aplicações nos diversos setores produtivos.

A integração e combinação dessas tecnologias em um sistema robusto e flexível são essenciais para garantir que as indústrias estejam preparadas para operar em um cenário no qual diversos stakeholders estão interconectados. Além disso, a interação homem-máquina torna-se cada vez mais ágil e flexível (ROBLEK; MEŠKO; KRAPEŽ, 2016).

Alguns exemplos de aplicações da indústria 4.0 em diferentes setores são:

- a manufatura aditiva, que permite a criação de peças complexas e personalizadas por meio da impressão 3D;
- a agricultura de precisão, que usa sensores, drones e sistemas de irrigação inteligentes para otimizar o uso de recursos e aumentar a produtividade;
- a saúde digital, que usa dispositivos vestíveis, telemedicina e análise de dados para monitorar e melhorar a qualidade de vida dos pacientes;

- a energia inteligente, que usa redes elétricas inteligentes, fontes renováveis e armazenamento de energia para gerenciar a demanda e a oferta de energia de forma eficiente e sustentável.

A Indústria 4.0 é uma realidade que já está transformando o mundo em que vivemos e que exigirá de nós uma constante adaptação e aprendizagem. A Indústria 4.0 não é apenas uma questão técnica, mas também uma questão social, política, econômica, cultural e ética. A Indústria 4.0 não é um fim em si mesma, mas um meio para alcançar um desenvolvimento sustentável, inclusivo e humano.

3.1.2 Sistemas Ciberfísicos

De acordo com (HERMANN; PENTEK; OTTO, 2015), um dos aspectos centrais da Indústria 4.0 é a convergência entre o mundo físico e o digital. A concepção desta fusão se dá pelos Sistemas Ciberfísicos (SCF) - termo inicialmente proposto pela cientista americana Helen Gill na *National Science Foundation* (NSF) e posteriormente descrito por Lee como sendo a integração entre processos físicos e computacionais, ao qual as redes e computadores integrados controlam e monitoram os processos físicos em um processo de loop de feedback e interferem na computação e vice-versa (LEE; BAGHERI; KAO, 2015).

Conforme apresentado por (COLOMBO; KARNOUSKOS; KAYNAK, 2017), os SFCs têm sua origem em uma ampla aplicação de sistemas embarcados, embora os mesmos não sejam apenas sistemas embarcados em rede mas sim sistemas autônomos baseados em software com a capacidade de adaptação e evolução. À medida que evoluem, os SFCs podem ser categorizados em três gerações distintas. A primeira geração introduz informações por meio de tecnologias como tags RFID (*Radio Frequency Identification*), que permitem a identificação unificada de peças e produtos. Na segunda geração, são incorporados aos SFCs dispositivos como sensores e atuadores. Já na terceira geração, esses sistemas se tornam integrados à rede, possuindo a capacidade de armazenamento e análise de dados, como apontado por (HERMANN; PENTEK; OTTO, 2015).

Um dos principais conceitos dos SCFs apresentado por (PETRONI; GONÇALVES; SATYRO, 2018) é o do "3C", sendos esses:

- Computação: responsável por processar os dados coletados pelos sensores e gerar comandos para os atuadores;
- Comunicação: é o meio pelo qual os dados e os comandos são transmitidos entre os dispositivos e as plataformas de software;
- Controle: é a disciplina que estuda como projetar e implementar sistemas que

garantam o desempenho desejado dos processos físicos, considerando as incertezas e as restrições do ambiente.

Mediante a integração dos 3Cs, os SFC demonstram a capacidade de obter informações em tempo real, realizar um controle dinâmico e disponibilizar diversos serviços relacionados a equipamentos ou sistemas específicos. Essas informações são adquiridas por meio de transdutores, independentemente de sua localização, como apontado por (LIAO, Ying *et al.*, 2017). Esse enfoque na integração ciberfísica e nos loops de feedback permite o monitoramento e controle de entidades físicas de maneira confiável, segura, colaborativa, robusta e eficiente, como também destacado por (LIAO, Ying *et al.*, 2017).

3.1.3 Gêmeos Digitais

Os gêmeos digitais são representações virtuais em tempo real de objetos, processos e sistemas. Eles permitem simular, monitorar, analisar e otimizar o desempenho de um sistema físico, usando dados coletados por sensores e dispositivos conectados à internet. Os gêmeos digitais são considerados uma das principais tecnologias da Indústria 4.0, pois possibilitam a integração entre os mundos físico e digital, aumentando a eficiência, a qualidade e a inovação dos processos industriais (UTILIZAÇÃO. . . , s.d.).

Os gêmeos digitais podem ser aplicados em diversos setores e contextos, como por exemplo, na manufatura, na saúde, na agricultura, na energia, na mobilidade, na segurança e na educação. Alguns exemplos de uso dos gêmeos digitais são:

- Na manufatura, os gêmeos digitais podem auxiliar no planejamento, na simulação, na operação e na manutenção de máquinas, equipamentos e linhas de produção, permitindo testar diferentes cenários, identificar gargalos, prever falhas, reduzir custos e melhorar a qualidade dos produtos (QUINALHA, 2018) (GÊMEOS. . . , 2022);

- Na saúde, os gêmeos digitais podem ser usados para criar modelos virtuais de órgãos, tecidos e sistemas do corpo humano, facilitando o diagnóstico, o tratamento, a prevenção e a pesquisa de doenças, além de possibilitar a personalização da medicina e o desenvolvimento de novos fármacos e dispositivos médicos (O. . . , s.d.);

- Na agricultura, os gêmeos digitais podem contribuir para o aumento da produtividade, da sustentabilidade e da segurança alimentar, ao permitir o monitoramento e a gestão inteligente de cultivos, solos, clima, pragas, doenças, recursos hídricos e insumos agrícolas, usando técnicas de agricultura de precisão e de agricultura digital (GÊMEOS. . . , 2022).

Os gêmeos digitais são baseados em três pilares: o modelo virtual, o modelo físico e a conexão entre eles. O modelo virtual é uma réplica digital que reproduz as características, o comportamento e o estado do modelo físico, que é o objeto, processo ou sistema real. A conexão entre eles é feita por meio de sensores, dispositivos e plataformas de comunicação, que permitem a coleta, o envio, o armazenamento e o processamento de dados em tempo real, gerando informações e insights que podem ser usados para aprimorar o desempenho do modelo físico(GÊMEOS. . . , 2021).

3.1.3.1 Arquitetura GD

Conforme apresentado por (RABELO, Ricardo J; MAGALHÃES; CABRAL, 2020), a arquitetura de um GD possui quatro camadas principais, sendo essas:

- Usuários do GD;
- Funcionalidades e serviços do GD;
- Coleta de dados e controle do SCF;
- Entidades de chão-de-fábrica.

3.1.3.1.1 Usuários do GD

A camada de usuários do GD é a parte da arquitetura de referência que representa os diferentes tipos de usuários que podem interagir com o GD, como operadores, gestores, clientes, etc. Essa camada permite que os usuários acessem as funcionalidades e serviços do GD, como visualização, análise, simulação, gestão, entre outros. Ela também pode receber informações e comandos do GD, como alertas, recomendações, atuações, etc. A camada de usuários do GD pode se comunicar com o GD por meio de diferentes interfaces, como web, mobile, realidade virtual ou aumentada, softbot e afins. Além disso, essa camada deve garantir a segurança, a privacidade e a conformidade dos dados e das ações dos usuários.

3.1.3.1.2 Funcionalidades e Serviços do GD

A camada de Funcionalidades e Serviços do GD é a parte da arquitetura de referência que representa as capacidades e os recursos que o GD oferece aos usuários e aos sistemas externos. Essa camada é responsável por:

- Coletar, tratar, enriquecer e armazenar os dados do SCF, garantindo a interoperabilidade, a integridade e a atualização dos dados;
- Criar e manter o modelo digital do SCF, permitindo a visualização, a gravação e a geração de históricos do seu comportamento;

- Realizar análises de dados, simulações, otimizações e atualizações inteligentes, baseadas nos dados do SCF e nos objetivos do GD;
- Gerenciar o GD, monitorando as variáveis críticas do SCF, gerando indicadores, alertas, recomendações e atuações, e garantindo o compliance e a segurança dos dados e das ações;
- Interagir com os usuários, fornecendo interfaces adequadas, dashboards, softbots, e permitindo a personalização e a adaptação do GD.

3.1.3.1.3 Coleta de Dados e Controle do SCF

A camada de Coleta de Dados e Controle do SCF é a parte da arquitetura de referência que representa a conexão entre o sistema ciberfísico (SCF) e o gêmeo digital (GD). Essa camada é responsável por:

- Coletar os dados dos sensores, atuadores e controladores do SCF, usando diferentes protocolos e redes de comunicação, como Fieldbus, Modbus, Profinet, OPC, MQTT, etc;
- Enviar os dados do SCF para o GD, garantindo a fidelidade, a atualização e a segurança dos dados;
- Receber as informações e os comandos do GD, que podem afetar o comportamento e o estado do SCF, como alertas, recomendações, atuações, etc;
- Controlar os dispositivos e os processos do SCF, seguindo as lógicas e os programas definidos nos controladores lógicos programáveis (CLPs) ou em outros sistemas de automação.

3.1.3.1.4 Entidades de chão-de-fábrica

A camada de Entidades de chão-de-fábrica é a parte da arquitetura de referência que representa os elementos físicos que compõem o sistema ciberfísico (SCF) de manufatura, tais como máquinas, sensores, atuadores, controladores, produtos, materiais, operadores, etc. Essa camada é responsável por:

- Executar os processos de produção, seguindo as lógicas e os programas definidos nos controladores lógicos programáveis (CLPs) ou em outros sistemas de automação;
- Gerar e transmitir os dados dos sensores, atuadores e controladores do SCF, usando diferentes protocolos e redes de comunicação, como Fieldbus, Modbus, Profinet, OPC, MQTT, etc;

- Receber as informações e os comandos do gêmeo digital (GD), que podem afetar o comportamento e o estado do SCF, como alertas, recomendações, atuações, etc;
- Interagir com os usuários, fornecendo interfaces adequadas, como interfaces homem-máquina (IHMs), realidade virtual ou aumentada, softbot, etc.

Os gêmeos digitais são uma tecnologia disruptiva que pode trazer diversos benefícios para as indústrias e para a sociedade, dentre esses:

- Aumentar a competitividade, a produtividade e a rentabilidade dos negócios, ao permitir a otimização dos processos, a redução dos desperdícios, a melhoria da qualidade e a inovação dos produtos e serviços;
- Aumentar a eficiência, a confiabilidade e a segurança dos sistemas, ao permitir a previsão, a detecção e a correção de problemas, a redução dos riscos, a melhoria da performance e a extensão da vida útil dos ativos;
- Aumentar a sustentabilidade, a responsabilidade e a transparência das operações, ao permitir o monitoramento e a gestão dos impactos ambientais, sociais e econômicos, a redução das emissões de gases de efeito estufa, o uso racional dos recursos naturais e a conformidade com as normas e regulamentações.

3.1.4 Application Programming Interface - API

Uma API (*Application Programming Interface*) é um conjunto de regras e especificações que permitem que diferentes sistemas se comuniquem entre si de forma padronizada e eficiente. As APIs são amplamente utilizadas no mundo da tecnologia e dos negócios, pois facilitam a integração, a inovação e a escalabilidade de serviços e produtos digitais (WHAT..., 2023).

As APIs podem ser classificadas em diferentes tipos, de acordo com o seu propósito, o seu nível de abstração, o seu formato de dados, o seu protocolo de comunicação, entre outros critérios. Alguns exemplos de tipos de APIs são: REST, SOAP, GraphQL, RPC, Webhooks, etc. Cada tipo de API tem suas vantagens e desvantagens, e cabe ao desenvolvedor escolher o mais adequado para o seu projeto (A..., 2022).

As APIs REST (*Representational State Transfer*) são um dos tipos mais populares e utilizados de APIs, pois seguem um conjunto de princípios e boas práticas que tornam as APIs mais simples, flexíveis e interoperáveis. As APIs REST usam o protocolo HTTP para realizar as operações de criação, leitura, atualização e exclusão de recursos, utilizando os métodos GET, POST, PUT, PATCH e DELETE. Além disso, as APIs REST usam formatos de dados comuns, como JSON ou XML, para representar os recursos e as respostas (REST..., 2023).

As APIs são fundamentais para a economia digital, pois permitem que diferentes empresas e organizações compartilhem dados e funcionalidades de forma segura e controlada, criando novas oportunidades de negócios e parcerias. As APIs também possibilitam que os desenvolvedores criem aplicações mais ricas e personalizadas, aproveitando os serviços e as informações de outras fontes. Por exemplo, uma aplicação de transporte pode usar a API do Google Maps para calcular rotas e mostrar mapas, ou uma aplicação de música pode usar a API do Spotify para reproduzir músicas e playlists (O. . . , 2023).

No contexto deste projeto, foi desenvolvida uma API para estabelecer contato com os gêmeos digitais em tempo real, disponibilizando as informações de operações das bancadas e permitindo também intereções de comando com o SCF.

3.1.5 Protocolo OPC-UA

O protocolo OPC-UA (*Open Platform Communications-Unified Architecture*) é um padrão aberto para comunicação industrial que permite a troca de dados e informações entre sistemas heterogêneos de forma segura, confiável e independente de plataforma (MAHNKE; LEITNER; DAMM, 2016). O OPC-UA surgiu como uma evolução do OPC Classic, um padrão baseado em tecnologias Microsoft que tinha limitações de interoperabilidade, escalabilidade e seguranç (GRÄSSLE *et al.*, 2016). O OPC-UA foi lançado em 2008 pela OPC Foundation, uma organização sem fins lucrativos que reúne mais de 700 empresas e instituições do setor industrial (OPC. . . , s.d.).

É reconhecido como "a tecnologia mais importante de interoperabilidade no cenário industrial atual"pelo ARC Advisory Group, uma empresa líder em pesquisa e consultoria em automação industrial (OPC. . . , s.d.). Ele é compatível com os conceitos e requisitos da Indústria 4.0 (Subseção 3.1.1), sendo capaz de fornecer uma comunicação padronizada, transparente e semântica entre os dispositivos e sistemas que compõem uma fábrica inteligente, desde os sensores e atuadores até os sistemas de gestão e planejamento (MAHNKE; LEITNER; DAMM, 2016), (GRÄSSLE *et al.*, 2016).

O protocolo OPC-UA utiliza uma arquitetura cliente-servidor, na qual o servidor expõe um modelo de informação que representa os dados e serviços disponíveis, e o cliente acessa esse modelo de forma assíncrona e orientada a eventos (GRÄSSLE *et al.*, 2016). O modelo de informação é baseado em um conjunto de objetos, variáveis, métodos e tipos que podem ser organizados em uma estrutura hierárquica e descritos por meio de metadados. Permite a definição de modelos de informação específicos para cada domínio ou indústria, chamados de *Companion Specifications*, que facilitam a integração e a interoperabilidade entre diferentes aplicações (MAHNKE; LEITNER; DAMM, 2016), (SCHRIEGEL; JASPERNEITE, 2018).

Ele também oferece mecanismos de segurança robustos, que incluem a autenticação, a autorização, a criptografia e a assinatura digital dos dados e mensagens

trocados entre os nós da rede (GRÄSSLE *et al.*, 2016). Além disso, o OPC-UA suporta diferentes protocolos de transporte, como TCP, HTTP, HTTPS e UDP, e diferentes formatos de codificação, como XML, JSON e binário, o que permite a adaptação às necessidades e restrições de cada cenário de comunicação (MAHNKE; LEITNER; DAMM, 2016), (SCHRIEGEL; JASPERNEITE, 2018).

O protocolo OPCUA é considerado uma tecnologia habilitadora para a convergência das redes de comunicação industrial, que atualmente são fragmentadas e incompatíveis entre si, podendo se beneficiar da integração com o TSN (*Time-Sensitive Networking*), um padrão IEEE para *Ethernet* que suporta o comportamento em tempo real. A combinação de OPC-UA e TSN pode proporcionar uma rede única, aberta e padronizada para a comunicação de dados críticos e não críticos entre os dispositivos e sistemas da Indústria 4.0 e da Internet das Coisas Industrial (GRÄSSLE *et al.*, 2016), (SCHRIEGEL; JASPERNEITE, 2018).

No apêndice A deste documento se encontra um passo a passo de como se construir e configurar o protocolo OPCUA. No escopo deste projeto, o protocolo será utilizado para estabelecer a comunicação entre o SCF e o gêmeo digital.

3.1.6 Desenvolvimento Ágil

O desenvolvimento ágil é uma forma de criar software que se adapta às mudanças e às necessidades do cliente, de forma colaborativa, iterativa e incremental. Essa abordagem surgiu como uma reação contra os métodos tradicionais de desenvolvimento, que eram considerados burocráticos, lentos e inflexíveis. O desenvolvimento ágil se baseia em quatro valores e doze princípios, que foram definidos no Manifesto Ágil (MARTIN, 2003), publicado em 2001 por um grupo de 17 desenvolvedores. Os quatro valores do desenvolvimento ágil são:

- Indivíduos e interações mais que processos e ferramentas, isso é, o desenvolvimento ágil prioriza as pessoas e a comunicação entre elas, em vez de seguir processos e usar ferramentas pré-definidas. Isso implica em formar equipes auto-organizadas, motivadas e colaborativas, que se comunicam de forma frequente, transparente e eficaz, tanto entre si quanto com os clientes e os usuários;
- Software em funcionamento mais que documentação abrangente, no qual o desenvolvimento ágil foca em entregar software que funcione e que resolva os problemas dos clientes, em vez de produzir documentação extensa e detalhada sobre o projeto. Isso implica em entregar software de forma rápida e contínua, e obter feedback constante dos clientes e dos usuários, para validar e melhorar o produto;
- Colaboração com o cliente mais que negociação de contratos, onde o desenvolvimento ágil busca envolver o cliente em todas as fases do projeto, em vez de se

limitar a um contrato fixo e rígido. Isso implica em estabelecer uma relação de confiança e parceria com o cliente, e entender suas necessidades, expectativas e demandas, para entregar um produto que o satisfaça e o surpreenda;

- Responder a mudanças mais que seguir um plano, dessa forma o desenvolvimento ágil aceita e se adapta às mudanças que ocorrem durante o projeto, em vez de seguir um plano pré-estabelecido e imutável. Isso implica em ser flexível e criativo, e aproveitar as mudanças como oportunidades de melhoria e inovação, para entregar um produto que seja relevante e atualizado.

Os doze princípios do desenvolvimento ágil são:

- A maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado;
- Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento, pois os processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente;
- Entregas frequentes do software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo;
- Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto;
- Construir projetos em torno de indivíduos motivados. Dar a eles o ambiente e o suporte necessário e confiar neles para fazer o trabalho;
- O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa *face to face*;
- O software funcionando é a medida primária de progresso;
- Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente;
- Contínua atenção à excelência técnica e bom design aumenta a agilidade;
- Simplicidade - a arte de maximizar a quantidade de trabalho não realizado - é essencial;
- As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis;

- Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Existem diferentes métodos e frameworks que seguem a abordagem ágil, como Scrum, Kanban, *Extreme Programming* (XP), *Test Driven Development* (TDD), *Behavior Driven Development* (BDD), *Domain Driven Design* (DDD), entre outros. Cada um deles tem suas próprias características, vantagens e desvantagens, mas todos compartilham os mesmos valores e princípios. O objetivo é escolher o método que melhor se adapte ao contexto e às necessidades do projeto, e que permita entregar software de qualidade, com rapidez e eficiência, satisfazendo os clientes e os usuários.

No contexto deste trabalho, foram utilizados principalmente os métodos de TDD para escrever os testes e os códigos que descrevem o comportamento do software e o DDD para modelar o domínio do software e a arquitetura adequada.

Além disso, o desenvolvimento ágil possui algumas etapas que orientam o desenvolvimento, apresentadas na Figura 7. Conforme mostra a figura, há 6 etapas - a nomenclatura e a quantidade variam de acordo com as fontes, mas todas apresentam as mesmas características - sendo essas:

- Planejamento: Nessa etapa, o escopo, o cronograma, o orçamento e os requisitos do projeto são definidos, bem como as prioridades e os riscos. O planejamento também envolve a definição das equipes, das ferramentas e dos métodos que serão usados no projeto;

- Especificação: Nessa etapa, o domínio do problema é analisado, os requisitos são refinados e validados, e as soluções são propostas e avaliadas. A análise também envolve a elaboração de casos de uso, histórias de usuário, diagramas e protótipos;

- Design: Nessa etapa, a arquitetura e o design do software são definidos, seguindo os padrões e as boas práticas de desenvolvimento. O design também envolve a escolha das tecnologias, das bibliotecas e dos frameworks que serão usados no projeto;

- Desenvolvimento iterativo: Nessa etapa, o código do software é escrito, seguindo as especificações e os testes definidos nas etapas anteriores. O desenvolvimento também envolve a integração, a revisão e a refatoração do código;

- Teste: Nessa etapa, o software é testado para verificar se ele atende aos requisitos, às expectativas e aos padrões de qualidade. O teste também envolve a identificação e a correção de erros, falhas e defeitos;

- Produção: Nessa etapa, o software é implantado em um ambiente de produção, onde ele pode ser usado pelos clientes finais. A implantação também envolve a configuração, a instalação e a atualização do software;
- Monitoramento: Nessa etapa, o software é monitorado, suportado e melhorado, de acordo com as necessidades e os feedbacks dos clientes. A manutenção também envolve a correção de bugs, a adição de novas funcionalidades e a otimização do desempenho.

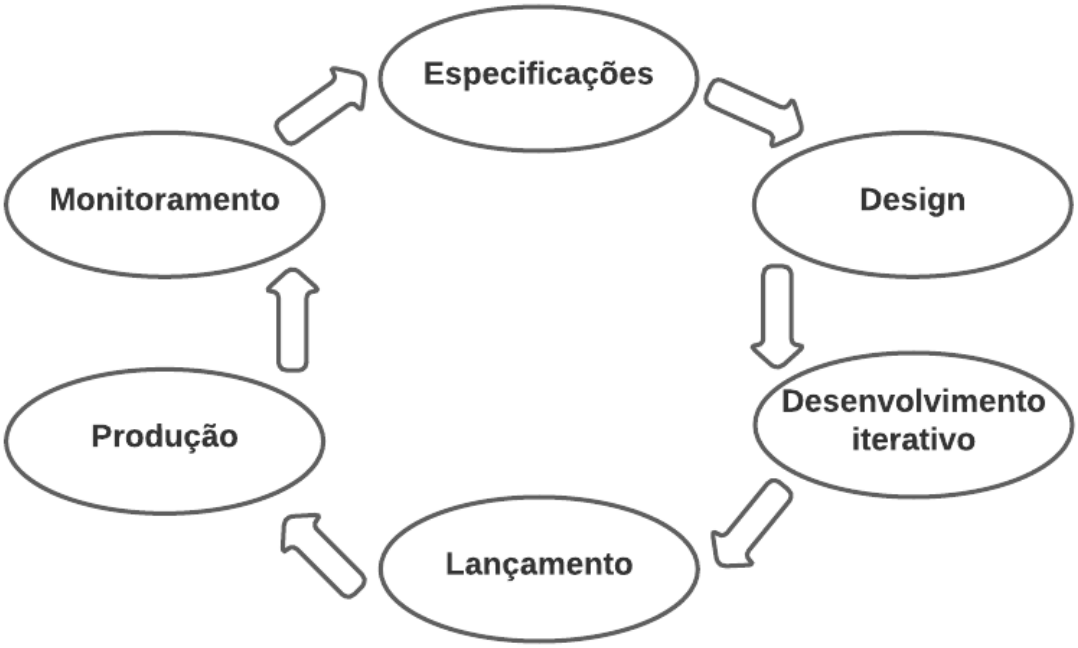


Figura 7 – Representação das etapas da metodologia de desenvolvimento ágil.

Fonte: Arquivo pessoal.

3.2 FERRAMENTAS

Nesta seção são apresentadas as ferramentas utilizadas para a realização deste projeto.

3.2.1 PlantSimulation

O software *PlantSimulation* da Siemens é uma ferramenta que permite a simulação e a otimização de sistemas e processos de produção, usando uma abordagem baseada em objetos e modelos hierárquicos. Com o *PlantSimulation*, é possível criar representações virtuais de sistemas logísticos, como fábricas, linhas de produção, armazéns, transportes, etc., e explorar as suas características e desempenho. O software

também permite realizar experimentos e cenários hipotéticos sem afetar os sistemas reais ou antes de instalá-los (PLANT. . . , 2023).

O funcionamento do *PlantSimulation* se baseia na coleta e no processamento de dados em tempo real, provenientes de sensores ou outras fontes de informação. Esses dados são aplicados aos modelos digitais, que podem ser visualizados em 3D, usando dados CAD ou o formato JT. O software também utiliza técnicas de inteligência artificial, computação em nuvem e aprendizado de máquina para analisar os dados e gerar soluções otimizadas para os problemas encontrados. Além disso, oferece diversas ferramentas e gráficos para avaliar o desempenho dos sistemas, como detecção de gargalos, análise de fluxo, utilização de recursos, consumo de energia, custos, diagramas de Sankey e gráficos de Gantt.

O *PlantSimulation* é uma ferramenta essencial para o desenvolvimento de um gêmeo digital, que é uma réplica virtual de um produto ou sistema físico, que pode ser atualizada com dados em tempo real e usar simulação, aprendizado de máquina e raciocínio para auxiliar na tomada de decisão. Um gêmeo digital pode ser usado para testar, monitorar, otimizar e melhorar o produto ou sistema ao longo do seu ciclo de vida. Com o *PlantSimulation*, é possível criar gêmeos digitais de diferentes níveis de complexidade e abrangência, desde instalações de produção até componentes específicos. O software também permite integrar os gêmeos digitais com outros aplicativos da Siemens, como NX Line Designer, Teamcenter, Simcenter HEEDS, Opcenter APS, TIA Portal, PLCSIM Advanced e SIMIT.

Além disso, é um software que traz diversos benefícios para as empresas que querem se adaptar à Quarta Revolução Industrial. Com o *PlantSimulation*, é possível reduzir custos, aumentar a produtividade, melhorar a qualidade, diminuir o tempo de lançamento no mercado, antecipar problemas e soluções, inovar produtos e processos e criar novos modelos de negócio. O software também contribui para a sustentabilidade ambiental, ao permitir a redução do consumo de energia e dos resíduos gerados pelos sistemas produtivos, sendo assim uma ferramenta poderosa para criar gêmeos digitais que podem transformar a indústria e a sociedade.

No escopo deste projeto, o *PlantSimulation* foi utilizado para o desenvolvimento do GD, realizando tanto as animações dos objetos como a comunicação em tempo real com as estações, além de gerar gráficos e dashboards do processo e salvar os dados para um banco de dados temporário.

3.2.2 MongoDB e MongoDB Atlas

MongoDB é um software de banco de dados NoSQL, que pode ser usado em conjunto com o gêmeo digital para armazenar e processar os dados em tempo real provenientes dos sistemas físicos. MongoDB permite criar modelos de dados flexíveis e adaptáveis às mudanças dos sistemas, sem a necessidade de definir um esquema

fixo. MongoDB também oferece uma alta performance, escalabilidade e disponibilidade para aplicações que exigem uma grande quantidade e variedade de dados. MongoDB possui diversas funcionalidades que o tornam um banco de dados poderoso e versátil para o gêmeo digital, como:

- Suporte a consultas ad hoc, que permitem buscar dados por qualquer campo, com operadores lógicos, expressões regulares, agregações e funções geoespaciais. Isso facilita a análise e a visualização dos dados do gêmeo digital, bem como a detecção de anomalias e padrões;
- Suporte a transações ACID (atomicidade, consistência, isolamento e durabilidade), que permitem executar operações complexas e dependentes em múltiplos documentos e coleções de forma segura e confiável. Isso garante a integridade e a consistência dos dados do gêmeo digital, bem como a sincronização entre o modelo virtual e o sistema físico;
- Suporte a replicação, que permite manter cópias sincronizadas dos dados em vários servidores, aumentando a disponibilidade e a tolerância a falhas. Isso evita a perda ou a corrupção dos dados do gêmeo digital, bem como permite o acesso aos dados por diferentes usuários e aplicações;
- Suporte a *sharding*, que permite distribuir os dados em vários servidores, aumentando a escalabilidade e o desempenho. Isso permite lidar com grandes volumes de dados do gêmeo digital, bem como reduzir o tempo de resposta das consultas e operações.

Para utilizar o MongoDB na nuvem, é necessário de uma solução auxiliar como a plataforma MongoDB Atlas.

O MongoDB Atlas é uma plataforma de dados *multi-cloud* que oferece um pacote integrado de banco de dados em nuvem com serviços para agilizar e simplificar a maneira de trabalhar com os dados. Ele possui benefícios como modelo de dados de documento semelhante ao JSON, API de consulta unificada e a escalabilidade e confiabilidade do MongoDB, sem a preocupação com o gerenciamento da infraestrutura, segurança e recuperação de dados (MONGODB, 2023).

Ele permite a implantação de um banco de dados na nuvem de forma rápida e ágil, usando a interface do usuário do Atlas, a CLI ou um provedor de recursos IaC (*Infrastructure-as-Code*). É possível escolher entre uma instância gratuita, uma instância serverless ou uma configuração de instância dedicada à aplicação. Também pode-se selecionar o provedor de nuvem e a região de acordo com a preferência do usuário, podendo até mesmo distribuir os dados entre diferentes nuvens para obter o melhor desempenho e disponibilidade (MONGODB, 2023).

Além disso, o MongoDB Atlas oferece uma série de recursos e serviços para facilitar o desenvolvimento e a análise de dados, como banco de dados gráficos, data lake, pesquisa, monitoramento, backup, alertas, automação, triggers, funções, charts, entre outros. É possível integrar o MongoDB Atlas com ferramentas e linguagens de programação diversas, como o *Studio 3T,* o Mongo Express, o *Terraform*, o *AWS CloudFormation*, Python, Java, Node.js, etc (MONGODB, 2023).

Dessa forma, o MongoDB Atlas é uma ferramenta ótima para auxiliar no desenvolvimento de gêmeos digitais que podem transformar a indústria e a sociedade. No contexto deste projeto, foi utilizado para armazenar os dados em tempo real das estações de trabalho, permitindo a comunicação com a API (*Application Programming Interface*) do gêmeo digital.

3.2.3 Vercel

O Vercel é uma plataforma de nuvem para o desenvolvimento e a implantação de aplicações web modernas, baseadas em tecnologias como React, Next.js, Node.js e Serverless. O Vercel foi fundado em 2016, com o nome de ZEIT, pelos criadores do framework Next.js, que é um dos mais populares e poderosos para a construção de sites e aplicações web com React. O Vercel tem como missão proporcionar a melhor experiência de desenvolvimento e a melhor infraestrutura para a criação, o escalonamento e a segurança de experiências web mais rápidas e personalizadas(VERCEL. . . , s.d.).

O Vercel oferece uma série de vantagens e benefícios para os desenvolvedores e as empresas que usam a sua plataforma, tais como:

- Uma integração fácil e rápida com os principais serviços de hospedagem de código, como GitHub, GitLab e Bitbucket, permitindo o deploy automático de qualquer repositório Git com um único comando ou um clique;
- Uma arquitetura baseada em componentes e em funções serverless, que possibilita a composição, a reutilização e a otimização de código, reduzindo a complexidade e o tempo de desenvolvimento;
- Uma rede de distribuição de conteúdo (CDN) global e inteligente, que garante uma alta performance, uma baixa latência e uma alta disponibilidade para as aplicações web, independentemente da localização dos usuários;
- Uma observabilidade e uma análise de ponta, que fornecem informações e insights em tempo real sobre o comportamento, o desempenho e o impacto das aplicações web, permitindo a identificação e a correção de problemas, a melhoria da qualidade e a tomada de decisões baseadas em dados;

- Uma segurança e uma privacidade integradas, que protegem as aplicações web e os dados dos usuários, usando recursos como SSL automático, SAML Single-Sign On (SSO), autenticação e autorização, proteção contra ataques DDoS e conformidade com as normas e regulamentações.

O Vercel é uma plataforma flexível e versátil, que pode ser usada para desenvolver e implantar diversos tipos de aplicações web, como por exemplo, sites estáticos, sites dinâmicos, blogs, e-commerces, dashboards, aplicações de inteligência artificial, aplicações de realidade aumentada e virtual, entre outras. O Vercel também conta com uma ampla variedade de integrações com plataformas de conteúdo, de comércio e de banco de dados, facilitando a conexão e a manipulação de dados nas aplicações web.

Além disso, é uma plataforma em constante evolução, que acompanha as tendências e as inovações do mercado e da comunidade de desenvolvimento web, investindo em pesquisa, desenvolvimento, capacitação, suporte e parcerias, buscando oferecer as melhores soluções e as melhores práticas para os seus clientes e usuários. O Vercel é uma plataforma líder e referência no segmento de nuvem para o frontend, sendo usada por milhares de desenvolvedores e empresas de todos os tamanhos e setores, como por exemplo, Airbnb, Netflix, Uber, Twitch, GitHub, Nike, Starbucks, entre outras.

Também visa facilitar, acelerar e melhorar o processo de desenvolvimento e implantação de aplicações web, proporcionando uma experiência de desenvolvimento ágil, colaborativa e produtiva, e uma infraestrutura de nuvem robusta, escalável e segura, que possibilita a criação e a entrega de experiências web mais rápidas, mais personalizadas e mais impactantes, que atendem às expectativas e às necessidades dos usuários e dos negócios na era digital.

No escopo deste projeto, o vercel foi utilizado para a realização *deploy* da API que disponibiliza informações do gêmeo digital, permitindo a comunicação do mesmo com um usuário externo. Foi escolhida por ser de simples utilização, rapidez no *deploy* e por ser gratuita.

4 ESPECIFICAÇÕES DO GÊMEO DIGITAL

Este capítulo tem o intuito de apresentar as especificações do projeto, requisitos funcionais e não funcionais, além de casos de uso.

4.1 REQUISITOS DO SISTEMA

Conforme apresentado por (RABELO, Ricardo J; MAGALHÃES; CABRAL, 2020), de acordo com o modelo de referência ISA-95 de automação industrial apresentado na Figura 8, tem-se que os níveis 0-1-2 estão conectados ao SCF real, enquanto os níveis 3 e 4 funcionam como sistemas-clientes, tanto do SCF quanto do GD. Embora o GD atue simbioticamente e como uma representação virtual do SCF físico, também pode ser considerado um cliente do SCF.

Para simular a situação do SCF em tempo real, um GD precisa considerar e analisar vários fatores na hora de elaborar um projeto de implementação específico, conforme o objetivo principal de cada ator, mostrados nas posições *(i), (ii)* e *(iii)* na Figura 8 (RABELO, Ricardo J; MAGALHÃES; CABRAL, 2020).

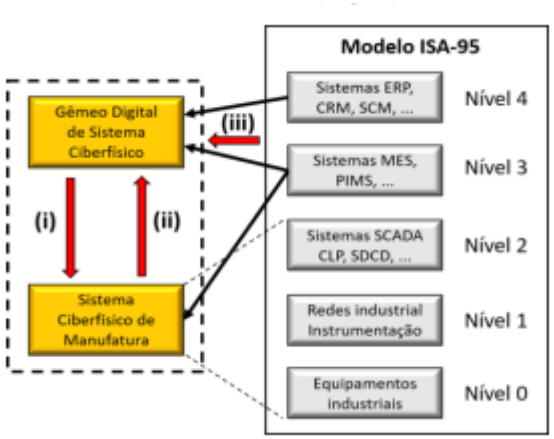


Figura 8 – Gêmeo Digital e Sistema Ciberfísico no modelo ISA-95.

Fonte: Adaptado de (RABELO, Ricardo J; MAGALHÃES; CABRAL, 2020).

Em relação a (*i*), o ator GD tem como interesse principal obter informações do SCF e transmitir informações e instruções para ele, levando em conta aspectos como o conteúdo, o momento, o método e o local de armazenamento dos dados obtidos, bem como o momento e o método de intervenção no SCF. Assim, para cada aspecto, o GD precisa:

- **O que coletar e enviar:** selecionar os dados pertinentes e imprescindíveis do sistema ciberfísico (SCF) para refletir sua atuação e situação em tempo real, além de transmitir informações e instruções para o SCF conforme a necessidade;

- **Quando coletar e enviar:** levar em conta a periodicidade, a quantidade e a sincronia dos dados produzidos pelo SCF e pelos protocolos de comunicação existentes, procurando reduzir o atraso e a perda de dados;
- **Como coletar e enviar:** utilizar métodos de compatibilização, processamento, resumo e aprimoramento de dados para enfrentar as variações de formatos, protocolos, significados e qualidade dos dados que vêm do SCF e de outros sistemas
- **Onde armazenar:** definir onde guardar os dados que são obtidos e transmitidos, levando em conta fatores de preço, eficiência, proteção e acesso, podendo usar soluções próximas ou na internet;
- **Como armazenar:** estabelecer o modelo, a categoria e a forma dos dados guardados, visando simplificar o uso, o questionamento, o estudo e a exibição dos dados, podendo adotar normas livres ou exclusivas;
- **Quando e como atuar:** estipular os cenários, as limitações e os métodos para intervir no SCF, com base nos estudos e experimentos feitos no modelo virtual, levando em conta fatores de tempo real, gestão, conformidade e segurança.

Em relação a (*ii*) e no contexto de GD, lhe enviar informações é o principal interesse do ator SCF, que deve considerar as seguintes questões:

- **Comunicação e transmissão de dados:** comunicar-se com o seu GD, outros sistemas e usuários, usando os meios certos para cada dado e situação;
- **Encapsulamento orientado a serviços:** usar um padrão de comunicações que o faz um provedor de serviços e que esconde as diferenças de TIC (Tecnologia de Informação e Comunicação) de cada SCF, assegurando sua autonomia e baixa dependência na estrutura geral de controle;
- **Autonomia e inteligência:** ser capaz de se auto-gerenciar e de escolher com base em informações e avaliações, tendo a habilidade de trabalhar em conjunto com outros SCFs de maneira equilibrada;
- **Compliance:** seguir o modelo de governança / compliance estabelecido, assegurando que apenas as ações planejadas e permitidas sejam realizadas, obedecendo aos protocolos de segurança, privacidade e qualidade.

Finalmente, no que diz respeito a (*iii*), os clientes estão principalmente interessados na observação do funcionamento do SCF por meio de seu Gerenciador de Dados (GD), na obtenção de diversas informações do GD e no acesso às informações do SCF. Para alcançar esse objetivo, é necessário considerar:

- Como realizar a virtualização do modelo do Sistema de Controle Financeiro (SCF) e quais são os métodos para fazê-lo;
- Quais dados devem ser armazenados no GD;
- Quais elementos devem ser observados tanto no GD quanto no SCF;
- Quais aspectos devem ser analisados e simulados no GD;
- Em que situações, momentos e de que maneira agir no SCF.

Com os requisitos gerais de um GD, é possível então definir o conjunto de requisitos funcionais e não-funcionais deste projeto. De acordo com (VALENTE, 2022), requisitos funcionais definem o que um sistema deve fazer, ou seja, suas funcionalidades. Já os requisitos não-funcionais definem quais as restrições do sistema, isto é, estão relacionados com a qualidade do serviço prestado pelo sistema, incluindo características como desempenho, disponibilidade, níveis de segurança etc.

Dessa forma, considerando os requisitos gerais para implementação de um GD apresentados e as características deste projeto, foram definidos requisitos funcionais e os requisitos não-funcionais atrelados, como pode ser verificado nas Tabelas 1 – 6.

Requisito Funcional		
Visualização do GD	Visualização do sistema SCF em tempo real pelo GD.	
Requisito Não-Funcional	**Obrigatório**	**Permanente**
Usabilidade	**(X)**	**(X)**

Tabela 1 – Requisito funcional e não-funcional.

Fonte: Arquivo Pessoal.

Requisito Funcional		
Receber / Atuar no SCF	Receber informações do SCF e ser capaz de atuar no mesmo via interface ou API no GD.	
Requisitos Não-Funcionais	**Obrigatório**	**Permanente**
Comunicação sem grandes atrasos / estável	**(X)**	**(X)**
QoS: comunicação em tempo real	**(X)**	**(X)**

Tabela 2 – Requisitos funcionais e não-funcionais.

Fonte: Arquivo Pessoal.

Além disso, a Tabela 7 apresenta os requisitos não funcionais do sistema em um contexto geral, sendo todos eles obrigatórios e permanentes além de serem aplicados a todos os requisitos funcionais já apresentados.

Requisito Funcional		
Armazenamento de dados local / nuvem	Armazenar os dados em um banco de dados local e online/nuvem.	
Requisitos Não-Funcionais	**Obrigatório**	**Permanente**
Segurança básica na autenticação à API	**(X)**	**(X)**
Disponibilidade dos dados 24 h/7 d dos repositórios locais e na nuvem.	**(X)**	**(X)**

Tabela 3 – Requisitos funcional e não-funcionais.

Fonte: Arquivo Pessoal.

Requisito Funcional		
Analytics	Disponibilizar dados e processamentos dos mesmos para análises e gerar informações novas.	
Requisito Não-Funcional	**Obrigatório**	**Permanente**
Disponibilidade dos dados via API.	**(X)**	**(X)**

Tabela 4 – Requisitos funcionais e não-funcionais.

Fonte: Arquivo Pessoal.

Requisito Funcional		
Simulação de eventos passados	Reproduzir simulações de intervalos de operação passados no GD.	
Requisito Não-Funcional	**Obrigatório**	**Permanente**
Disponibilidade dos dados para reprodução da simulação.	**(X)**	**(X)**

Tabela 5 – Requisitos funcional e não-funcional.

Fonte: Arquivo Pessoal.

Requisito Funcional		
Alarmes de erros e logs	Gerar alarmes de erros e logs do estado de operação.	
Requisito Não-Funcional	**Obrigatório**	**Permanente**
Monitoramento do estado de operação.	**(X)**	**(X)**

Tabela 6 – Requisitos funcional e não-funcional.

Fonte: Arquivo Pessoal.

4.2 CASOS DE USO

Com base nos requisitos funcionais e não-funcionais levantados na Seção 4.1 foram modelados os casos de uso deste projeto.

Requisitos Não-Funcionais	Descrição
Integrabilidade	Capacidade de integrar o GD a novos serviços / sistemas externos ou internos.
Interoperabilidade	Capacidade de aceitar novos serviços / aplicações ao GD.
Escalabilidade	Capacidade de escalonar o GD conforme às necessidades.
Lock-in	Diminuir o máximo possível limitações das aplicações / ferramentas envolvendo o GD.

Tabela 7 – Requisitos não-funcionais gerais do GD.

Fonte: Arquivo Pessoal.

De acordo com (UML:. . . , 2022), os casos de uso descrevem como os usuários interagem com as funcionalidades do sistema além de fornecer uma visão externa do sistema. No entanto, não especifica como o sistema deve realizar suas funcionalidades, isto é, não apresenta quais são os procedimentos realizados a fim de realizar cada funcionalidade do sistema.

A Figura 9 apresenta o diagrama de casos de uso modelado para este projeto.

Conforme mostra a Figura 9, existem quatro possíveis usuários para o sistema, sendo eles:

- Administrador remoto: usário afastado do local onde se encontra o SCF com privilégios de administrador capaz de interagir com o SCF via API e adicionar outros usuários;
- Operador local: usuário com acesso aos principais recursos do GD, como visualização do SCF em tempo real, visualização de métricas e gráficos de operação em tempo real, reprise de simulações passadas além do envio de comandos ao SCF via *PlantSimulation* e obtenção das informações de operação;
- Operador remoto: usuário afastado fisicamente do local onde se encontra o SCF, sem privilégios de administrador, consegue apenas obter informações de operação em tempo real via API;
- Serviço externo: tal como operador remoto, pode obter as informações de operação em tempo real via API.

Embora os atores "Operador Remoto"e "Serviço Externo"desempenhem a mesma ação no caso de uso, eles possuem propósitos diferentes de ação. O operador remoto foi modelado com intuito de simular diferentes níveis de autenticação no sistema, contendo operador remoto e administrador remoto. Já o serviço externo foi modelado

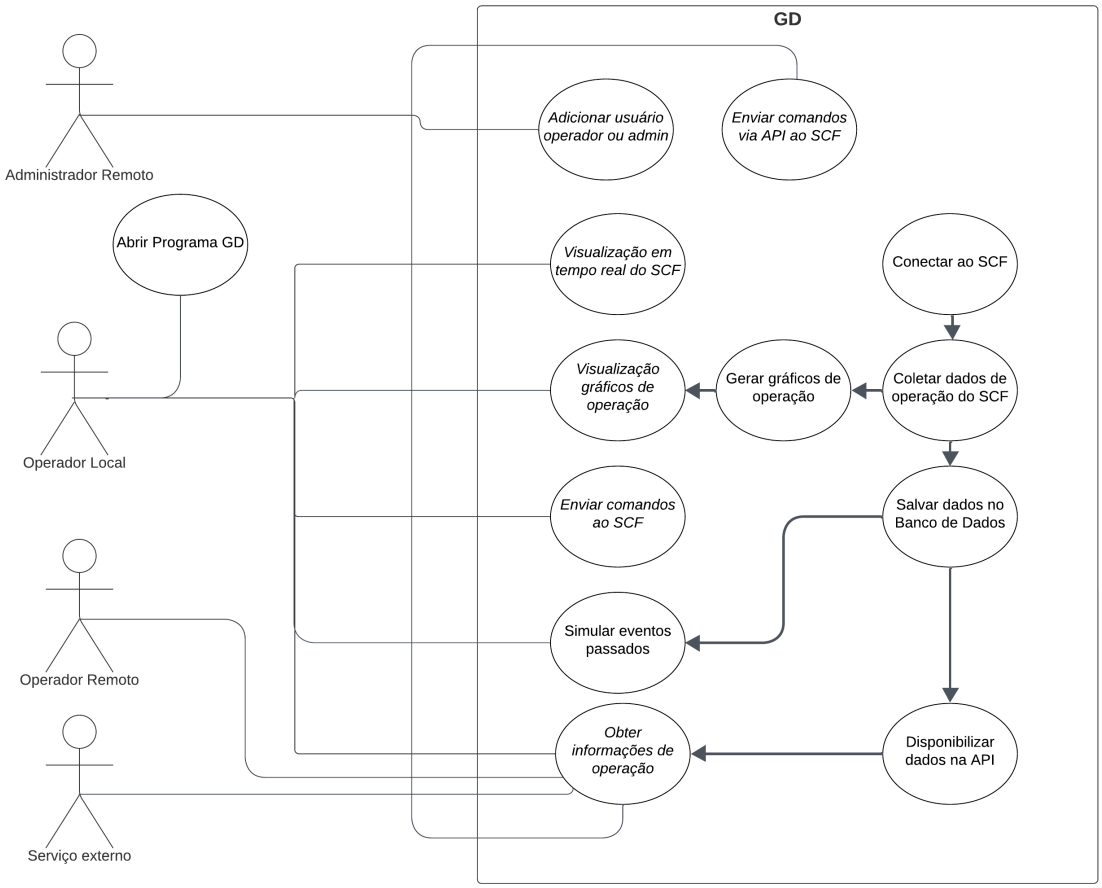


Figura 9 – Diagrama de casos de uso do sistema.

Fonte: Arquivo pessoal.

com intuito de representar uma aplicação/serviço que deseja utilizar as informações do sistema para utilização em seu próprio serviço e habilitar a integração entre diferentes serviços e agentes. Dessa forma, as atribuições para operador remoto e serviço externo podem expandir e divergir conforme a necessidade em futuros projetos.

Com os casos de uso estabelecidos, são desenvolvidos os diagramas de sequência com intuito de mostrar as mensagens trocadas entre os objetos durante a execução do sistema, bem como o estado e o comportamento de cada objeto em cada momento. Os diagramas UML de sequência são uma forma de representar graficamente as interações entre os objetos de um sistema em ordem temporal e podem ser usados para modelar processos, funções ou serviços complexos, mostrando como eles se realizam e quais são os resultados esperados (BOOCH; RUMBAUGH; JACOBSON, 2012).

A Figura 10 apresenta o diagrama de sequência para o envio de comando do administrador remoto. O usuário do tipo administrador remoto realiza o *login* na API, que por sua vez verifica o usuário e a senha no banco de dados na nuvem MongoDB, caso esteja correto, o usuário é autenticado na API. Após a autenticação, o usuário

envia o comando desejado a realizar no SCF via API, a API adiciona o comando no MongoDB. O middleware é o componente que verifica constantemente se um novo comando é adicionado no MongoDB, caso seja, ele envia o comando para o SCF que o executa. Por fim, é retornada a execução do comando até o administrador remoto.

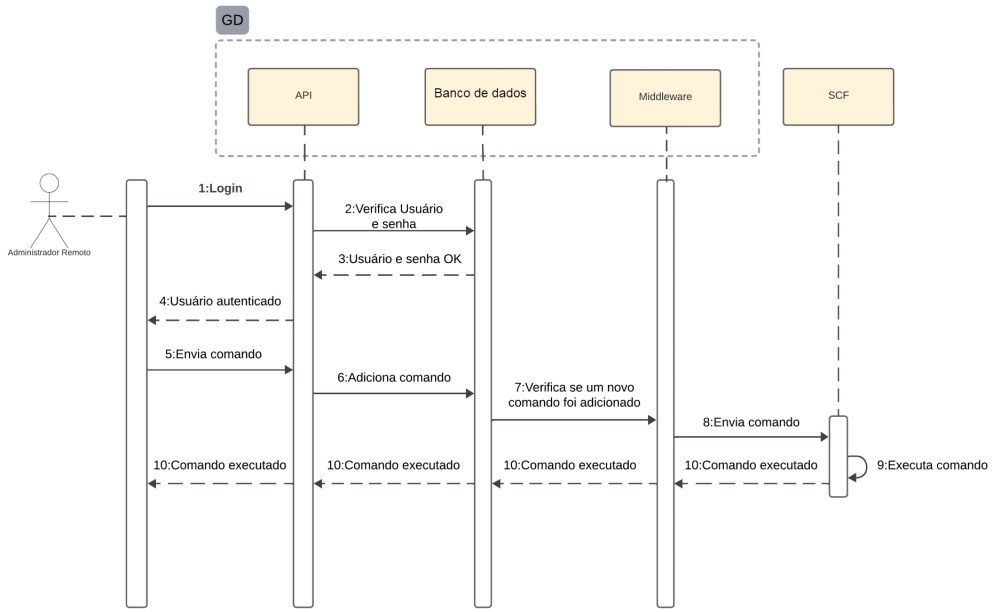


Figura 10 – Diagrama de Sequência do envio de comando do administrador remoto.

Fonte: Arquivo pessoal.

A Figura 11 mostra o diagrama de sequencia da obtenção dos dados de operação via API. Esse caso de uso é comum a todos os usuários do sistema. É necessário que o usuário realize o *login* na API e tenha seu usuário autenticado conforme descrito no caso de envio de comando do administrador remoto apresentado na Figura 10. Após autenticação, o usuário envia a requisição para obtenção das informações de operação, a API consulta o MongoDB que retorna os dados da operação e por fim a API retorna esses dados ao usuário. A Figura 12 apresenta uma camada mais profunda nessa etapa (Comunicação SCF/GD), onde o Cliente GD está constantemente se comunicando e obtendo dados de operação do SCF e salvando esses dados no MongoDB.

A Figura 16 apresenta os componentes que constituem o GD desenvolvido neste projeto. Cada caso de uso pode utilizar um ou mais dos componentes do GD, dessa forma, para esclarecer o entendimento dos diagrams de sequência, foram adicionados caixas tracejadas em volta dos objetos que compõe o GD.

As Figuras 13, 14 e 15 apresentam respectivamente os diagramas de sequência dos casos de uso de visualização dos gráficos de operação em tempo real, visualização do SCF em tempo real e simular eventos passados (reprise de simulação). Conforme mostra a Figura 13, para a visualização dos gráficos de operação em tempo real, o

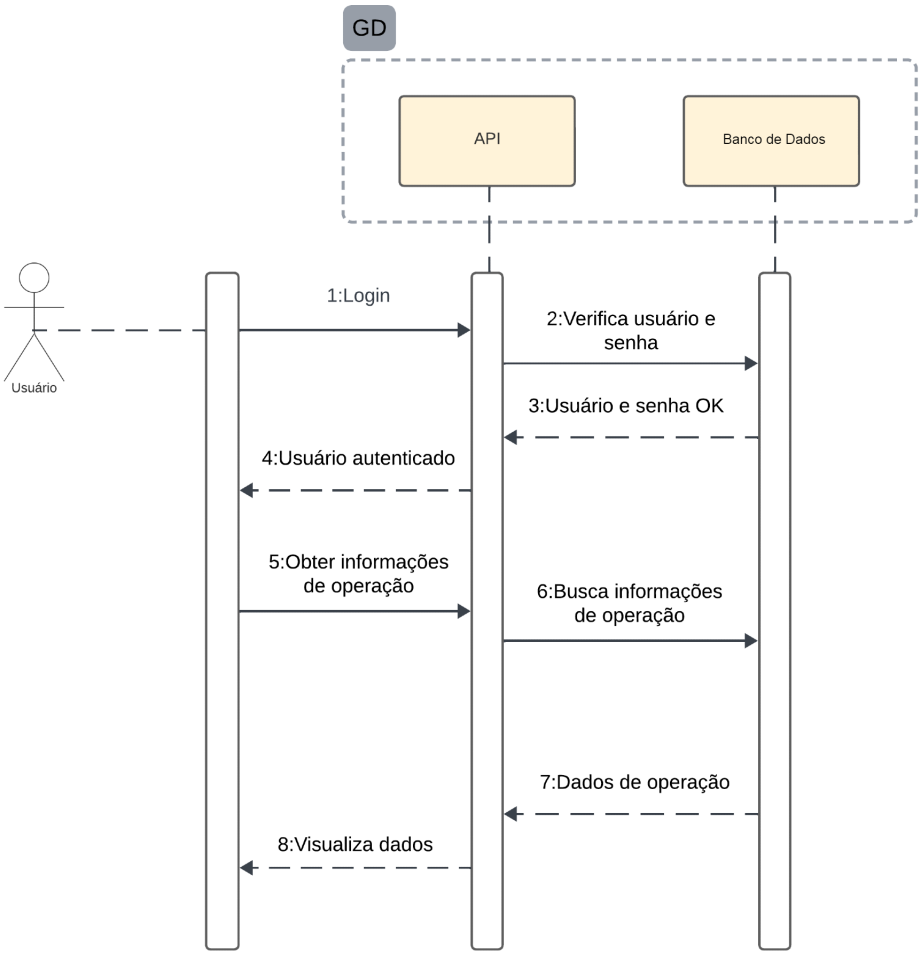


Figura 11 – Diagrama de Sequência da obtenção dos dados de operação.

Fonte: Arquivo pessoal.

operador local executa a simulação do GD que se conecta ao SCF, se a conexão estiver funcionando adequadamente, o operador local pode solicitar os gráficos ao GD. Por fim, o GD gera os gráficos em tempo real e apresenta ao usuário. O diagrama de sequencia apresentado na Figura 14 possui os primeiros passos iniciais que o diagrama de visualização dos gráficos de operação, e difere ao final no quarto passo, onde o GD apresenta a visualização do SCF em tempo real.

No diagrama de sequencia apresentado na Figura 15, são utilizados dois componentes do GD não apresentados nos diagramas de sequencia anteriores, o banco de dados SQLite e o software PlantSimulation (embora não explicitado em outros diagramas, ele é a ferramenta principal nos casos de uso de visualização do SCF em tempo real e da visualização dos gráficos de operação). Para realizar o reprise da simulação, o operador local abre o *PlantSimulation* e se conecta ao SQLite. Se a comunicação estiver estabelecida adequadamente, o operador informar o intervalo de datas no qual ele gostaria de rever a simulação e o *PlantSimulation* envia a busca ao SQLite, que

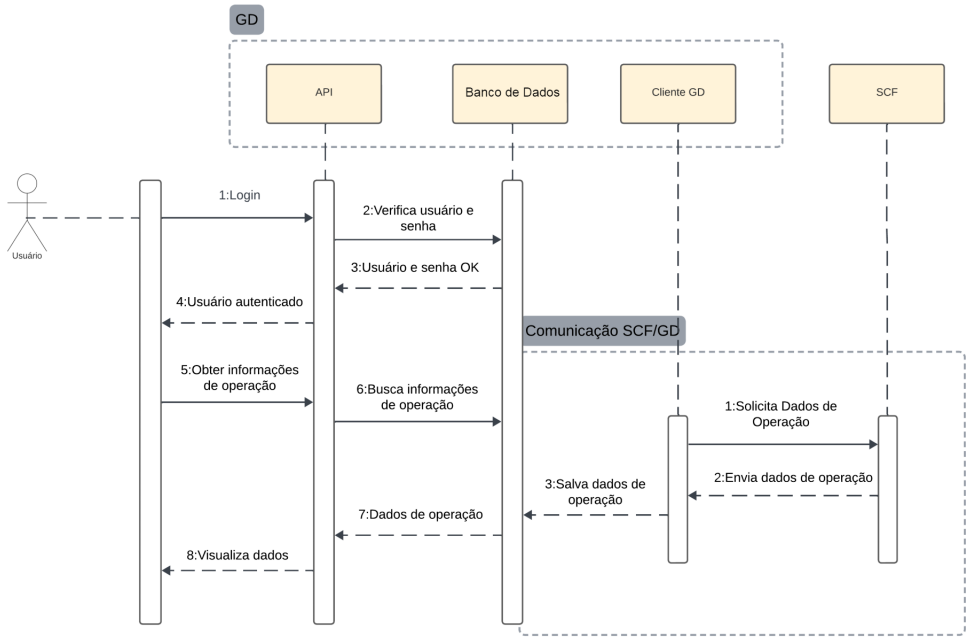


Figura 12 – Diagrama de Sequência da obtenção dos dados de operação aprofundado.

Fonte: Arquivo pessoal.

busca os dados e os trata para envia-losna forma correta para que o *PlantSimulation* possa realizar o reprise da simulação e mostrar ao operador local.

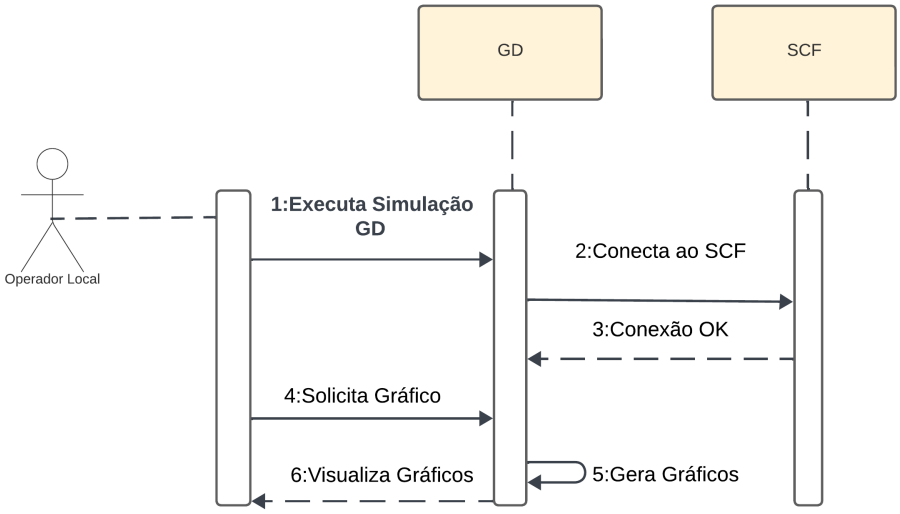


Figura 13 – Diagrama de Sequência da visualização dos gráficos de operação.

Fonte: Arquivo pessoal.

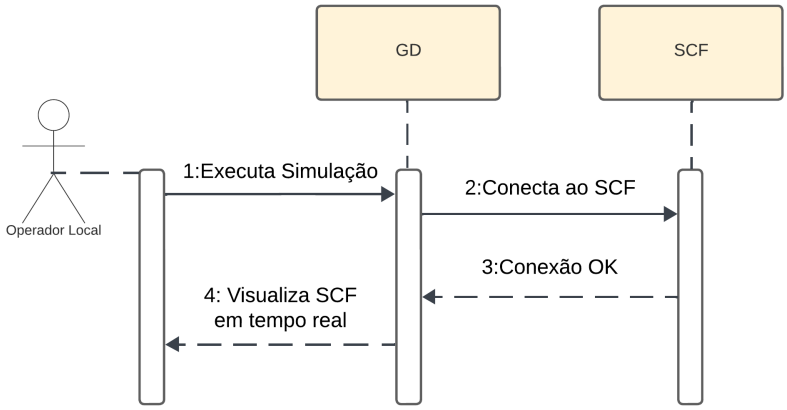


Figura 14 – Diagrama de Sequência da visualização do SCF em tempo real.

Fonte: Arquivo pessoal.

4.3 ARQUITETURA DO PROJETO

Esta seção apresenta a arquitetura geral desenvolvida para este projeto.

A Figura 17 apresenta o diagrama de *deployment* do sistema, mas com ele pode-se apresentar também a arquitetura do projeto, composta pelos três módulo principais, sendos esses o GD, o SCF e a nuvem. Cada bancada da planta possui um CLP Siemens e comunica-se com as estações adjacentes. Junto disso, cada bancada também se conecta à um computador na mesma rede, o qual é responsável por coletar os dados via cliente e executar o software PlantSimulation para visualização do gêmeo digital, além da conexão com o software TIA Portal para execução do programa nos CLPs. Os dados são salvos pelo cliente utilizando tanto um banco de dados local SQLite quanto em um banco de dados NoSQL na nuvem, o MongoDB Atlas. O middleware entre o GD e a API verifica constantemente se há alterações realizadas na coleção de comandos no MongoDB, e caso haja se comunica com o cliente do GD para realizar a ação necessária. Por fim, a API se comunica com o banco de dados na nuvem via requisições REST realizadas pelo usuário.

A Figura 18 apresenta o diagrama UML de classes modelado para este projeto.

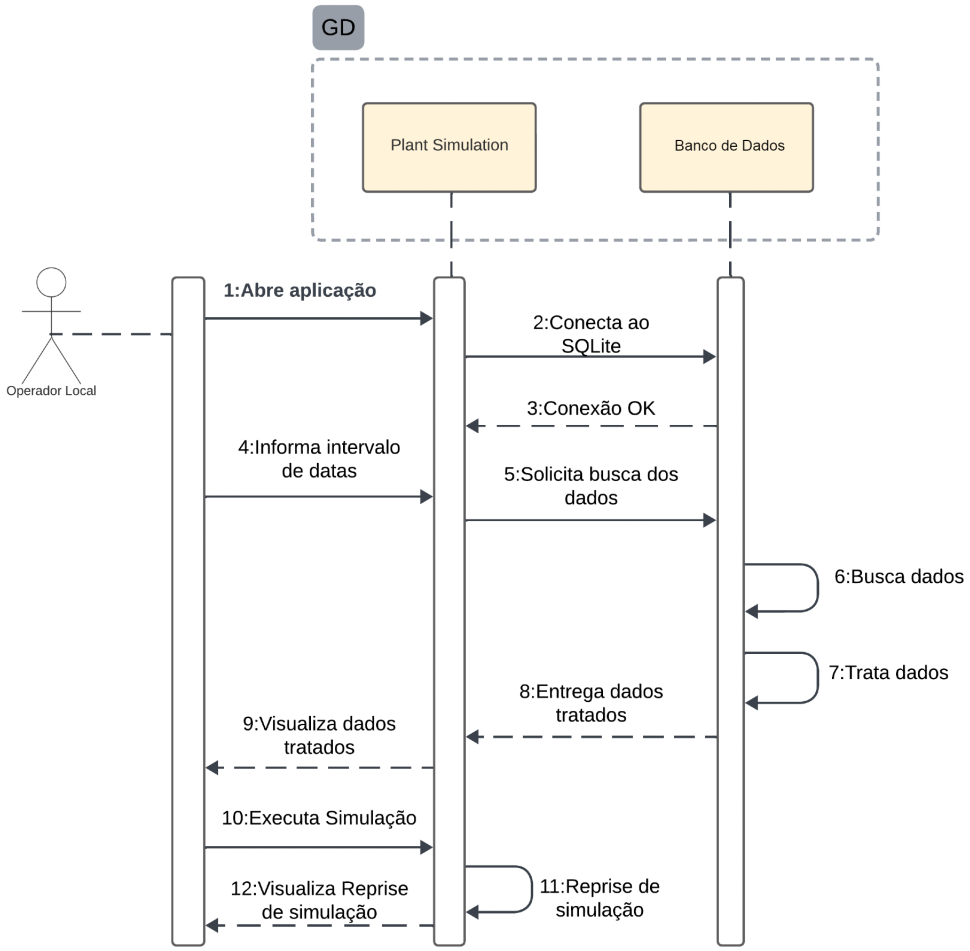


Figura 15 – Diagrama de Sequência do reprise de simulação.

Fonte: Arquivo pessoal.

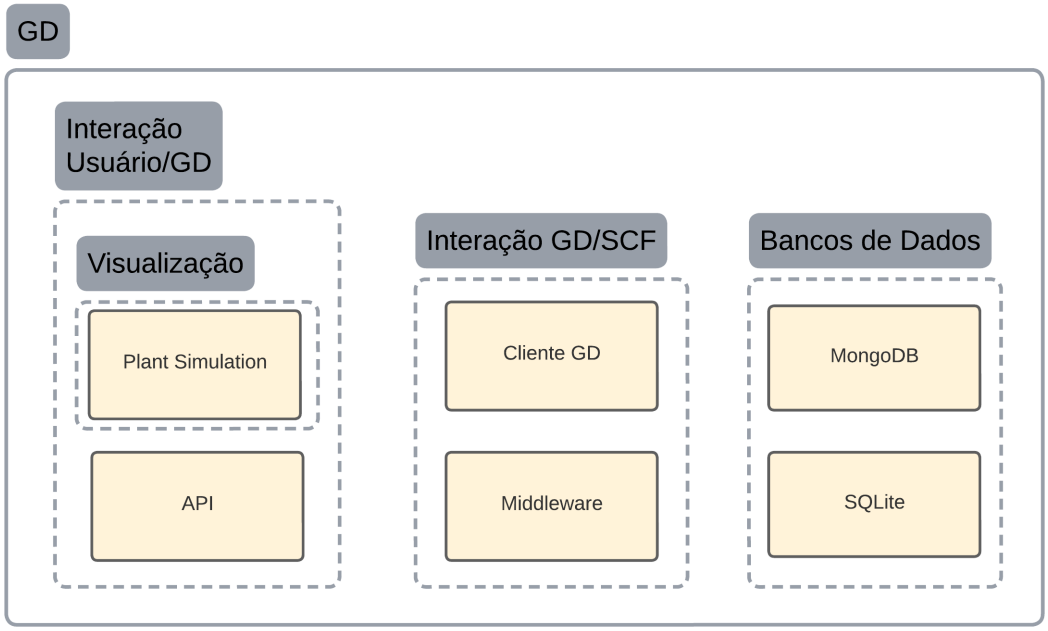


Figura 16 – Componentes que constituem o GD.

Fonte: Arquivo pessoal.

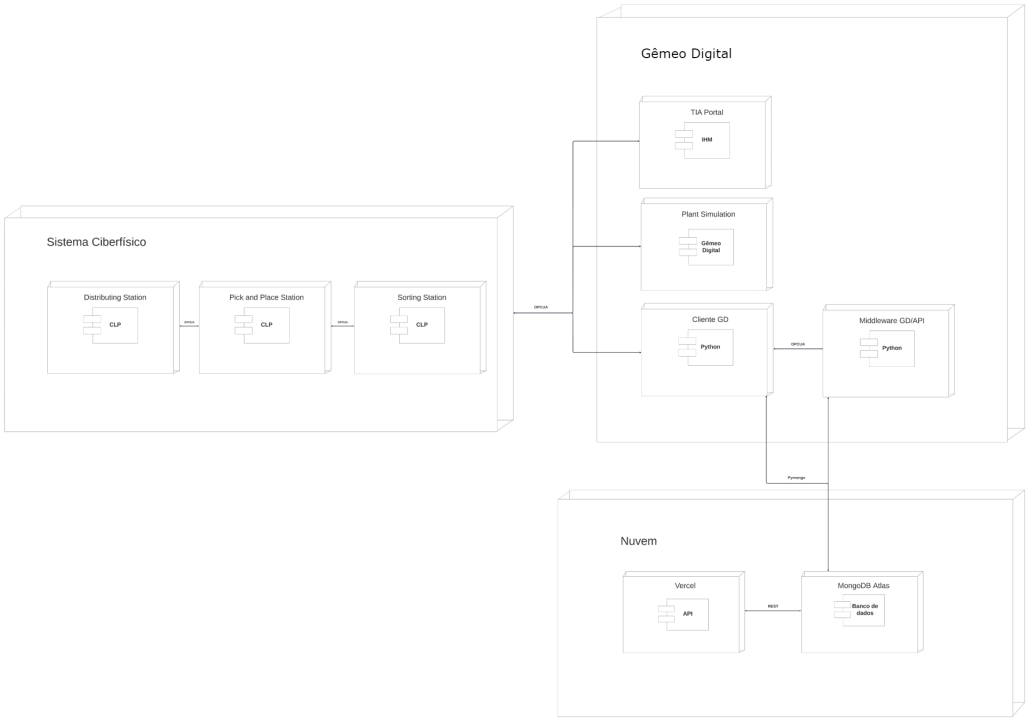


Figura 17 – Diagrama de *Deployment* do sistema.

Fonte: Arquivo pessoal.

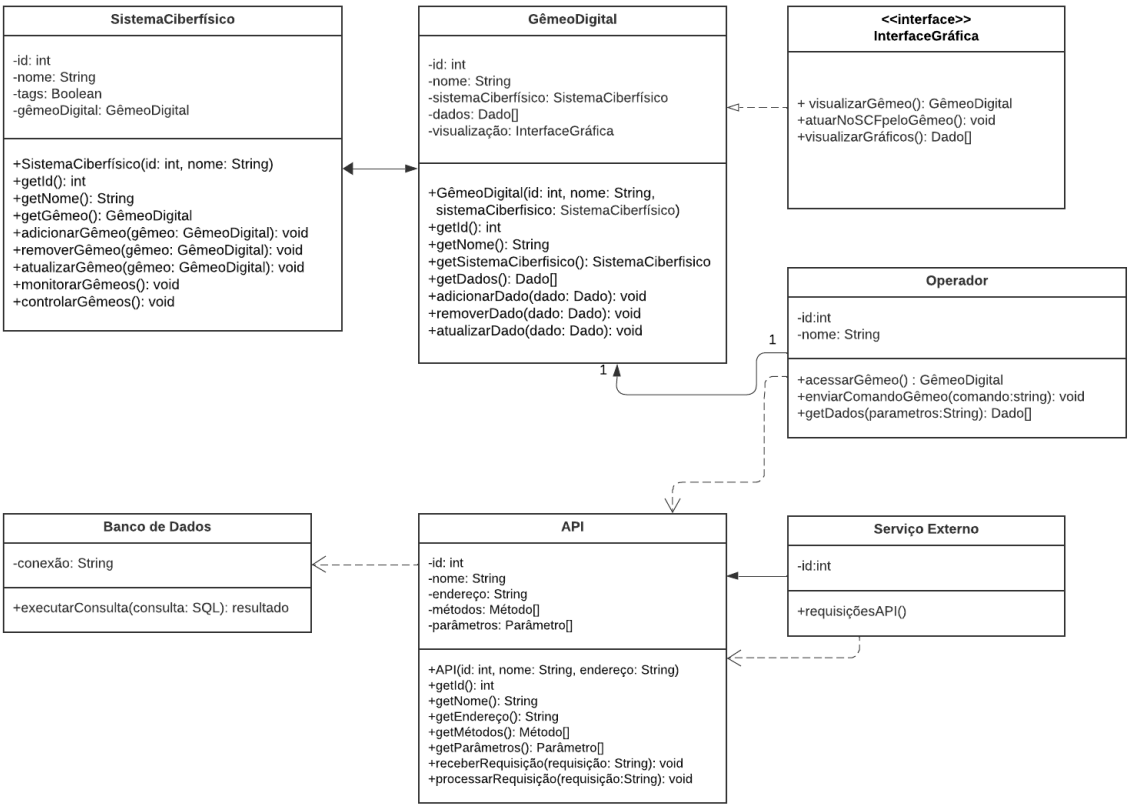


Figura 18 – Diagrama de Classes do sistema.

Fonte: Arquivo pessoal.

5 IMPLEMENTAÇÃO

Este capítulo mostra a implementação do projeto e discute os resultados obtidos.

5.1 BACK-END

Nesta seção são apresentados o desenvolvimento dos principais serviços da estrutura de *back-end* do gêmeo digital como os serviços de processamento e armazenamento dos dados, alarmes e *logs*, além de aspectos de segurança, integrabilidade e interoperabilidade implementados. Todos os códigos desenvolvidos ao longo deste projeto podem ser encontrados em repositórios do github na conta do autor.

5.1.1 Cliente

De forma a habilitar a implementação dos requisitos funcionais e não funcionais apresentados na Seção 4.1, foi desenvolvido um script em python que cria um cliente conectado aos servidores OPC-UA das bancadas, dessa forma implementando o componente do módulo do GD conforme apresentado anteriormente na Figura 17.

No cliente, os dados coletados das tags são armazenados em um repositório MongoDB e um banco de dados local SQLite, dessa forma cumprindo os requisitos funcionais e não funcionais apresentados na Tabela 3. Além disso, o cliente também esta conectado ao *middleware* (discutido mais a fundo na Subseção 5.1.2) que verifica no MongoDB na nuvem se há algum comando recente enviado via API para o SCF, caso haja, o cliente envia o comando para o servidor OPC-UA das bancadas.

Devido à natureza da verificação no banco de dados ser um evento preso em um *loop*, a solução mais simples encontrada para habilitar a interação de um usuário administrador com o SCF (conforme apresentado no diagrama de sequência de envio do comando na Figura 10) e manter o cliente rodando em um outro *loop* que está constantemente atualizando o valor das tags, foi executar os dois loops em diferentes scripts utilizando *multithreading*.

O *multithreading* é a capacidade de uma unidade central de processamento (CPU) ou um núcleo de executar múltiplas *threads* de execução simultaneamente, com o suporte do sistema operacional. Uma *thread* é uma sequência independente de instruções que pode executar em paralelo com outras *threads* que fazem parte do mesmo processo raiz. O objetivo principal do *multithreading* é aumentar a utilização de um único núcleo, usando paralelismo em nível de *thread* e paralelismo em nível de instrução para aumentar o rendimento (MARIO NEMIROVSKY, 2013)

Dessa forma, o *middleware* roda paralelamente com o cliente do GD e interage com o mesmo caso seja necessário. A Figura 19 apresenta um diagrama representando o funcionamento do cliente e do *middleware* em *multithread*.

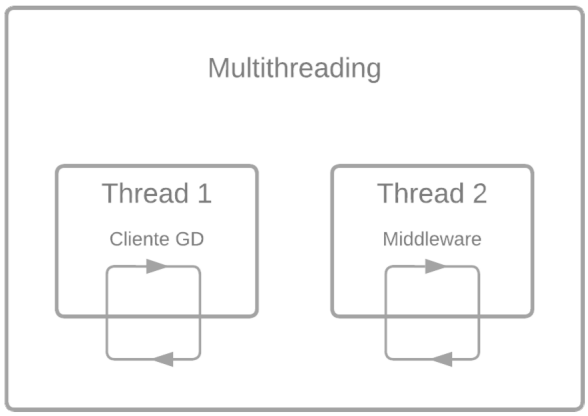


Figura 19 – Diagrama de representação do multithreading.

Fonte: Arquivo pessoal.

No cliente também há a geração dos principais logs de operação do sistema, que indicam os principais eventos e a data e hora em que ocorreram, além de um alarme indicando quando a bancada for paralizada, implementando assim os requisitos apresentados na Tabela 6.

5.1.2 Middleware

De acordo com (BRASIL - UAB | IF SUL-RIO-GRANDENSE, 2023), um *middleware* é um software que facilita a integração e a comunicação entre aplicações, sistemas e tecnologias diferentes. Ele simplifica o trabalho dos programadores na criação de conexões necessárias em sistemas distribuídos. Ele também fornece uma solução para melhorar aspectos como qualidade do serviço, segurança, mensagens, serviço de diretório, etc.

Um *middleware* é usado para transferir informações e dados entre programas que usam diferentes protocolos de comunicação, plataformas e sistemas operacionais. Ele é formado por módulos que possuem APIs de alto nível, que permitem a sua integração com aplicações escritas em várias linguagens de programação, e interfaces de baixo nível, que garantem a sua independência em relação ao dispositivo. Seu objetivo é esconder a heterogeneidade e oferecer um modelo de programação mais produtivo para os programadores de aplicações. Ele é composto por um conjunto de processos ou objetos em um grupo de computadores, que se comunicam entre si para implementar comunicação e suporte para compartilhamento de recursos e aplicações distribuídas. Normalmente, o *middleware* do lado do cliente é implementado pelo sistema operacional, que possui bibliotecas que executam todos os recursos para a comunicação pela rede (BRASIL - UAB | IF SUL-RIO-GRANDENSE, 2023).

De forma a habilitar a interação de um usuário administrador com o SCF, foi

necessário criar um serviço que verifica constantemente se um comando foi enviado via API no banco de dados na nuvem. Para a construção do serviço *middleware*, foi criado um script em python que consiste em um *loop* que verifica a cada um segundo se um comando novo foi inserido na coleção de comandos.

Caso um comando seja adicionado a coleção de comandos, o *middleware* se comunica com o cliente OPC-UA da bancada desejada e atualiza o valor das tags para efetivar a ação de comando desejado. A Figura 20 apresenta uma esquematização do sistema com o *middleware*.

Figura 20 – Esquematização do sistema com o *middleware*.

Fonte: Arquivo pessoal.

Essa solução foi adotada por se apresentar a mais simples e eficiente de implementar, outras soluções foram cogitadas como a criação de um *webhook*, que verificaria se um determinado *endpoint* da API foi chamado e notificaria o cliente GD caso ocorresse a chamada. Porém, o *webhook* não se apresenta como a melhor solução, visto que o cliente GD não é um serviço que possui chamadas HTTP, sendo necessário implementar algum outro serviço/*broker* para poder enviar a notificação da chamada do *endpoint*.

5.1.3 Armazenamento dos dados

Para o armazenamento dos dados foram utilizados dois bancos de dados diferentes, o MongoDB e o SQLite.

O MongoDB foi selecionado pois permite a integração do gêmeo digital com uma API *online* (apresentada de forma detalhada na Subseção 5.1.4), além de aspectos como a fácil utilização, familiaridade do autor com a tecnologia e por habilitar requisitos como escalabilidade, integrabilidade e interoperabilidade apresentados na Tabela 7. A Figura 21 apresenta a estrutura do banco de dados MongoDB desenvolvida. Conforme

pode ser vista na mesma, há um banco de dados para o GD e outro para Usuários. O banco de dados GD possui a coleção de comando e uma coleção para cada bancada do SCF. A coleção de comando é responsável por armazenar os comandos enviados via API para o SCF, contendo o nome, o dia e hora em que foi realizado, o tipo de comando, qual bancada irá atuar e o usuário que realizou a requisição do *endpoint*. A coleção de bancada possui os dados do nome da bancada, o dia e a hora em que foram salvos e a lista de tags com seus respectivos valores.

Já no banco de dados do Usuário, há a coleção de usuário que contém as informações de nome do usuário, cargo, email e a senha encriptada.

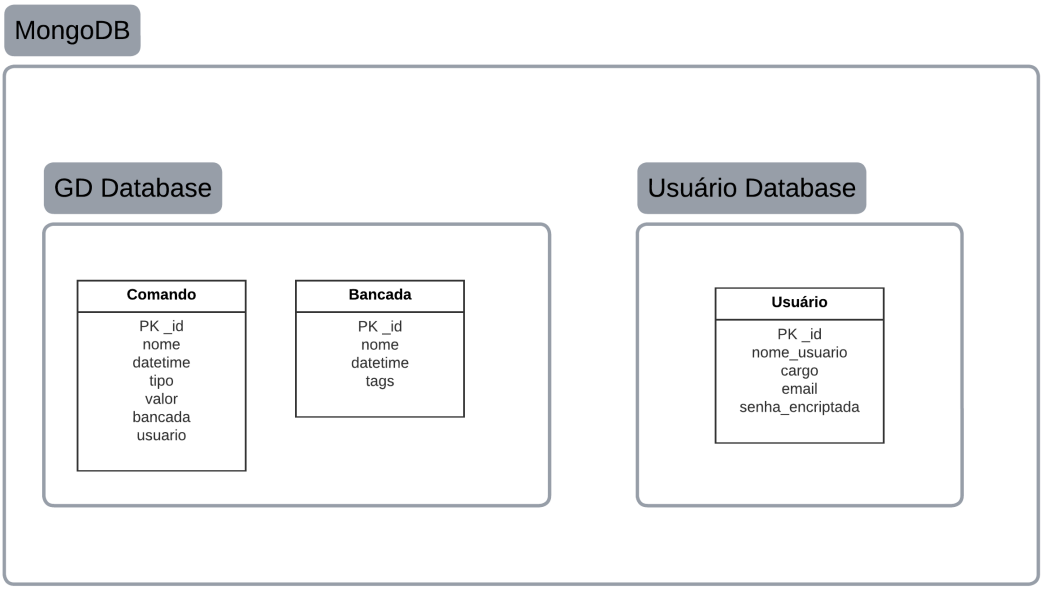


Figura 21 – Estrutura do banco de dados MongoDB.

Fonte: Arquivo pessoal.

A escolha do banco de dados SQLite para o armazenamento de dados local foi devido ao aspecto de integração com o software *PlantSimulation*, visto que o mesmo utiliza apenas bancos de dados relacionais. Assim, com intuito de implementar o requisito de reprise de simulação apresentado na Tabela 5, e com as ferramentas disponibilizadas pelo próprio *PlantSimulation*, optou-se pela utilização do SQLite. A Figura 22 apresenta a estrutura do banco de dados desenvolvido. Primeiramente os dados das tags são salvos em uma tabela principal denominada OPCUA-Data, em seguida são criadas tabelas separadas para cada Tag, contendo seu histórico de valores, o dia e hora, além da diferença de tempo normalizada (timestamp_delta). Por fim, é criada a tabela de peça que será utilizada no *PlantSimulation* para criação das peças ao realizar um reprise de simulação, que contém as informações de operação sobre a peça manufaturada.

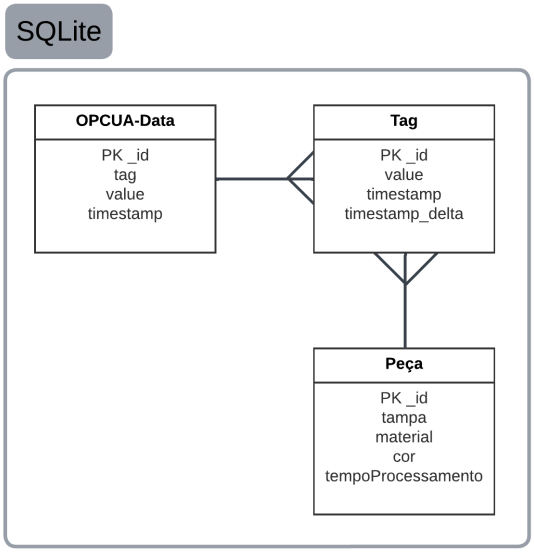


Figura 22 – Estrutura do banco de dados SQLite.

Fonte: Arquivo pessoal.

A implementação de ambos os bancos de dados foi realizada no *script* do cliente e com as ferramentas disponibilizadas no *PlantSimulation*.

5.1.4 API

A API é o serviço principal por habilitar a implementação dos requisitos de interoperabilidade e integrabilidade apresentados na Tabela 7, e faz parte do módulo de nuvem conforme apresentado na Figura 17. Para a construção da mesma, foi desenvolvido um projeto em python contando com bibliotecas que agilizam o desenvolvimenteo de uma API, como *fastapi*, e para o *deployment* dela e sua utilização na internet de forma *online* foi utilizado o serviço de hospedagem em nuvem Vercel. Para a realização do *deployment* da API, basta apenas conectar o repositório do projeto a uma conta na plataforma Vercel e realizar o *deploy* pela plataforma.

O projeto foi desenvolvido com base no padrão *Model-View-Controller* (MVC). Esse padrão de API é uma forma de organizar o desenvolvimento de uma API seguindo os princípios do MVC e visa separar as responsabilidades e facilitar a manutenção, o teste e a reutilização do código (SILVA, J. A. da, 2015). Os componentes principais do padrão MVC de API são:

- ***Model:*** É a camada que representa os dados da aplicação, como as entidades, os atributos, as regras de negócio, as validações, etc. O modelo é responsável por interagir com o banco de dados ou outras fontes de dados, e fornecer uma

abstração para o controlador e a vista. O modelo pode usar bibliotecas como o Pydantic, o SQLAlchemy, o Peewee, etc. para definir e manipular os dados (DEVMEDIA, 2020);

- ***View:*** É a camada que representa a interface da aplicação, como as páginas web, os templates, os formulários, os gráficos, etc. A vista é responsável por renderizar os dados do modelo de forma amigável e interativa para o usuário. A vista pode usar bibliotecas como o Jinja2, o Bootstrap, o Chart.js, etc. para criar e estilizar o HTML, o CSS e o JavaScript da aplicação (CÓDIGO, 2011);
- ***Controller:*** É a camada que representa a lógica da aplicação, como as rotas, os parâmetros, as requisições, as respostas, as autenticações, as autorizações, etc. O controlador é responsável por receber as entradas do usuário, comunicar-se com o modelo e a vista, e retornar as saídas para o usuário. O controlador pode usar bibliotecas como o FastAPI, o Flask, o Django, etc. para definir e gerenciar a API da aplicação (ILUSTRADEV, 2019).

Ele pode trazer vários benefícios para o desenvolvimento de uma API, como:

- **Modularidade:** Ao separar as responsabilidades em camadas distintas, o padrão MVC de API permite que o código seja mais modular e independente. Isso facilita a alteração, a adição e a remoção de funcionalidades sem afetar outras partes da aplicação. Por exemplo, é possível mudar o banco de dados sem alterar a vista, ou mudar o template sem alterar o controlador (SOUZA, 2016);
- **Testabilidade:** Ao separar as responsabilidades em camadas distintas, o padrão MVC de API permite que o código seja mais testável e confiável. Isso facilita a criação, a execução e a automação de testes unitários, de integração e de aceitação para cada camada da aplicação. Por exemplo, é possível testar o modelo sem depender da vista, ou testar a vista sem depender do controlador (WAGON, 2020);
- **Reutilização:** Ao separar as responsabilidades em camadas distintas, o padrão MVC de API permite que o código seja mais reutilizável e escalável. Isso facilita o aproveitamento, a extensão e a integração de código entre diferentes aplicações ou módulos. Por exemplo, é possível reutilizar o modelo em outras APIs, ou reutilizar a vista em outros projetos.

Dessa forma, o padrão MVC é uma forma popular e eficiente de desenvolver APIs com python, embora não seja a única. Existem outros padrões e arquiteturas que podem ser usados para criar APIs, como o REST, o GraphQL, o RPC, o Microservices, etc. Cada padrão tem suas vantagens e desvantagens, e no contexto deste projeto,

o padrão MVC foi selecionado devido à familiaridade do autor com o padrão e ágil implementação do mesmo.

Visto que o foco deste projeto no desenvolvimento da API é o de permitir a implementação dos requisitos funcionais e não-funcionais apresentados nas Tabelas 1 – 7, o principal foco no padrão MVC foi dado para os componentes de modelos 5.1.4.1 e controladores 5.1.4.2, e para o componente de vista foi desenvolvido uma tela de login para exemplificação apresentada na Figura 23 de forma a permitir melhorias e atualizações em trabalhos futuros que tenham este projeto como base.

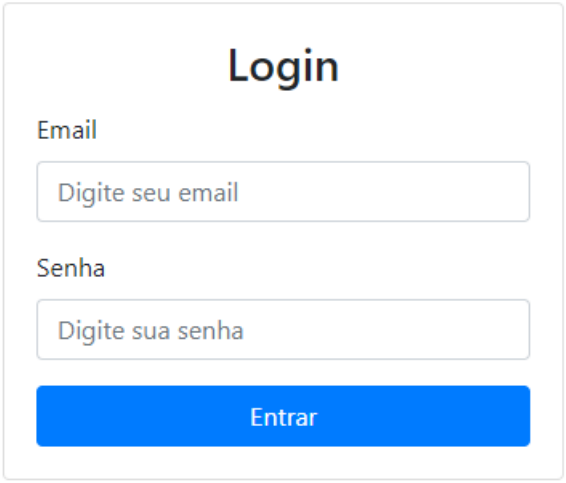


Figura 23 – Tela de login.

Fonte: Arquivo pessoal.

Para a utilização da API, é necessário que o usuário realize *login* na mesma a partir da página principal da aplicação encontrada no endereço "https://gd-api-liard.vercel.app/", ou pode-se realizar o *login* direto via linhas de comando e *scripts em python* como mostra a Figura 24. A partir da Figura 24, pode-se visualizar também a maneira de utilizar a API para obter as informações e realizar operações desejadas, basta realizar uma requisição http com o endereço URL sendo constituído do endereço onde esta hospedado a API (https://gd-api-liard.vercel.app/), concatenando com o *endpoint* desejado (os *endpoints* implementados são apresentados e descritas suas funcionalidades na Tabela 9).

A Figura 25 mostra o retorno que o usuário recebe ao realizar o *login* na API, ele recebe um *acess_token* que será passado no cabeçalho (*header*) das próximas chamadas dos *endpoints*. A Figura 26 apresenta um *script* em python que exemplifica o fluxo de chamadas na API e como realizá-las, sendo feito primeiramente o *login*, obtendo o *access_token* e utilizando ele no *header* da requisição de enviar o comando de *Stop* para a bancada *Distributing*. A Tabela 8 apresenta os *endpoints*, seus res-

pectivos tipos de requisição HTTP e seus parâmetros necessários para utilização. Os parâmetros *start* e *end* no parâmetro de *query* de algumas requisições se referem as datas de ínicio e fim desejadas. Por exemplo, caso deseje-se saber o tempo total de trabalho de uma determinada bancada, serão passadas as datas de início '2024-10-01' e de fim '2024-10-02', e a resposta retornará o tempo total de trabalho da estação desejada entre os dias 1 de janeiro de 2024 e 2 de janeiro de 2024.

```
atr1 = {"username": "Exemplo", "password": "Exemplo_Senha"}
r1 = requests.post('https://gd-api-liard.vercel.app/users/login', data=atr1)
```

Figura 24 – *Script* em python para realizar *login* na API.

Fonte: Arquivo pessoal.

{"access_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJVc2VyMiIsInJvbGUiOiJhZG1pbiIsImV4cCI6MTcwNTIwMjQ4OH0.-wEh8mRrz58SsesEfUA9g0Jnt1b3a_6Z_fJXlRlB2WE","token_type":"bearer"}

Figura 25 – Retorno da chamada de *login* na API.

Fonte: Arquivo pessoal.

```
atr1 = {"username": "Exemplo", "password": "Exemplo_Senha"}
r1 = requests.post('https://gd-api-liard.vercel.app/users/login', data=atr1)

token = r1.json()["access_token"]
headers = {"Authorization": f"Bearer {token}"}
r = requests.post(f'https://gd-api-liard.vercel.app/gd/stop_station',  params='station=Distributing', headers=headers)
```

Figura 26 – Exemplificação do fluxo de chamadas na API e como realiza-las.

Fonte: Arquivo pessoal.

5.1.4.1 Modelos

Os modelos desenvolvidos levam em consideração tanto aspectos do GD, como as bancadas e tags, quanto dos usuários com permissões específicas de acordo com a função do usuário e o processo de autenticação.

A Figura 27 mostra a representação dos modelos das bancadas e das tags, com seus devidos atributos e métodos de validação dos dados. Os modelos são usados como base nas respostas e requisições dos *endpoints* da API, assim como a representação dos dados presentes no banco de dados na nuvem.

Os modelos relacionados aos usuários e autenticação da API são apresentados na Figura 28. Os modelos Token e TokenData estão relacionados à autenticação do usuário na API e sua navegação pela mesma.

5.1.4.2 Controladores

No contexto deste projeto, os controladores foram desenvolvidos como os endpoints da API, que permitem a navegação e interação dos usuários com a mesma.

Endpoint	Requisição HTTP	Parâmetros
/users/login	POST	body: username, password
/users/createUser	POST	body: username, email, role query: password header: access_token
/users/deleteUser	DELETE	body: username, email, role header: access_token
/gd/getAllStatus	GET	header: access_token
/gd/getStatus	GET	query: nome da bancada header: access_token
/gd/getTotalWorkTime	GET	query: start (data de inicio), end (data de fim), nome da bancada header: access_token
/gd/getTotalProduction	GET	query: start, end header: access_token
/gd/getTotalFails	GET	query: start, end header: access_token

Tabela 8 – Enpoints API.

Fonte: Arquivo pessoal.

Foram definidos dois níveis de usuário. Um usuário tem a função de administrador (chamado de admin). Possui as permissões para adicionar novos usuários e deletar usuários no banco de dados, além de ser o único tipo de usuário que pode enviar comandos para o SCF via API. O usuário do tipo operador pode chamar os endpoints mais simples que retornam informações dos estados de operações das bancadas, assim como informações sobre a produção e tempo de ociosidade.

A Tabela 9 apresenta os endpoints, suas descrições detalhadas e os cargos que podem realizar as chamadas dos mesmos.

Com a API é possível escalonar o projeto conforme a necessidade e demanda, sendo de fácil manutenção e utilização da mesma.

5.2 FRONT END

Nesta seção são apresentados o desenvolvimento dos principais componentes da estrutura de front-end do gêmeo digital como a virtualização das bancadas no PlantSimulation, processos de reproduzir uma simulação já realizada, a interação do sistema virtual com o SCF e a geração de gráficos. As etapas foram desenvolvidas primeiramente em modelos separados para fins de facilitar os testes de cada etapa, e por fim foram integradas.

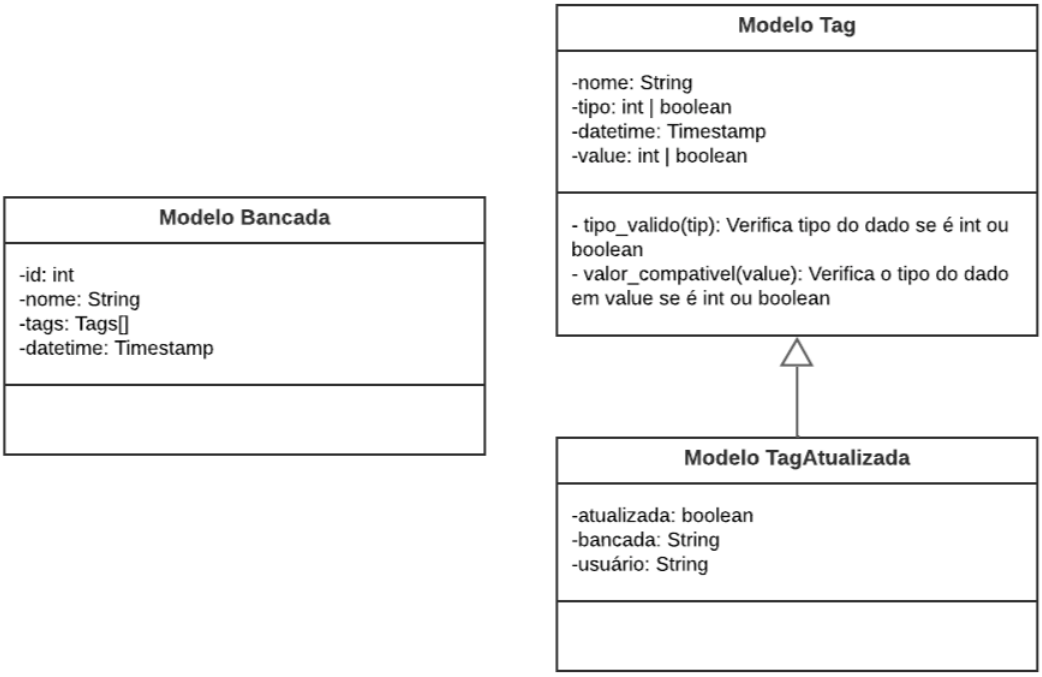


Figura 27 – Modelos de dados do GD.

Fonte: Arquivo pessoal.

5.2.1 Virtualização das bancadas

A virtualização dos modelos das bancadas foi realizado inteiramente no PlantSimulation, utilizando os recursos nativos do software para aspectos de animações. Para editar os modelos digitais dos componentes do SCF, foi utilizado o software Inventor da Autodesk e a parceria com a equipe de design da Festo que disponibilizou os arquivos .step das bancadas e de seus componentes. A figura 29 mostra o sistema final virtualizado das bancadas MPS 400 da Festo presentes no LAI.

O software *PlantSimulation* oferece uma arquitetura de sistema aberta que suporta várias interfaces e recursos de integração, como ActiveX, CAD, Oracle SQL, ODBC, XML, Socket etc. Também permite o modelamento em diferentes níveis de detalhamento de forma integrada, possibilitando que os estudos possam ser realizados em fases

Os objetos do Plant Simulation são componentes básicos que representam elementos do sistema de produção, como máquinas, equipamentos, estoques, mão-de-obra, entre outros. Os objetos podem ser criados usando bibliotecas padrões ou personalizadas, e podem ser conectados por meio de linhas ou cabos. Os objetos podem ter propriedades como massa, velocidade, posição, tempo, etc., que podem ser alteradas durante a simulação. Os objetos também podem ter eventos associados a eles, como ligar/desligar, iniciar/parar o processo, receber/despachar material e

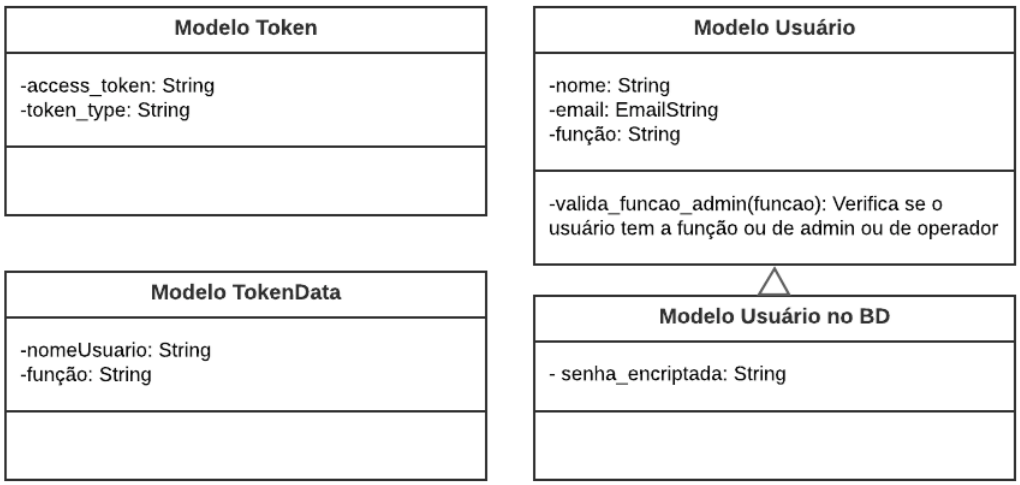


Figura 28 – Modelos de dados do usuário e autenticação.

Fonte: Arquivo pessoal.

Endpoint	Descrição	Função/Cargo
/users/login	Realiza login na API	Admin / operador
/users/createUser	Cria um novo usuário com acesso a API.	Admin
/users/deleteUser	Deleta um usuário do banco de dados na nuvem.	Admin
/gd/getAllStatus	Retorna o status de operação de todas as bancadas.	Admin / operador
/gd/getStatus	Retorna o status de operação de uma bancada específica.	Admin / operador
/gd/getTotalWorkTime	Retorna o tempo de trabalho de bancada específica em um dado intervalo.	Admin / operador
/gd/getTotalProduction	Retorna o total de peças produzidas em um dado intervalo.	Admin / operador
/gd/getTotalFails	Retorna o número de peças rejeitadas em um dado intervalo.	Admin / operador

Tabela 9 – Endpoints API.

Fonte: Arquivo Pessoal.

diversas outras funcionalidades que são executados quando o objeto é ativado.

No escopo deste projeto, no ambiente de desenvolvimento do *PlantSimulation*, foram criados os objetos de acordo com o tipo de processamento realizado pelo SCF, isto é, a bancada *Pick and Place* pode ser considerada um objeto *Assembler*, enquanto as bancadas de *Sorting* e *Distribution* foram criadas como objetos *Station*. Para a realização do transporte, tanto no SCF quanto na virtualização são utilizadas esteiras (objeto *Conveyor* no *PlantSimulation*).

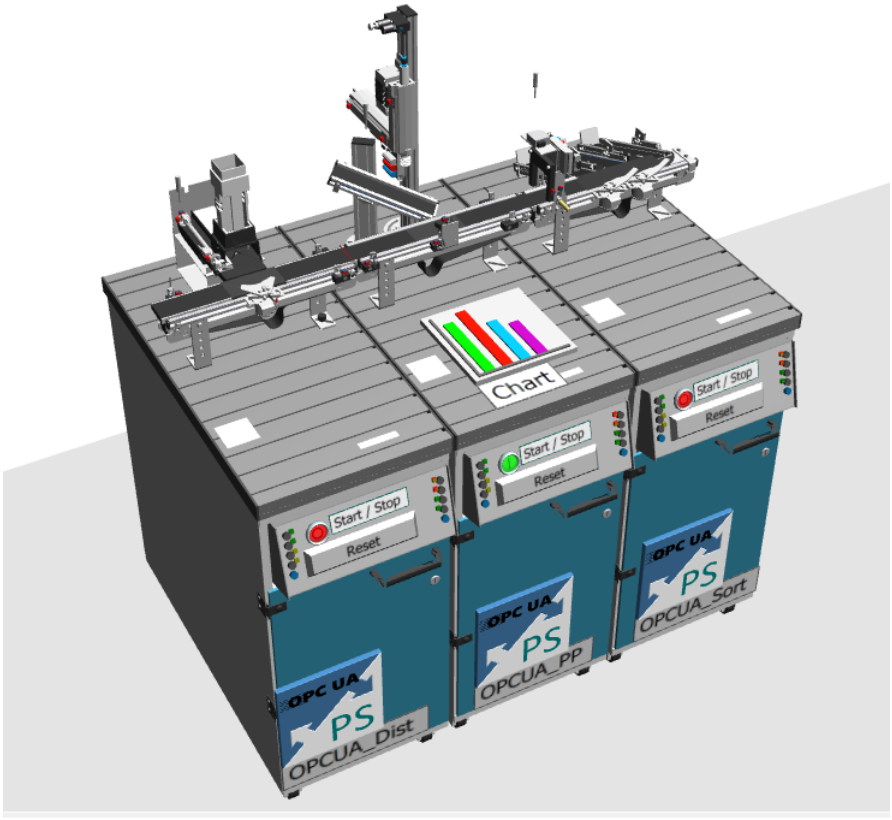


Figura 29 – SCF virtualizado no PlantSimulation.

Fonte: Arquivo pessoal.

Algumas adaptações foram necessárias para sincronizar animações de operação dos atuadores e o trabalho da bancada. De forma a operar a bancada *Sorting*, os atuadores que realizam a organização das peças são os objetos de *station*, e o conjunto todo de esteiras e dos atuadores que compõem a bancada. A bancada de *Pick and Place* possui dois atuadores principais, o atuador que segura a peça (trava) e o atuador que encaixa a tampa na peça, dessa forma, para poder simular a montagem da peça. A trava foi criada como um objeto do tipo *assembler*, e o atuador de encaixe da tampa como um objeto *station*. Por fim, a bancada *Distribution* possui apenas o atuador que trava a peça, que também foi criado como um objeto *station*.

Toda a lógica das animações foi construida utilizando os objetos de métodos, levando em consideração os valores das tags dos sensores lidas em tempo real. A Figura 30 apresenta um exemplo de código desenvolvido nos métodos para realizar as animações e seguir o fluxo do processo das bancadas. Conforme pode ser visto na Figura 30, são definidas as variáveis de entrada do método, determinadas com o marcador *param*, e criada a variável de animação *animations* com base no objeto "PickAndPlace". Por fim, dentro do loop é atribuido o valor da variável *tag*, e caso seja igual a *true*, é realizada a animação do objeto.

Além disso, para estabelecer a conexão com o SCF, foi utilizado o objeto de

```
param resultado : string, tag : boolean

var animations : any := PickAndPlace._3D.getObject("PickAndPlace").SelfAnimations

while True
    tag := OPCUA_testPP.getItemValue("B_I4")
    if tag = true
        animations.resetAnimation
        animations.Down_Z.schedule
        animations.playAnimation
        exitLoop

    else
        sleep(0.4 , false)
    end
end
```

Figura 30 – Exemplo de código desenvolvido para o método realizar animações.

Fonte: Arquivo pessoal.

comunicação/informação OPC-UA, que habilita a comunicação do GD com o SCF.

5.2.2 Interação GD com SCF

A interação do GD com o SCF é um dos requisitos (Tabela 2) mais importantes neste projeto, pois é um dos principais fatores que diferenciam um GD de uma sombra digital. Para realizar a interação entre os serviços, além da API, há também as ferramentas disponibilizadas pelo *PlantSimulation*, como os objetos botões e métodos, que juntos permitem enviar comandos para o SCF, escrevendo os valores das tags de acordo com a lógica implementada nos CLPs. É importante ressaltar nessa etapa que os comandos e tags utilizadas para envio de comandos do GD para o SCF estejam implementadas também no CLP da bancada. Caso não seja o caso, não é possível implementar a interação de forma precisa e determinística.

Conforme exposto no início desta seção, as etapas do desenvolvimento do GD foram desenvolvidas em modelos separados. Para desenvolver e testar as funcionalidades de interação do GD, foi criado um modelo contendo os objetos de conexão OPC-UA e botões, além de métodos para implementar a lógica ao clicar nos botões. A Figura 31 apresenta o modelo desenvolvido para os testes de interação do GD com SCF.

A comunicação ocorre com *delay* de 200 ms devido às limitações de comunicação das próprias bancadas, sendo 200 ms o tempo mínimo possível de atualização das *tags*. Devido à natureza do sistema das bancadas e como é realizada a operação da mesma, considerou-se que até o tempo de um segundo satisfaria a especificação de qualidade do serviço (QoS - *Quality of Service* Tabela 2) de comunicação em tempo real. Além disso, com a utilização do OPC-UA e das ferramentas internas do próprio *PlantSimulation*, foi possível estabelecer a comunicação do GD com o SCF de maneira estável e sem grandes atrasos, outro requisito também apresentado na Tabela 2.

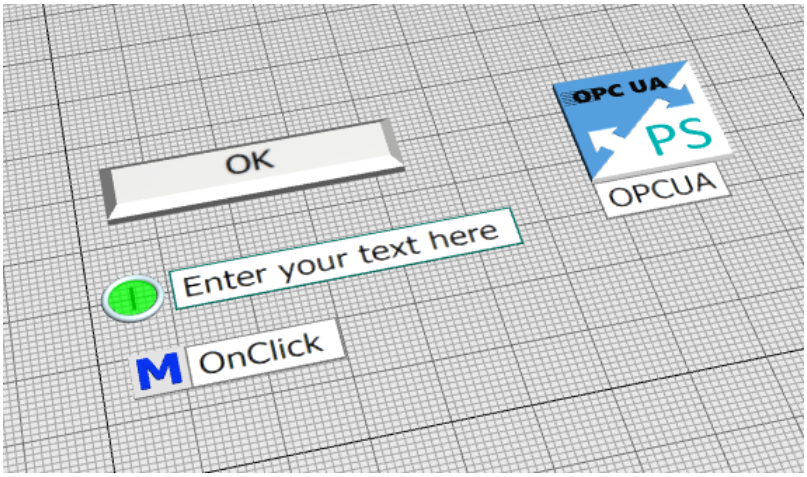


Figura 31 – Modelo desenvolvido para os testes de interação do GD com SCF.

Fonte: Arquivo pessoal.

5.2.3 Geração de Gráficos e Analytics

A geração de gráficos e relatórios é uma parte vital na gestão e controle de processos independentemente de área e setor em que se encontre. Para a implementação desse requisito do sistema, foi utilizado o objeto *Charts* do *PlantSimulation*, que é responsável por coletar e apresentar os dados de operações das bancadas em tempo real, tais como tempo de trabalho, tempo de ociosidade, entre outros indicadores. A Figura 32 apresenta um exemplo simulado de trabalho das estações. Para conseguir as informações de operação e estatísticas das bancadas, o próprio objeto *Station* possui esses atributos, que são atribuídos conforme a simulação ocorre. Assim, o objeto *Charts* apresenta essas informações do objeto *station* em tempo real. Conforme mostra na legenda da Figura 32, há nove indicadores de operação que são computados, sendo esses:

- *Working* (Verde claro): porcentagem do tempo em que a bancada (objeto *Station*) está trabalhando;
- *Seeting-Up* (Marrom): porcentagem do tempo em que a bancada está se preparando para operar;
- *Waiting* (Cinza): porcentagem do tempo em que a bancada está aguardando / esperando uma peça de trabalho;
- *Blocked* (Amarelo): porcentagem do tempo em que a bancada esta bloqueada;
- *PoweringUpDown* (Roxo): porcentagem do tempo em que a bancada esta ligada ou desligada (por ser em tempo real a simulação, esse atributo não foi utilizado neste projeto pois se a bancada está desligada, não há como realizar a comunicação entre o GD e o SCF);

- *Failed* (Vermelho): porcentagem do tempo em que ocorreram falhas na operação da bancada;
- *Paused* (Azul escuro): porcentagem do tempo em que a bancada está parada;
- *Unplanned* (Azul claro): porcentagem do tempo em que a bancada está operando fora do período estipulado (por ser em tempo real a simulação, esse atributo não foi utilizado neste projeto, já que o intuito deste atributo é para simulações com calendário de operação já programado).

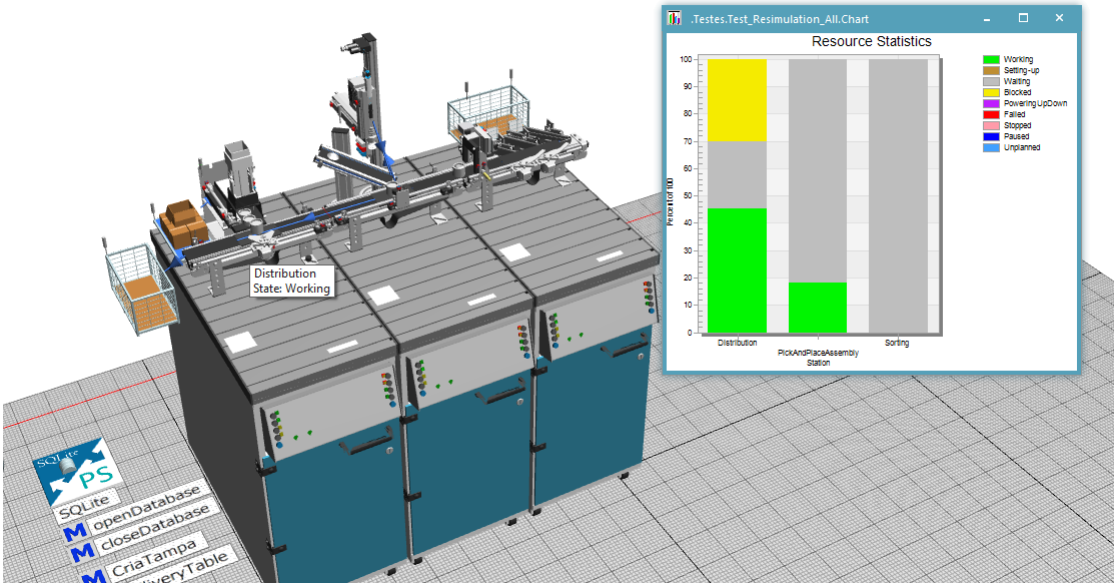


Figura 32 – Exemplo simulado de trabalho das estações.

Fonte: Arquivo pessoal.

Além disso, o *PlantSimulation* disponibiliza a geração de relatórios ao final da simulação. O anexo A apresenta um exemplo de relatório gerado de uma simulação. Nesse relatório são apresentados todos os indicadores dos objetos *station* de forma detalhada.

5.2.4 Reprisar simulação

O reprise de simulações é um dos principais requisitos deste projeto. Para a implementação desse requisito, foram desenvolvidos scripts em python para o tratamento dos dados, desde as consultas de busca utilizando intervalos de datas até a filtragem e classificação das peças.

O aspecto mais desafiador na implementação do reprise de simulação foi a parte visual, utilizando o *PlantSimulation*. Embora seja possível importar todos os dados necessários via SQLite, a utilização desses dados com os objetos de gráficos visuais não é trivial.

Cada objeto de estação desenvolvida como um atuador no *PlantSimulation* possui diversos métodos e configurações possíveis, mas, em muitas das vezes não é possível configurá-los de forma direta. Para poder contornar esse obstáculo, foi desenvolvida uma solução alternativa, que aproveita o modo de operação da simulação do software para carregar e transportar as informações de trabalho necessárias.

Essa solução consiste em utilizar as peças manufaturadas como os transportadores de informações da operação (as peças manufaturadas no software são representadas pelo objeto *Mobile Units* - MUs), como a classificação da peça por tipo de material, cor da peça e tempo de processamento em cada estação. Para isso, os scripts em python realizam a classificação das peças de acordo com os ciclos de trabalho de cada estação e salva em uma tabela no SQLite para o formato adequado a ser utilizado no objeto *Source* no *PlantSimulation*, que irá gerar as peças com todas as informações de processo.

A Figura 33 apresenta o sistema virtualizado utilizado para a realização do reprise de simulação. A construção da parte gráfica é a mesma em relação ao sistema virtualizado utilizado para simulação em tempo real, porém, os métodos e o comportamento entre eles são diferentes, por isso a necessidade de criar dois modelos gráficos para o mesmo sistema.

5.3 SISTEMA INTEGRADO

A implementação do sistema integrado foi a última etapa do projeto. Durante o desenvolvimento do GD, foram criados scripts em python que simulavam o comportamento das tags OPC-UAs das bancadas de trabalho do SCF. Dessa forma, foi possível desenvolver e implementar a maior parte deste projeto remotamente, excetuando-se a etapa de interação do GD com o SCF em tempo real, onde foi necessário a presença física para validação desta etapa. A Figura 34 apresenta o SCF das três bancadas de operação, e a Figura 35 apresenta o sistema virtualizado GD.

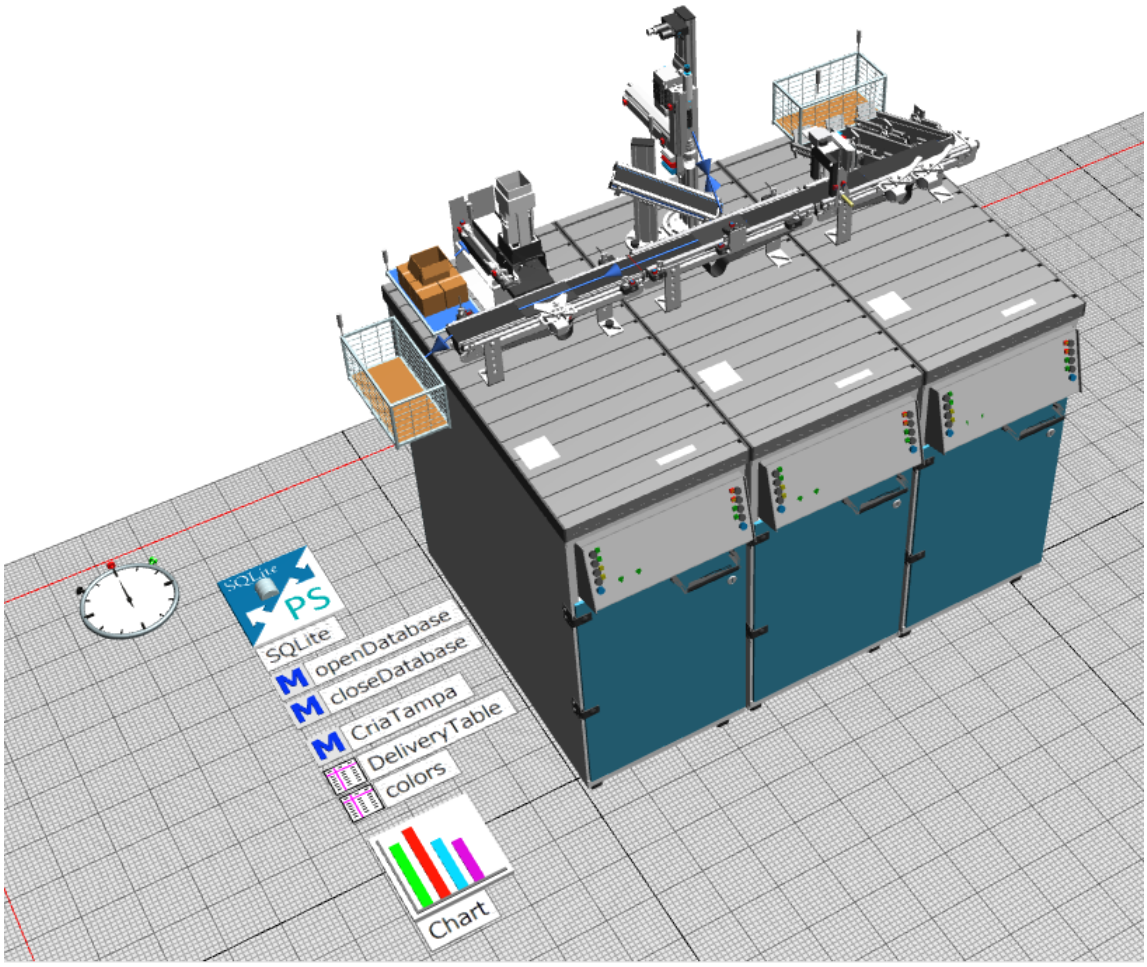


Figura 33 – Sistema virtualizado para reprise de simulação.

Fonte: Arquivo pessoal.

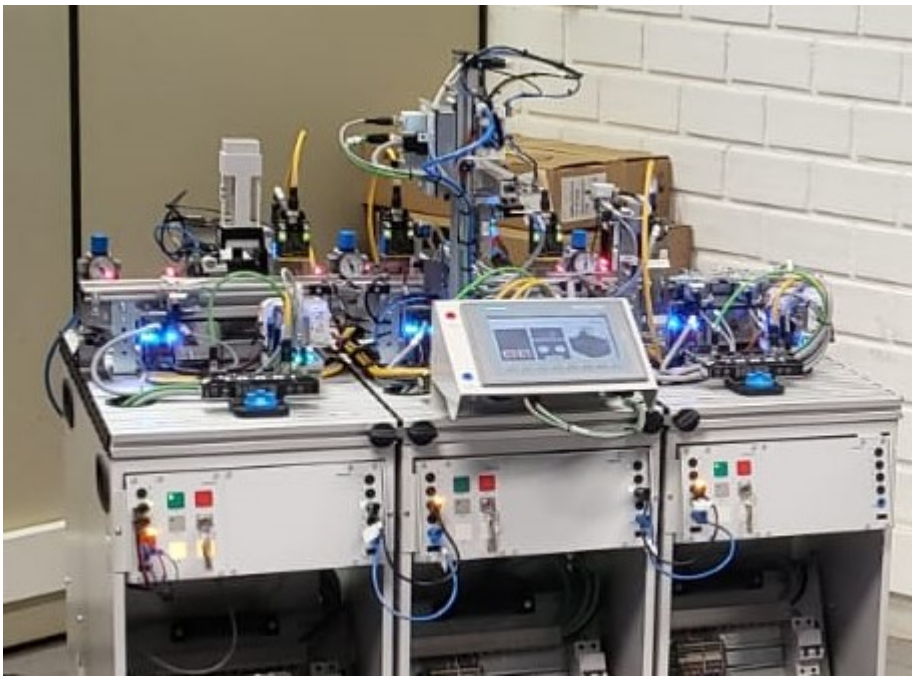


Figura 34 – Sistema Ciberfísico.

Fonte: Arquivo pessoal.

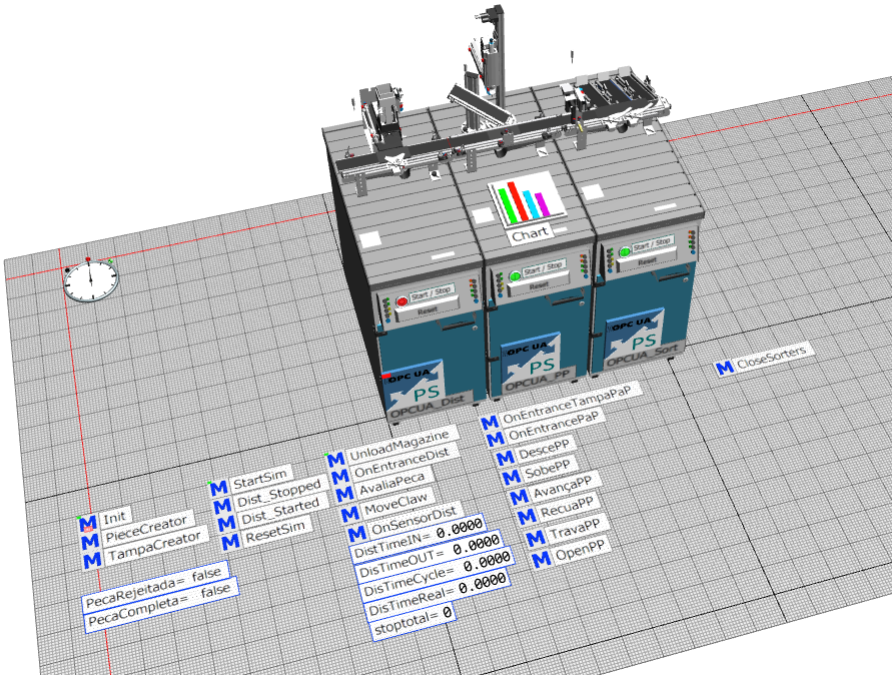


Figura 35 – Sistema Virtualizado GD.

Fonte: Arquivo pessoal.

6 CONCLUSÃO

Este projeto teve como objetivo o desenvolvimento de um Gêmeo Digital para as bancadas recém adquiridas ao LAI, seguindo uma arquitetura proposta em trabalhos anteriores. Além disso, foram utilziadas algumas das ferramentas utilizadas em trabalhos anteriores, como o *PlantSimulation* para a digitalização do SCF e o MongoDB como banco de dados local. Este trabalho adicionou o o SQLite como banco de dados local e o MongoDB como banco de dados na nuvem, permitindo a implementação do GD como um serviço que pode ser acessado por serviços externos por meio da API desenvolvida neste projeto, podendo requisitar informações em tempo real de operação do SCF. Os trabalhos anteriores foram realizados em versões mais antigas das bancadas de operação FESTO, que suportam apenas o protocolo de comunicação OPC-DA, sendo necessário uma atualização dos componentes físicos do SCF e CLPs para permitir o uso do OPC-UA.

De forma geral, conclui-se que o projeto foi implementado de acordo com as especificações e com os requisitos funcionais e não funcionais, atingindo os resultados esperados.

Como este projeto tem o intuito de implementar uma primeira versão para o GD das novas bancadas do LAI, foram utilizadas abordagens simplificadas para os requisitos iniciais deste projeto, como os aspectos de segurança na autenticação da API e do protocolo OPC-UA. O *analytics* foi construído utilizando as próprias ferramentas do *PlantSimulation* sem adicionar outros possíveis indicadores de operação. Alarmes de erros foram implementados apenas quando as bancadas estão paradas, *logs* indicando apenas mudanças de estados das *tags*, dessa forma, sendo postergado para trabalhos futuros a implementação de aspectos e requisitos mais complexos, além de possíveis melhorias e atualizações, de acordo com as necessidades desses projetos futuros. O GD implementado neste projeto, somado ao GD anterior desenvolvido, será útil para continuação de trabalhos de pesquisa em Indústria 4.0, bem como de apoio ao ensino em algumas disciplinas do curso de Eng. Controle e Automação.

Para o desenvolvimento de trabalhos futuros, deve-se buscar adicionar o Robotino ao GD, visto que a integração do mesmo neste projeto não foi desenvolvida devido a falta de familiaridade tanto do autor quanto dos responsáveis pelo LAI com o equipamento. Além disso, com intuito de manter a escalabiliade, integrabilidade e a interoperabilidade de futuros projetos que venham a utilizar este trabalho como base, recomenda-se o desacoplamento de novos componentes que serão implementados e adicionados aos novos projetos, seguindo os príncípios de desenvolvimento ágil apresentados na Seção 3.1.6.

REFERÊNCIAS

A guide to the different types of APIs. [*S.l.*: *s.n.*], 2022. https://blog.postman.com/different-types-of-apis/.

ALAM, Muhammad; SADDIK, Abdulhussain. C2PS: A digital twin architecture reference model for the cloud-based cyber-physical systems. **IEEE Access**, v. 5, p. 2050–2062, 2017.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. 2a. Rio de Janeiro: Elsevier, 2012.

BRASIL - UAB | IF SUL-RIO-GRANDENSE, Sistema Universidade Aberta do. **Middleware**. [*S.l.*: *s.n.*], 2023. Acesso em: 05 dez. 2023. Disponível em: http://uab.ifsul.edu.br/tsiad/conteudo/modulo6/_pdf/trc_ud.pdf.

CERVO, Amado Luiz; BERVIAN, Pedro Alcino; SILVA, Roberto da. **Metodologia científica**. [*S.l.*]: Pearson Prentice Hall, 2007.

CÓDIGO, Linha de. **Abordando a arquitetura MVC, e Design Patterns: Observer, Composite, Strategy**. [*S.l.*: *s.n.*], 2011. Disponível em: http://www.linhadecodigo.com.br/artigo/2367/abordando-a-arquitetura-mvc-e-design-patterns-observer-composite-strategy.aspx. Acesso em: 6 dez. 2023.

COLOMBO, Armando Walter; KARNOUSKOS, Stamatis; KAYNAK, Okyay. Industrial Cyberphysical Systems: A Backbone of the Fourth Industrial Revolution. **IEEE Industrial Electronics Magazine**, v. 11, p. 6–16, 2017.

DEVMEDIA. **Introdução ao padrão MVC**. [*S.l.*: *s.n.*], 2020. Disponível em: https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308. Acesso em: 6 dez. 2023.

DISTRIBUTION. [*S.l.*: *s.n.*], 2023. https://www.festo.com/gb/en/p/distributing-station-id_PROD_DID_8034566/.

DISTRIBUTION Station. [*S.l.*: *s.n.*], 2023. https://ip.festo-didactic.com/InfoPortal/MPS/DistributionConveyorStation/EN/index.html.

GÊMEOS Digitais: a união entre o físico e o virtual no combater a crimes cibernéticos. [*S.l.*: *s.n.*], 2022. https://www.industria40.ind.br/artigo/22142-gemeos-digitais-uniao-entre-fisico-virtual-combate-crimes-ciberneticos.

GÊMEOS Digitais: conheça o Futuro da Indústria. [*S.l.*: *s.n.*], 2021. https://consumidormoderno.com.br/2021/03/25/digital-twins-tecnologia-disruptiva-ajudar-mercado/.

GÊMEOS Digitais: conheça o Futuro da Indústria. [*S.l.*: *s.n.*], 2022. https://www.birmind.com.br/blog/gemeos-digitais-conheca-o-futuro-da-industria/.

GHOBAKHLOO, Morteza. The future of manufacturing industry: a strategic roadmap toward Industry 4.0. **Journal of Manufacturing Technology Management**, Emerald Publishing Limited, v. 29, n. 6, p. 910–936, 2018.

GIL. **Como elaborar projetos de pesquisa**. 5ª ed. São Paulo, Brazil: Editora Atlas, 2017.

GRÄSSLE, Michael; SCHLEIPEN, Miriam; ZOITL, Alois; STRASSER, Thomas. OPC UA: The next generation of OPC. **at - Automatisierungstechnik**, v. 64, n. 2, p. 81–88, 2016. DOI: 10.1515/auto-2015-0077.

HERMANN, Mario; PENTEK, Tobias; OTTO, Boris. Design Principles for Industrie 4.0 Scenarios: A Literature Review, jan. 2015. DOI: 10.13140/RG.2.2.29269.22248.

HOW to create custom OPC UA Information Models. [*S.l.*: *s.n.*], 2023. https://profanter.medium.com/how-to-create-custom-opc-ua-information-models-1e9a461f5b58.

ILUSTRADEV. **O que é MVC? Explicação Simples**. [*S.l.*: *s.n.*], 2019. Disponível em: https://ilustradev.com.br/o-que-e-mvc-explicacao-simples/. Acesso em: 6 dez. 2023.

INDUSTRY 4.0: The Future of Manufacturing. [*S.l.*: *s.n.*], 2019. https://www.bbc.com/news/business-49418160.

KAGERMANN, Henning; WAHLSTER, Wolfgang; HELBIG, Johannes. **Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Final report of the Industrie 4.0 working group**. [*S.l.*], 2013a. P. 1–44. Disponível em:

https://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Material_fuer_Sonderseiten/Industrie_4.0/Final_report_Industrie_4.0_accessible.pdf.

KAGERMANN, Henning; WAHLSTER, Wolfgang; HELBIG, Johannes. **Recommendations for implementing the strategic initiative industrie 4.0: Final report of the Industrie 4.0 working group**. [*S.l.*: *s.n.*], 2013b. Disponível em: https://www.din.de/blob/76902/e8cac883f42bf28536e7e8165993f1fd/recommendations-forimplementing-industry-4-0-data.pdf.

LEE, Jay; BAGHERI, B; KAO, Hung-An. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. **Manufacturing Letters**, 2015.

LIAO, Ying; DESCHAMPS, Frédéric; LOURES, Eduardo de Freitas Rocha; RAMOS, Luiz Fernando Pereira. Past, Present and Future of Industry 4.0 - A Systematic Literature Review and Research Agenda Proposal. **International Journal of Production Research**, v. 55, n. 12, p. 3609–3629, 2017.

LIAO, Yongxin; DESCHAMPS, Fernando; LOURES, Eduardo de Freitas Rocha; RAMOS, Luiz Fernando Pereira. Past, present and future of industry 4.0 - a systematic literature review and research agenda proposal. **International Journal of Productions Research**, v. 55, n. 12, p. 3609–3629, 2017. DOI: 10.1080/00207543.2017.1308576.

LU, Y. Industry 4.0: A survey on technologies, applications and open research issues. **Journal of Industrial Information Integration**, Elsevier, v. 6, p. 1–10, 2017.

MAHNKE, Wolfgang; LEITNER, Stefan-Helmut; DAMM, Matthias. **OPC Unified Architecture: The Everyman's Guide to OPC UA**. [*S.l.*]: CreateSpace Independent Publishing Platform, 2016. ISBN 978-1530901886.

MARIO NEMIROVSKY, Dean M. Tullsen. **Multithreading Architecture**. [*S.l.*]: Springer, 2013.

MARTIN, Robert C. **Agile software development, Principles, Patterns and Practices**. [*S.l.*]: Pearson Education, 2003.

MOEUF, Alexandre; PELLERIN, Robert; LAMOURI, Samir; TAMAYO-GIRALDO, Santiago; BARBARAY, Robin. The industrial management of

SMEs in the era of industry 4.0. **International Journal of Productions Research**, v. 56, n. 3, p. 1118–1136, 2018. DOI: 10.1080/00207543.2017.1372647.

MONGODB. **MongoDB Atlas Tutorial**. [*S.l.*: *s.n.*], 2023. Acesso em: 05 dez. 2023. Disponível em: https://www.mongodb.com/docs/atlas/.

O que é API? Entenda como funciona e como utilizar na sua empresa. [*S.l.*: *s.n.*], 2023. https://lyncas.net/api-o-que-e-e-beneficios/.

O que é um gêmeo digital? Uma representação virtual em tempo real. [*S.l.*: *s.n.*]. https://itforum.com.br/noticias/o-que-e-um-gemeo-digital-uma-representacao-virtual-em-tempo-real/.

OPC Foundation. [*S.l.*: *s.n.*]. Disponível em: https://opcfoundation.org/.

OPC UA Animal Server. [*S.l.*: *s.n.*], 2023. https://github.com/Pro/opcua-animal-server#opc-ua-animal-server.

OPC UA Modeling Tutorial. [*S.l.*: *s.n.*], 2023. https://github.com/Pro/opcua-modeling-tutorial.

OPC UA Super Complete Tutorial. [*S.l.*: *s.n.*], 2023. https://joseamaita.com/en/opc-ua-super-complete-tutorial/.

OPC Unified Architecture UA Modelling Best Practices. [*S.l.*: *s.n.*], 2023. https://opcfoundation.org/wp-content/uploads/2020/09/OPC-11030-Whitepaper-UA-Modeling-Best-Practices-1.00.00.pdf.

PETRONI, Benedito; GONÇALVES, Rodrigo; SATYRO, Walter. Sistemas Cyber Físicos. *In*: [*S.l.*: *s.n.*], nov. 2018. P. 39. ISBN 9788521213703.

PICK and Place. [*S.l.*: *s.n.*], 2023. https://www.festo.com/br/pt/p/estacao-pick-place-id_PROD_DID_8034567/.

PICK and Place Station. [*S.l.*: *s.n.*], 2023. https://ip.festo-didactic.com/InfoPortal/MPS/PickandPlaceStation/EN/index.html.

PLANT Simulation Trial. [*S.l.*: *s.n.*], 2023. https://www.plm.automation.siemens.com/store/pt-br/trial/plant-simulation.html.

QUASAR framework - OPC-UA server generation. [*S.l.*: *s.n.*], 2023. https://kt.cern/technologies/quasar-framework-opc-ua-server-generation.

QUINALHA, EDUARDO. **GÊMEOS DIGITAIS, O FUTURO DA INDÚSTRIA 4.0: ESTUDO DE CASO**. 2018. Monografia de Especialização (Curso de Especialização em Redes de Computadores e Teleinformática) – UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ.

RABELO, Ricardo J; MAGALHÃES, Luciano C; CABRAL, Felipe G. Uma Proposta de Arquitetura de Referência de Gêmeo Digital para Sistemas Ciberfísicos em um cenário de Indústria 4.0. **Congresso Brasileiro de Automática (CBA)**, 2020.

RABELO, Ricardo José. **Indústria 4.0 – História, definições e visões**: Na esteira da Quarta Revolução Industrial. O que faz da chamada Indústria 4.0 uma revolução? [*S.l.*: *s.n.*], 2021. Disponível em: https://pgeas.ufsc.br/revista-ppgeas/revista-ppgeas-numero-2-industria-4-0/. Acesso em: 28 nov. 2023.

REINER, Anderl. Industrie 4.0 - Advanced Engineering of Smart Products and Smart Production. **International Seminar on High Technology**, October, p. 1–14, 2014. Disponível em: %5E1%5E.

REST API Tutorial. [*S.l.*: *s.n.*], 2023. https://restfulapi.net/.

ROBLEK, Vasja; MEŠKO, Maja; KRAPEŽ, Anton. A Complex View of Industry 4.0. **SAGE Open**, v. 6, n. 2, 2016. DOI: 10.1177/2158244016653987.

ROBOTINO. [*S.l.*: *s.n.*], 2023. https://ip.festo-didactic.com/InfoPortal/Robotino/Overview/EN/index.html.

SCHRIEGEL, Stefan; JASPERNEITE, Jürgen. OPC UA and TSN: A new level of interoperability for industrial communication. **at - Automatisierungstechnik**, v. 66, n. 11, p. 907–916, 2018. DOI: 10.1515/auto-2018-0069.

SCHWAB, Klaus. **The Fourth Industrial Revolution**. [*S.l.*]: Currency, 2017.

SILLER, H.; ROMERO, D.; RABELO, R. J. Advanced CPS Service Oriented Architecture for Smart Injection Molding and Molds 4.0. *In*: ANAIS IEEE International Conference on Intelligent Systems. [*S.l.*: *s.n.*], 2018. P. 1–10.

SILVA, Jodas Ademir da. **Uso do padrão MVC em métodos ágeis**. [*S.l.*]: Novas Edições Acadêmicas, 2015.

SORTING Station. [*S.l.*: *s.n.*], 2023. https://ip.festo-didactic.com/InfoPortal/MPS/SortingStation/EN/index.html.

SOUZA, Rafael de. **Uso do padrão MVC em métodos ágeis: Para desenvolvimento de sistemas web**. [*S.l.*]: Novas Edições Acadêmicas, 2016. ISBN 978-3-639-74891-5. Acesso em: 6 dez. 2023.

THE Role of Artificial Intelligence in Industry 4.0. [*S.l.*: *s.n.*], 2020. https://www.forbes.com/sites/cognitiveworld/2020/05/07/the-role-of-artificial-intelligence-in-industry-40/?sh=34a522076129.

TRIPP, David. Pesquisa-ação: uma introdução metodológica. **Educação e Pesquisa**, Faculdade de Educação da Universidade de São Paulo, v. 31, n. 3, p. 443–466, set. 2005. ISSN 1517-9702. DOI: 10.1590/S1517-97022005000300009. Disponível em: https://doi.org/10.1590/S1517-97022005000300009.

UML: Casos de Uso. [*S.l.*: *s.n.*], 2022. Disponível em: https://moodle.unesp.br/pluginfile.php/25934/mod_resource/content/1/diagrama_casos_uso.pdf.

UTILIZAÇÃO dos gêmeos digitais no planejamento de processos. [*S.l.*: *s.n.*]. https://adelpha-api.mackenzie.br/server/api/core/bitstreams/f4f184b2-3637-45cf-b63b-3e03da93192a/content.

VALENTE, Marco Tulio. **Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade**. [*S.l.*]: Independente), 2022. v. 1.

VERCEL. [*S.l.*: *s.n.*]. https://vercel.com/.

WAGON, Le. **O que é padrão MVC? Entenda arquitetura de softwares!** [*S.l.*: *s.n.*], 2020. Disponível em: https://blog.lewagon.com/pt-br/skills/o-que-e-padrao-mvc/. Acesso em: 6 dez. 2023.

WHAT is an API? [*S.l.*: *s.n.*], 2023. https://www.ibm.com/topics/api.

APÊNDICE A – OPC-UA

A.1 CONFIGURAR MODELO DE PROTOCOLO OPC-UA

Para criar e configurar um modelo de protocolo OPC-UA do zero, é preciso seguir alguns passos básicos:

- Definir o modelo de informação, que é a estrutura lógica dos dados e serviços que serão expostos pelo servidor OPC-UA. O modelo de informação é composto por nós, que são as entidades que contêm os dados ou oferecem os serviços, e referências, que são as relações entre os nós. O modelo de informação pode ser definido usando uma linguagem específica chamada OPC UA NodeSet XML Schema ou usando ferramentas gráficas como o UA Modeler;
- Gerar o código-fonte do servidor OPC-UA, que é o programa que implementa o modelo de informação e a comunicação com os clientes OPC-UA. O código-fonte pode ser gerado usando ferramentas como o UA SDK ou o open62541, que oferecem bibliotecas e exemplos para diferentes linguagens de programação, como C, C++, Java, Python, etc. O código-fonte deve ser compilado e executado em uma plataforma compatível com OPC-UA, como Windows, Linux, etc;
- Testar e validar o servidor OPC-UA, que é o processo de verificar se o servidor OPC-UA está funcionando corretamente e se está em conformidade com as especificações e normas do OPC-UA. O teste e a validação podem ser feitos usando ferramentas como o UA Expert ou o UaTester, que permitem conectar-se ao servidor OPC-UA, navegar pelo seu espaço de endereçamento, ler e escrever dados, invocar métodos, assinar eventos, etc.

O código 1 apresenta como configurar um modelo de protocolo OPC-UA em um nível mais alto, dependendo de bibliotecas auxiliares responsáveis pelas camadas mais profundas descritas mais a frente neste apêndice. No código 1 pode ser visto fatores importantes para o protocolo OPC-UA, tais como o tipo de dados, que é a forma como os dados são representados e codificados no OPC-UA, o método, que é a função que pode ser chamada pelo cliente OPC-UA para executar uma ação no servidor OPC-UA e o endpoint, que é o ponto de acesso que permite a comunicação entre o cliente e o servidor OPC-UA.

O tipo de dados pode ser um tipo primitivo, como booleano, inteiro ou string, ou um tipo complexo, como uma estrutura, uma matriz ou uma união. O tipo de dados também pode ser definido por meio de um esquema XML ou JSON, que especifica a estrutura e as restrições dos dados. Já método é definido por um nome, uma lista de parâmetros de entrada e saída e um código de retorno. O método também pode ter propriedades associadas, como descrição, visibilidade e segurança.

Por fim, o endpoint é definido por uma URL, um perfil de segurança e um conjunto de serviços suportados. O perfil de segurança especifica os mecanismos de autenticação, autorização, criptografia e integridade dos dados. O conjunto de serviços especifica as operações que podem ser realizadas pelo cliente no servidor.

Código 1

```
Importar as bibliotecas necessarias
from opcua import Server, ua

# Criar uma instancia do servidor OPC-UA
server = Server()

# Definir o namespace do modelo de informacao
uri = "http:example.com/opcua"
idx = server.register_namespace(uri)

# Obter uma referencia para o no raiz do modelo de informacao
root = server.get_root_node()

# Criar um no do tipo objeto para representar o sistema
system = server.nodes.objects.add_object(idx, "System")

# Criar um no do tipo variavel para
# representar a temperatura do sistema
temperature = system.add_variable(idx, "Temperature", 0.0)

# Definir o tipo de dados da variavel como float
temperature.set_data_type(ua.VariantType.Float)

# Definir a propriedade writable da variavel como True
temperature.set_writable(True)

# Criar um no do tipo metodo para
# representar a funcao de ligar o sistema
def turn_on(parent):
    print("System turned on")
    return ua.StatusCode.Good

turn_on_method = system.add_method(idx, "TurnOn", turn_on)
```

```
# Definir a descricao do metodo
turn_on_method.set_attribute(
    ua.AttributeIds.Description,
    ua.LocalizedText("Turn on the system")
    )

# Definir o endpoint do servidor OPC-UA
url = "opc.tcp:localhost:4840"
server.set_endpoint(url)

# Definir o perfil de seguranca do servidor OPC-UA
server.set_security_policy([
    ua.SecurityPolicyType.NoSecurity,
    ua.SecurityPolicyType.Basic256Sha256_SignAndEncrypt,
    ua.SecurityPolicyType.Basic256Sha256_Sign])

# Iniciar o servidor OPC-UA
server.start()

# Aguardar uma interrupcao do usuario para encerrar
# o servidor OPC-UA
try:
    input("Press Enter to exit\n")
finally:
    server.stop()
```

Fontes utilizadas e auxiliares: (OPC. . . , 2023), (OPC. . . , 2023), (OPC. . . , 2023).

A.2 MODELO DE INFORMAÇÃO

O modelo de informação do OPC-UA é a estrutura lógica dos dados e serviços que são expostos pelo servidor OPC-UA para o cliente OPC-UA. O modelo de informação é composto por nós, que são as entidades que representam os dados, e referências, que são as relações entre os nós. O modelo de informação pode ser baseado em modelos padrão definidos pela OPC Foundation ou por outras organizações, ou pode ser personalizado de acordo com as necessidades da aplicação.

Para definir o modelo de informação do OPC-UA, é preciso seguir alguns passos básicos:

- Escolher um namespace, que é o espaço de nomes que identifica os nós do

modelo de informação. O namespace pode ser um número (namespace index) ou uma URI (namespace URI). O namespace deve ser único e consistente para evitar conflitos ou ambiguidades entre os nós. O namespace também deve ser registrado na OPC Foundation para garantir a interoperabilidade;

- Escolher um tipo de nó, que é a classe que define as características e o comportamento dos nós. Os tipos de nó mais comuns são: objeto, variável, método, tipo de objeto, tipo de variável e tipo de método. Cada tipo de nó possui atributos obrigatórios e opcionais, como nome, descrição, valor, tipo de dados, etc;
- Escolher um tipo de referência, que é a classe que define o tipo e a direção da relação entre os nós. Os tipos de referência mais comuns são: hierárquica, não hierárquica, agregação e composição. Cada tipo de referência possui propriedades obrigatórias e opcionais, como nome, descrição, inverso, simétrico, etc;
- Criar os nós e as referências usando uma ferramenta de modelagem ou uma linguagem de programação. As ferramentas de modelagem permitem criar o modelo de informação de forma gráfica e gerar um arquivo XML ou JSON que representa o modelo. As linguagens de programação permitem criar o modelo de informação de forma dinâmica e integrada ao código do servidor OPC-UA.

Um exemplo de como fazer um modelo de informação do OPC-UA:

- Escolher um namespace tal que o namespace pode ser uma URI como:

  http://example.com/opcua/animal;
- Escolher um tipo de nó tal que o tipo de nó pode ser um tipo de objeto que representa uma entidade complexa com propriedades e métodos. Neste exemplo o tipo de objeto é "AnimalType", que define os atributos comuns a todos os animais, como nome, idade e peso;
- Escolher um tipo de referência tal que o tipo de referência pode ser uma referência hierárquica que representa uma relação de subordinação ou composição. Neste exemplo a referência hierárquica é "HasComponent", que indica que um nó é um componente de outro nó.
- Criar os nós e as referências usando uma ferramenta de modelagem ou uma linguagem de programação. Por exemplo, a ferramenta UaModeler permite criar o modelo de informação de forma gráfica e gerar um arquivo XML que representa o modelo.

O código 2 apresenta o exemplo de código em XML para criar um modelo de informação do OPC-UA para animais.

Código 2

```
<?xml version="1.0" encoding="utf-8"?>
<UANodeSet xmlns="http://opcfoundation.org/UA/2011/03/UANodeSet.xsd">
  <NamespaceUris>
    <Uri>http://example.com/opcua/animal</Uri>
  </NamespaceUris>
  <Models>
    <Model ModelUri="http://example.com/opcua/animal"
        Version="1.0"
        PublicationDate="2023-04-15T00:00:00Z">
      <RequiredModel ModelUri="http://opcfoundation.org/UA/"
        Version="1.04"
        PublicationDate="2018-11-14T00:00:00Z"/>
    </Model>
  </Models>
  <UAObjectType NodeId="ns=1;i=1001" BrowseName="1:AnimalType">
    <DisplayName>AnimalType</DisplayName>
    <References>
      <Reference ReferenceType="HasSubtype" IsForward="false">i=58</Reference>
    </References>
    <Definition Name="AnimalType">
      <Field Name="Name" DataType="String"/>
      <Field Name="Age" DataType="UInt16"/>
      <Field Name="Weight" DataType="Double"/>
    </Definition>
  </UAObjectType>
  <UAObject NodeId="ns=1;i=2001" BrowseName="1:Cat" TypeDefinition="i=1001">
    <DisplayName>Cat</DisplayName>
    <References>
      <Reference ReferenceType="Organizes" IsForward="false">i=85</Reference>
      <Reference ReferenceType="HasComponent">ns=1;i=3001</Reference>
    </References>
  </UAObject>
  <UAVariable NodeId="ns=1;i=3001"
     BrowseName="1:Name"
     DataType="String"
     ValueRank="-1"
     ArrayDimensions=""
     AccessLevel="3"
     UserAccessLevel="3">
```

```
    <DisplayName>Name</DisplayName>
    <References>
      <Reference ReferenceType="HasComponent" IsForward="false">
        ns=1;i=2001
        </Reference>
      <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
    </References>
    <Value>
      <uax:String>Cattie</uax:String>
    </Value>
  </UAVariable>
</UANodeSet>
```

Esse código cria um tipo de objeto chamado "AnimalType", que possui três campos: nome, idade e peso. Em seguida, cria um objeto do tipo "AnimalType", chamado "Cat", que representa um gato. Por fim, cria uma variável do tipo "String", chamada "Name", que representa o nome do gato e tem o valor "Cattie". O código também define as referências entre os nós, usando os tipos padrão da OPC Foundation.

Fontes: (OPC. . . , 2023), (HOW. . . , 2023) e (OPC. . . , 2023).

A.3 GERAR CÓDIGO-FONTE

Para gerar o código-fonte do servidor OPC-UA, é preciso ter um modelo de informação do OPC-UA definido, que representa a estrutura lógica dos dados e serviços que serão expostos pelo servidor. O modelo de informação pode ser baseado em modelos padrão da OPC Foundation ou personalizado de acordo com as necessidades da aplicação. O modelo de informação pode ser criado usando uma ferramenta de modelagem ou uma linguagem de programação.

Existem diversas ferramentas e bibliotecas que permitem gerar o código-fonte do servidor OPC-UA a partir do modelo de informação, usando diferentes linguagens de programação e plataformas, sendo algumas delas:

- Quasar framework, que permite gerar um servidor OPC-UA em C++ usando um arquivo XML que descreve o modelo de informação. O framework também gera um esqueleto de código para adicionar a lógica específica de integração com o sistema ou dispositivo alvo;
- UaModeler, que permite gerar um servidor OPC-UA em C# usando uma interface gráfica que facilita a criação e a edição do modelo de informação. O UaModeler usa o SDK .NET da OPC Foundation para gerar o código-fonte do servidor;

- Node-RED, que permite gerar um servidor OPC-UA em JavaScript usando um ambiente de programação visual baseado em fluxos. O Node-RED usa o pacote node-red-contrib-opcua para gerar o código-fonte do servidor;

Utilizando o Quasar framework é possível gerar o código-fonte do servidor OPC-UA com os seguintes passos:

- Criar um arquivo XML que descreve o modelo de informação do OPC-UA, seguindo as regras e os exemplos do framework. Por exemplo, o código 2;
- Executar o comando ‘designToGenerated.sh‘ para gerar o código-fonte do servidor OPC-UA em C++ a partir do arquivo XML. O comando cria uma pasta chamada "generated"com os arquivos necessários para compilar e executar o servidor;
- Executar o comando ‘make‘ para compilar o código-fonte do servidor OPC-UA. O comando cria um executável chamado "server"na pasta "bin";
- Executar o comando ‘./server‘ para iniciar o servidor OPC-UA. O comando inicia o servidor na porta 4840 e expõe o modelo de informação definido no arquivo XML;
- Conectar um cliente OPC-UA ao servidor usando a URL ‘opc.tcp://localhost:4840‘ e verificar se os dados e serviços estão disponíveis conforme o modelo de informação.

Fontes de (QUASAR. . . , 2023) e (OPC. . . , 2023).

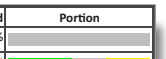
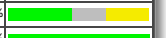
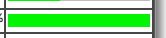
ANEXO A – RELATÓRIO

Exemplo de relátorio gerado pelo software *PlantSimulation*.

Tecnomatix Plant Simulation 16 Statistics

Created on	sábado, 6 de janeiro de 2024 11:53
Model name	C:\Users\pedro\Downloads\PFC\PFC_bckp_Holder27.spp
Simulation time	16.3943

Resource Statistics - Resource Statistics

Object	Working	Set-up	Waiting	Blocked	Powering up/down	Failed	Stopped	Paused	Unplanned	Portion
Source	0.00%	0.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
Distribution	45.41%	0.00%	24.40%	30.19%	0.00%	0.00%	0.00%	0.00%	0.00%	
ConvCreator	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
RejectedPiece	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
TestPiece	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
PickAndPlaceAssembly	18.30%	0.00%	81.70%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
PickAndPlace	36.60%	0.00%	63.40%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
PassedPiece	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
Sorting	0.00%	0.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
Sorted1	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
Sorted2	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
Sorted3	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
ConvTampa	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
Drain	0.00%	0.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
Drain1	0.00%	0.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	

Portions of the States

Object	Number of Entries	Number of Exits	Minimum Contents	Maximum Contents	Relative Empty	Relative Full	Relative Occupation without Interruptions	Relative Occupation with Interruptions
Source	6	6	0	1	100.00%	-	0.00%	0.00%
Distribution	3	2	0	1	24.40%	-	75.60%	75.60%
ConvCreator	6	3	0	3	12.20%	-	23.09%	23.09%
RejectedPiece	0	0	0	0	100.00%	-	0.00%	0.00%
TestPiece	2	2	0	1	84.75%	-	1.61%	1.61%
PickAndPlaceAssembly	2	1	0	1	33.55%	-	66.45%	66.45%
PickAndPlace	1	1	0	1	63.40%	-	36.60%	36.60%
PassedPiece	1	0	0	1	91.19%	-	0.95%	0.95%
Sorting	0	0	0	0	100.00%	-	0.00%	0.00%
Sorted1	0	0	0	0	100.00%	-	0.00%	0.00%
Sorted2	0	0	0	0	100.00%	-	0.00%	0.00%
Sorted3	0	0	0	0	100.00%	-	0.00%	0.00%
ConvTampa	2	1	0	1	88.45%	-	1.73%	1.73%
Drain	0	0	0	0	100.00%	-	0.00%	0.00%
Drain1	0	0	0	0	100.00%	-	0.00%	0.00%

Material Flow Properties

Object	Portion	Count	Sum	Mean Value	Standard Deviation
Source	0.00%	0	0.0000	0.0000	0.0000
Distribution	45.41%	3	7.4443	2.4814	2.1924
ConvCreator	100.00%	1	16.3943	16.3943	0.0000
RejectedPiece	100.00%	1	16.3943	16.3943	0.0000
TestPiece	100.00%	1	16.3943	16.3943	0.0000
PickAndPlaceAssembly	18.30%	1	3.0000	3.0000	0.0000
PickAndPlace	36.60%	1	6.0000	6.0000	0.0000
PassedPiece	100.00%	1	16.3943	16.3943	0.0000
Sorting	0.00%	0	0.0000	0.0000	0.0000
Sorted1	100.00%	1	16.3943	16.3943	0.0000
Sorted2	100.00%	1	16.3943	16.3943	0.0000
Sorted3	100.00%	1	16.3943	16.3943	0.0000
ConvTampa	100.00%	1	16.3943	16.3943	0.0000
Drain	0.00%	0	0.0000	0.0000	0.0000
Drain1	0.00%	0	0.0000	0.0000	0.0000

Working Time

Object	Portion	Count	Sum	Mean Value	Standard Deviation
Source	0.00%	0	0.0000	0.0000	0.0000
Distribution	0.00%	0	0.0000	0.0000	0.0000
ConvCreator	0.00%	0	0.0000	0.0000	0.0000
RejectedPiece	0.00%	0	0.0000	0.0000	0.0000
TestPiece	0.00%	0	0.0000	0.0000	0.0000
PickAndPlaceAssembly	0.00%	0	0.0000	0.0000	0.0000
PickAndPlace	0.00%	0	0.0000	0.0000	0.0000
PassedPiece	0.00%	0	0.0000	0.0000	0.0000
Sorting	0.00%	0	0.0000	0.0000	0.0000
Sorted1	0.00%	0	0.0000	0.0000	0.0000
Sorted2	0.00%	0	0.0000	0.0000	0.0000
Sorted3	0.00%	0	0.0000	0.0000	0.0000
ConvTampa	0.00%	0	0.0000	0.0000	0.0000
Drain	0.00%	0	0.0000	0.0000	0.0000
Drain1	0.00%	0	0.0000	0.0000	0.0000

Set-up Time

Object	Portion	Count	Sum	Mean Value	Standard Deviation
Source	100.00%	6	16.3943	2.7324	0.6555
Distribution	24.40%	2	4.0000	2.0000	0.0000
ConvCreator	0.00%	0	0.0000	0.0000	0.0000
RejectedPiece	0.00%	0	0.0000	0.0000	0.0000
TestPiece	0.00%	0	0.0000	0.0000	0.0000
PickAndPlaceAssembly	81.70%	2	13.3943	6.6972	7.4286
PickAndPlace	63.40%	2	10.3943	5.1972	1.0647
PassedPiece	0.00%	0	0.0000	0.0000	0.0000
Sorting	100.00%	1	16.3943	16.3943	0.0000
Sorted1	0.00%	0	0.0000	0.0000	0.0000
Sorted2	0.00%	0	0.0000	0.0000	0.0000
Sorted3	0.00%	0	0.0000	0.0000	0.0000
ConvTampa	0.00%	0	0.0000	0.0000	0.0000
Drain	100.00%	1	16.3943	16.3943	0.0000
Drain1	100.00%	1	16.3943	16.3943	0.0000

Waiting Time

Object	Portion	Count	Sum	Mean Value	Standard Deviation
Source	0.00%	0	0.0000	0.0000	0.0000
Distribution	30.19%	1	4.9500	4.9500	0.0000
ConvCreator	0.00%	0	0.0000	0.0000	0.0000
RejectedPiece	0.00%	0	0.0000	0.0000	0.0000
TestPiece	0.00%	0	0.0000	0.0000	0.0000
PickAndPlaceAssembly	0.00%	0	0.0000	0.0000	0.0000
PickAndPlace	0.00%	0	0.0000	0.0000	0.0000
PassedPiece	0.00%	0	0.0000	0.0000	0.0000
Sorting	0.00%	0	0.0000	0.0000	0.0000
Sorted1	0.00%	0	0.0000	0.0000	0.0000
Sorted2	0.00%	0	0.0000	0.0000	0.0000
Sorted3	0.00%	0	0.0000	0.0000	0.0000
ConvTampa	0.00%	0	0.0000	0.0000	0.0000
Drain	0.00%	0	0.0000	0.0000	0.0000
Drain1	0.00%	0	0.0000	0.0000	0.0000

Blocked Time

Object	Portion	Count	Sum	Mean Value	Standard Deviation
Source	0.00%	0	0.0000	0.0000	0.0000
Distribution	0.00%	0	0.0000	0.0000	0.0000
ConvCreator	0.00%	0	0.0000	0.0000	0.0000
RejectedPiece	0.00%	0	0.0000	0.0000	0.0000
TestPiece	0.00%	0	0.0000	0.0000	0.0000
PickAndPlaceAssembly	0.00%	0	0.0000	0.0000	0.0000
PickAndPlace	0.00%	0	0.0000	0.0000	0.0000
PassedPiece	0.00%	0	0.0000	0.0000	0.0000
Sorting	0.00%	0	0.0000	0.0000	0.0000
Sorted1	0.00%	0	0.0000	0.0000	0.0000
Sorted2	0.00%	0	0.0000	0.0000	0.0000
Sorted3	0.00%	0	0.0000	0.0000	0.0000
ConvTampa	0.00%	0	0.0000	0.0000	0.0000
Drain	0.00%	0	0.0000	0.0000	0.0000
Drain1	0.00%	0	0.0000	0.0000	0.0000

Powering up/down Time

Object	Portion	Count	Sum	Mean Value	Standard Deviation
Source	0.00%	0	0.0000	0.0000	0.0000
Distribution	0.00%	0	0.0000	0.0000	0.0000
ConvCreator	0.00%	0	0.0000	0.0000	0.0000
RejectedPiece	0.00%	0	0.0000	0.0000	0.0000
TestPiece	0.00%	0	0.0000	0.0000	0.0000
PickAndPlaceAssembly	0.00%	0	0.0000	0.0000	0.0000
PickAndPlace	0.00%	0	0.0000	0.0000	0.0000
PassedPiece	0.00%	0	0.0000	0.0000	0.0000
Sorting	0.00%	0	0.0000	0.0000	0.0000
Sorted1	0.00%	0	0.0000	0.0000	0.0000
Sorted2	0.00%	0	0.0000	0.0000	0.0000
Sorted3	0.00%	0	0.0000	0.0000	0.0000
ConvTampa	0.00%	0	0.0000	0.0000	0.0000
Drain	0.00%	0	0.0000	0.0000	0.0000
Drain1	0.00%	0	0.0000	0.0000	0.0000

Stopped Time

Object	Portion	Count	Sum	Mean Value	Standard Deviation
Source	0.00%	0	0.0000	0.0000	0.0000
Distribution	0.00%	0	0.0000	0.0000	0.0000
ConvCreator	0.00%	0	0.0000	0.0000	0.0000
RejectedPiece	0.00%	0	0.0000	0.0000	0.0000
TestPiece	0.00%	0	0.0000	0.0000	0.0000
PickAndPlaceAssembly	0.00%	0	0.0000	0.0000	0.0000
PickAndPlace	0.00%	0	0.0000	0.0000	0.0000
PassedPiece	0.00%	0	0.0000	0.0000	0.0000
Sorting	0.00%	0	0.0000	0.0000	0.0000
Sorted1	0.00%	0	0.0000	0.0000	0.0000
Sorted2	0.00%	0	0.0000	0.0000	0.0000
Sorted3	0.00%	0	0.0000	0.0000	0.0000
ConvTampa	0.00%	0	0.0000	0.0000	0.0000
Drain	0.00%	0	0.0000	0.0000	0.0000
Drain1	0.00%	0	0.0000	0.0000	0.0000

Failed Time

Object	Portion	Count	Sum	Mean Value	Standard Deviation
Source	0.00%	0	0.0000	0.0000	0.0000
Distribution	0.00%	0	0.0000	0.0000	0.0000
ConvCreator	0.00%	0	0.0000	0.0000	0.0000
RejectedPiece	0.00%	0	0.0000	0.0000	0.0000
TestPiece	0.00%	0	0.0000	0.0000	0.0000
PickAndPlaceAssembly	0.00%	0	0.0000	0.0000	0.0000
PickAndPlace	0.00%	0	0.0000	0.0000	0.0000
PassedPiece	0.00%	0	0.0000	0.0000	0.0000
Sorting	0.00%	0	0.0000	0.0000	0.0000
Sorted1	0.00%	0	0.0000	0.0000	0.0000
Sorted2	0.00%	0	0.0000	0.0000	0.0000
Sorted3	0.00%	0	0.0000	0.0000	0.0000
ConvTampa	0.00%	0	0.0000	0.0000	0.0000
Drain	0.00%	0	0.0000	0.0000	0.0000
Drain1	0.00%	0	0.0000	0.0000	0.0000

Paused Time

Object	Portion	Count	Sum	Mean Value	Standard Deviation
Source	100.00%	6	16.3943	2.7324	0.6555
Distribution	24.40%	2	4.0000	2.0000	0.0000
ConvCreator	12.20%	2	2.0000	1.0000	0.0000
RejectedPiece	100.00%	1	16.3943	16.3943	0.0000
TestPiece	84.75%	3	13.8943	4.6314	5.4395
PickAndPlaceAssembly	33.55%	2	5.5000	2.7500	2.1213
PickAndPlace	63.40%	2	10.3943	5.1972	1.0647
PassedPiece	91.19%	1	14.9500	14.9500	0.0000
Sorting	100.00%	1	16.3943	16.3943	0.0000
Sorted1	100.00%	1	16.3943	16.3943	0.0000
Sorted2	100.00%	1	16.3943	16.3943	0.0000
Sorted3	100.00%	1	16.3943	16.3943	0.0000
ConvTampa	88.45%	2	14.5000	7.2500	4.2426
Drain	100.00%	1	16.3943	16.3943	0.0000
Drain1	100.00%	1	16.3943	16.3943	0.0000

Empty Time

Resource Statistics - Drain Statistics

Object	Working	Set-up	Waiting	Stopped	Failed	Paused	Mean Life Time	Mean Exit Time	Total Throughput	Throughput per Hour	Throughput per Day
Drain	5635.24%	5635.24%	5635.24%	5635.24%	5635.24%	5635.24%					
Drain1	5635.24%	5635.24%	5635.24%	5635.24%	5635.24%	5635.24%					

Cumulated Statistics of the Parts which the Drain Deleted

Object	All Types
Drain	0
Drain1	0

Part Types which the Drains Deleted

Service Statistics - Importer Statistics

Object	Total Delay	Reason for the Delay		
		Waiting for Services and Parts	Waiting for Parts	Waiting for Set-up Exporters
Source	0.00%	0.00%	0.00%	0.00%
Distribution	0.00%	0.00%	0.00%	0.00%
PickAndPlaceAssembly	48.15%	48.15%	48.15%	0.00%
PickAndPlace	0.00%	0.00%	0.00%	0.00%
Sorting	0.00%	0.00%	0.00%	0.00%

Importers Waiting for Services and Parts

Object	Waiting for Services and Parts	Count	Sum	Mean Value	Standard Deviation
Source	0.00%	0	0.0000	0.0000	0.0000
Distribution	0.00%	0	0.0000	0.0000	0.0000
PickAndPlaceAssembly	48.15%	2	7.8943	3.9472	5.3073
PickAndPlace	0.00%	0	0.0000	0.0000	0.0000
Sorting	0.00%	0	0.0000	0.0000	0.0000

Waiting Times for Services and Parts

Object	Waiting for Parts	Count	Sum	Mean Value	Standard Deviation
Source	0.00%	0	0.0000	0.0000	0.0000
Distribution	0.00%	0	0.0000	0.0000	0.0000
PickAndPlaceAssembly	48.15%	2	7.8943	3.9472	5.3073
PickAndPlace	0.00%	0	0.0000	0.0000	0.0000
Sorting	0.00%	0	0.0000	0.0000	0.0000

Waiting Times for Parts

Object	Waiting for Set-up Exporters	Count	Sum	Mean Value	Standard Deviation	Waiting Time while the Resource Waited	Waiting Time while the Resource Worked	Waiting Time while the Resource was Blocked
Source	0.00%	0	0.0000	0.0000	0.0000	0.00%	0.00%	0.00%
Distribution	0.00%	0	0.0000	0.0000	0.0000	0.00%	0.00%	0.00%
PickAndPlaceAssembly	0.00%	0	0.0000	0.0000	0.0000	0.00%	0.00%	0.00%
PickAndPlace	0.00%	0	0.0000	0.0000	0.0000	0.00%	0.00%	0.00%
Sorting	0.00%	0	0.0000	0.0000	0.0000	0.00%	0.00%	0.00%

Waiting Times for Set-up Exporters