UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

JONATHAN ALVES BATISTA

SISTEMA DE SUPERVISÃO SCADA: APLICAÇÃO DE BIBLIOTECAS OPEN SOURCE

PONTA GROSSA

2024

JONATHAN ALVES BATISTA

SISTEMA DE SUPERVISÃO SCADA: APLICAÇÃO DE BIBLIOTECAS OPEN SOURCE

Scada supervision sytem: application of open source libraries

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica do Curso de Bacharelado em Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná.

Orientador: Hugo Valadares Siqueira

PONTA GROSSA

2024

JONATHAN ALVES BATISTA

SISTEMA DE SUPERVISÃO SCADA: APLICAÇÃO DE BIBLIOTECAS OPEN SOURCE

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica do Curso de Bacharelado em Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 24/junho/2024

Hugo Valadares Siqueira
Doutorado
Universidade Tecnológica Federal do Paraná

Fernanda Cristina Correa
Doutorado
Universidade Tecnológica Federal do Paraná

Yara de Souza Tadano
Doutorado
Universidade Tecnológica Federal do Paraná

PONTA GROSSA

2024

Dedico este trabalho à minha família, pelos momentos de ausência.

AGRADECIMENTOS

Gostaria de expressar meus agradecimentos a todos aqueles que contribuíram para a realização deste trabalho e para o sucesso desta jornada acadêmica.

Aos meus pais, sou profundamente grato pela base sólida que me proporcionaram, oferecendo apoio e incentivo ao longo dessa trajetória.

Ao meu orientador, Hugo Valadares Siqueira, dedico um agradecimento especial pela confiança depositada em meu trabalho e pelo amparo durante o desenvolvimento. Seu apoio e orientação foram cruciais para o alcance dos resultados obtidos.

Aos amigos de graduação, que sempre estiveram presentes e acreditaram em meu potencial, agradeço por compartilharem comigo essa jornada desafiadora e enriquecedora.

Aos membros da banca, agradeço sinceramente por aceitarem o convite para avaliar meu trabalho. Sua expertise e contribuições foram essenciais para a qualidade final deste estudo.

RESUMO

Desde a primeira Revolução Industrial, o setor industrial tem experimentado uma evolução constante em termos de tecnologia e automação. Durante a primeira era, máquinas mecânicas foram introduzidas e substituíram a força humana na produção. Com a segunda Revolução Industrial, a produção foi ampliada com a utilização de energia elétrica e máquinas elétricas. A terceira Revolução Industrial, conhecida como Revolução Digital, trouxe a integração da tecnologia da informação e da automação para a produção industrial. Hoje, a quarta Revolução Industrial, também conhecida como a era da automação, tem como foco a criação de sistemas inteligentes e interconectados. Sistemas de supervisão SCADA (Supervisory Control and Data Acquisition) fazem parte desta era e são amplamente utilizados para controlar e monitorar processos industriais. Além disso, com o crescente uso de soluções open source, os sistemas SCADA são acessíveis a um número cada vez maior de empresas, permitindo uma automação eficiente e acessível. Este trabalho aborda um sistema de supervisão SCADA, aplicado a uma empresa do ramo alimentício situada nos Campos Gerais.

Palavras-chave: scada; supervisório; industria; automação; opc.

ABSTRACT

Since the first Industrial Revolution, the industrial sector has experienced a constant evolution in terms of technology and automation. During the first era, mechanical machines were introduced and replaced human labor in production. With the second Industrial Revolution, production was expanded through the use of electric power and electrical machines. The third Industrial Revolution, known as the Digital Revolution, brought the integration of information technology and automation to industrial production. Today, the fourth Industrial Revolution, also known as the era of automation, focuses on creating intelligent and interconnected systems. SCADA (Supervisory Control and Data Acquisition) supervisory systems are part of this era and are widely used to control and monitor industrial processes. Moreover, with the growing use of open-source solutions, SCADA systems are accessible to an increasing number of companies, enabling efficient and affordable automation. This work addresses a SCADA supervision system, applied to a food industry company located in the Campos Gerais region.

Keywords: scada; supervisory; industry; automation; opc.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGEM DE CÓDIGOS FONTE

LISTA DE ABREVIATURAS E SIGLAS

Siglas

CPU	Central Processing Unit
IDE	Integrated Development Environment
IP	Internet Protocol
OPC	Open Platform Communications
OPC-DA	Open Platform Communications - Data Access
OPC-UA	Open Platform Communications - Unified Architecture
PLC	Programmable Logic Controller
SCADA	Supervisory Control and Data Acquisition
SQL	Structured Query Language
TA	Tecnologia da Automação
TI	Tecnologia da informação

SUMÁRIO

1 INTRODUÇÃO

O setor industrial tem experimentado uma evolução constante desde a primeira Revolução Industrial. Durante a primeira era, máquinas mecânicas foram introduzidas e substituíram a força humana na produção. Com a segunda Revolução Industrial, a produção foi ampliada com a utilização de energia elétrica e máquinas elétricas. A terceira Revolução Industrial, também conhecida como Revolução Digital, levou a automação e tecnologia da informação ao processo de fabricação industrial (SCHWAB, 2019).

A quarta Revolução Industrial, também conhecida como a era da automação, tem como objetivo construir sistemas interconectados e inteligentes. Nesta época, os sistemas Supervisory Control and Data Acquisition (SCADA) são amplamente utilizados para monitorar e controlar processos industriais, oferecendo eficiência, precisão e segurança (SCHWAB, 2019).

Os sistemas SCADA são utilizados na maioria dos processos industriais, como a fabricação de aço, geração e distribuição de energia (convencional e nuclear), e química, permitindo a supervisão e aquisição de dados para o controle eficiente e seguro dessas operações.(DANEELS; SALTER, 1999)

No caso específico da indústria alimentícia, o uso de sistemas SCADA tem sido fundamental para supervisão e controle dos processos de produção, permitindo a monitoração remota de máquinas e equipamentos, o que agiliza a tomada de decisões.

Diante disso, fica evidente que a automação industrial é uma necessidade cada vez maior para garantir eficiência e segurança nos processos de produção. O uso de sistemas SCADA tem permitido uma supervisão precisa dos processos industriais, tornando a produção mais ágil e eficiente. No entanto, muitas empresas têm desafios financeiros ao implementar tais sistemas devido ao alto custo de soluções proprietárias. Felizmente, o crescente uso de bibliotecas *open source* (código aberto) permitiu a várias delas ter acesso a soluções eficientes para automação industrial.

No presente trabalho, foi abordado um sistema de supervisão SCADA, aplicado a uma empresa do ramo alimentício situada nos Campos Gerais, que se dedica à fabricação de comidas pré-preparadas saudáveis. O objetivo foi controlar e monitorar as 4 autoclaves da empresa, que realizam o cozimento dos preparos em embalagem a vácuo com vapor em alta pressão, garantindo assim a preservação dos nutrientes dos alimentos.

Para alcançar esse objetivo, foi utilizada a biblioteca *open source* Sharp7, que foi implementada no Visual Studio, utilizando a linguagem de programação `C#`. Essa biblioteca permite a comunicação entre o sistema de supervisão e as autoclaves através da conexão com Programmable Logic Controller (PLC) Siemens. Com isso, a empresa consegue ter acesso a uma solução flexível e escalável, ao mesmo tempo em que tem a possibilidade de customizar e aprimorar o sistema de acordo com as suas necessidades, sem precisar arcar com altos custos.

Espera-se que este trabalho possa contribuir para a melhoria dos processos de fabricação de comida pré-preparada saudável da empresa, por meio da implementação de tecnologias

avançadas de supervisão. Além disso, este estudo pode ser útil para outros atores do ramo alimentício que buscam soluções eficientes e seguras para o controle e monitoramento de seus processos produtivos. Esse trabalho é composto por referencial teórico com embasamento na literatura, materiais e métodos apresentando o desenvolvimento proposto e os resultados obtidos através do projeto.

1.1 Objetivo geral

O objetivo geral deste trabalho é demonstrar a aplicação da biblioteca *open source* Sharp7 na construção de um sistema SCADA para o ramo alimentício, a fim de controlar e monitorar as autoclaves de uma empresa e garantir a qualidade dos alimentos produzidos.

1.2 Objetivos específicos

Os objetivos gerais deste trabalho poderão ser alcançados por meio dos seguintes objetivos específicos:

- Criar a interface para comunicação com os PLCs da Siemens utilizando a biblioteca Sharp7;
- Estruturar a interface do usuário;
- Realizar a integração com banco de dados MySQL;
- Criar telas e tabelas para inserção e manipulação de registros de usuários, receitas e produtos;
- Criar tela de execução de receitas e visualizações;
- Visualizações de alarme e registro de produção.

2 REFERENCIAL TEÓRICO

Este capítulo realiza uma breve revisão bibliográfica e teórica sobre PLC, sistema SCADA, Open Platform Communications (OPC), *open source*, snap e Sharp7, banco de dados e Integrated Development Environment (IDE).

2.1 PLC

No final dos anos 1960, a *General Motors* especificou critérios de *design* para o primeiro PLC (*Programmable Logic Controller* - Controlador Lógico Programável) devido à necessidade de substituir os painéis de relés com fiação fixa usados em linhas de montagem automatizadas. Alguns dos critérios incluíam facilidade de programação, manutenção fácil, tamanho compacto e capacidade de comunicação com sistemas de coleta de dados centralizados (ERICKSON, 1996).

Os PLCs originalmente foram criados para substituir dispositivos eletromecânicos, como relés, em aplicações de controle industrial. Com o tempo, eles evoluíram para se tornar computadores padronizados, porém, com propriedades especiais para atender às necessidades da automação industrial. Suas características incluem modularidade, ampla variedade de opções de conexão, modelo de execução cíclico e priorizado, ambiente de execução estável e acessibilidade de programação para não especialistas. Embora o *hardware* tenha permanecido relativamente estável desde a década de 1980, as demandas de programação evoluíram para acomodar um maior número de tarefas, aplicações diversas e requisitos de controle mais complexos. A comunidade de PLCs respondeu a essas mudanças com novas linguagens de programação, ciclos de varredura especializados e melhorias de *hardware* (SEHR *et al.*, 2020).

2.2 Sistema SCADA

O SCADA, que significa *Supervisory Control And Data Acquisition*, é um sistema amplamente utilizado na indústria para o controle e aquisição de dados de processos industriais. Ele é um pacote de *software* posicionado sobre *hardware*, geralmente interconectado com controladores lógicos programáveis (PLCs) ou outros módulos comerciais. Em termos de arquitetura, o sistema possui duas camadas principais: a do cliente, que lida com a interação homem-máquina, e a do servidor, que lida com a maioria das atividades de controle de dados do processo. Os servidores se comunicam com dispositivos no campo por meio de controladores de processo, como PLCs (**??**).

Em termos de desenvolvimento de aplicativos, os sistemas SCADA oferecem ferramentas para configurar parâmetros, desenvolver gráficos, definir tendências, configurar alarmes e automatizar tarefas. Eles geralmente suportam bibliotecas de objetos gráficos que podem ser

vinculados a variáveis do processo e animados à medida que os valores mudam. Além disso, muitos sistemas SCADA estão incorporando tecnologias da *web* e padrões como OPC para melhorar a interoperabilidade e a comunicação com dispositivos de terceiros (**??**).

2.3 OPC - *Open Platform Communications*

Comunicação de Plataforma Aberta (OPC), originado da sigla OLE *(object linking and embedding)* para Controle de Processos, é um padrão de interoperabilidade que facilita a troca segura de dados na automação industrial e em várias indústrias. A Fundação OPC é responsável pelo gerenciamento desse padrão, que consiste em especificações que definem interfaces entre dispositivos de diferentes fabricantes. Inicialmente lançado em 1996 para abstrair protocolos de PLCs em uma interface padronizada, o OPC evoluiu de suas origens centradas no Windows (conhecidas como OPC Clássico) para enfrentar desafios em segurança e modelagem de dados.

As especificações Open Platform Communications - Unified Architecture (OPC-UA) foram desenvolvidas para atender a essas necessidades em evolução, oferecendo uma arquitetura de plataforma aberta à prova de futuro, escalável e extensível. Isso possibilita o fluxo de informações sem problemas entre dispositivos, com suporte a acesso a dados em tempo real, monitoramento de alarmes, recuperação de dados históricos e outras aplicações. O OPC encontrou ampla adoção em setores como manufatura, automação predial, petróleo e gás, energia renovável e serviços públicos (FOUNDATION, 2023). As comunicações OPC têm se destacado como um *middleware* eficaz, principalmente no setor industrial.

Um forte concorrente para liderar a padronização e integração de sistemas em *frameworks* é a Arquitetura Unificada (GONZÁLEZ *et al.*, 2019). A OPC-UA, lançada em 2008, é uma estrutura orientada a serviços independentes de plataforma que integra as funcionalidades do OPC *Classic*. Com uma abordagem multicamadas, busca equivalência funcional, independência de plataforma, segurança e extensibilidade, o OPC-UA, funcionalmente equivalente ao OPC *Classic*, oferece recursos aprimorados, como descoberta de servidores, representação hierárquica de dados e leitura/gravação sob demanda. Operando em várias plataformas, aborda questões de segurança como *firewall*, criptografia e autenticação. A arquitetura multicamadas é extensível, permitindo a incorporação de novas tecnologias sem afetar a compatibilidade com versões anteriores. A estrutura de modelagem de informações transforma dados em informações, com mecanismos de acesso, como consulta e notificação. Na comunicação Cliente-Servidor, segue a Arquitetura Orientada a Serviços (SOA) (FOUNDATION, 2023).

Antes da introdução do OPC-UA, a OPC *Foundation* desenvolveu várias especificações relacionadas ao OPC, incluindo o Open Platform Communications - Data Access (OPC-DA), também conhecido como OPC *Classic* (FARNHAM; BARILLERE, 2011). Essas especificações, estruturadas na tecnologia Windows, faziam uso do COM/DOM (Distributed Component Object Model) com o objetivo de simplificar a troca de dados entre diferentes componentes de software.

No contexto do OPC-DA, as diretrizes estabeleciam os parâmetros para a troca de dados, englobando informações cruciais, como valores, tempo e informação de qualidade. No entanto, à medida que as exigências tecnológicas progrediram e a necessidade de interoperabilidade entre sistemas de plataformas distintas se intensificou, a OPC *Foundation* lançou o OPC-UA em 2008. Essa arquitetura recém-introduzida, caracterizada por sua independência de plataforma, consolidou e refinou as funcionalidades do OPC *Classic*, enfrentando desafios contemporâneos e facilitando uma comunicação mais eficaz e segura em ambientes variados. O OPC-UA não apenas ampliou as capacidades do OPC-DA, representando uma evolução tecnológica, mas também atendeu às crescentes expectativas de flexibilidade, segurança e integração nos sistemas de automação industrial (FOUNDATION, 2023).

2.4 *Open Source*

O termo *"Open Source"* engloba duas interpretações principais: "*software* livre"e "código aberto". A definição de "*software* livre"pela *Free Software Foundation* destaca a liberdade dos usuários para executar, copiar, distribuir, estudar, alterar e melhorar o *software*, enfatizando a necessidade de acesso ao código-fonte (STALLMAN, 2023). Por outro lado, o termo "código aberto"surgiu posteriormente, sendo uma tentativa mais cautelosa de expressar o mesmo conceito, embora seja usado de maneira intercambiável.

A ambiguidade do termo surge de diferentes interpretações sobre a acessibilidade do código-fonte, e sua definição não está totalmente consolidada. No entanto, ambos os termos são frequentemente utilizados como sinônimos, referindo-se a qualquer forma de desenvolvimento de *software* em que o código-fonte é livremente acessível à comunidade global. Vale ressaltar que o *software open source* também pode ser bem-sucedido comercialmente, desafiando a ideia de que *software* comercial é exclusivamente proprietário (FUGGETTA, 2003).

O interesse pelo *Open Source* é notável em várias áreas. Uma comunidade diversificada, especialmente na pesquisa industrial e acadêmica, promove essa abordagem como resposta ao desconforto com os custos e restrições de produtos comerciais. Empresas importantes, como Sun e IBM, veem no código aberto uma estratégia para desafiar a hegemonia da Microsoft. Além disso, instituições governamentais, especialmente na Europa, mostram interesse no *Open Source* como uma estratégia para equilibrar o domínio tecnológico e promover uma indústria de *software* mais forte. A disponibilidade irrestrita do código-fonte é vista como uma vantagem para abordar preocupações de segurança e aumentar a independência em relação a fornecedores específicos (FUGGETTA, 2003).

2.5 Interface de Programação de Aplicações (API)

As Interfaces de Programação de Aplicações (APIs), do inglês *Application Programming Interface*, desempenham um papel fundamental no desenvolvimento de software. Desde os primeiros computadores, as APIs foram se aprimorando para facilitar a troca entre programas. Consideradas a espinha dorsal do ecossistema digital, as APIs interconectam pessoas, aplicativos e sistemas. Esse fenômeno é notável na economia de aplicativos móveis, em que as APIs desempenham um papel crucial, garantindo a confiabilidade e a interoperabilidade dos aplicativos para os desenvolvedores.

Além das definições técnicas as APIs são reconhecidas como facilitadoras da reutilização pragmática, melhorando a produtividade no desenvolvimento de softwares e permitindo uma troca de informações (OFOEDA; BOATENG; EFFAH, 2019).

2.5.1 Snap7

o Snap7 é uma suíte de comunicação Ethernet de código aberto projetada para integração nativa com PLCs Siemens S7. Ela oferece suporte a diversas plataformas, incluindo Windows, Linux, BSD, Oracle Solaris 11 e Apple OSX, tornando-a versátil desde servidores blade até dispositivos pequenos como o *Raspberry* PI.

O design nativo para múltiplas arquiteturas garante compatibilidade com diferentes Central Processing Unit (CPU)s. O Snap7 oferece modelos de transferência de dados tanto síncronos quanto assíncronos. É também conhecido por sua simplicidade, independência de bibliotecas de terceiros e fácil integração com diferentes linguagens de programação. Além disso, fornece *wrappers* orientados a objetos de alto nível, demonstrações abrangentes e documentação detalhada, tornando-o acessível para uma ampla variedade de usuários e plataformas (SNAP7HOMEPAGE, s.d.).

2.5.2 Sharp7

O Sharp7 é a versão em C# do Snap7 Client, representando uma implementação pura em C# do *S7Protocol*, sem depender de interfaces que carregam bibliotecas externas. Este projeto é apresentado como um único arquivo-fonte contendo classes que podem ser diretamente incorporadas em projetos .NET para comunicação com PLCs S7 (SNAP7HOMEPAGE, s.d.).

Destinado a funcionar em hardware .NET de pequeno porte ou em projetos extensos que não exigem funções de controle avançadas, o Sharp7 oferece código gerenciado "*safe*"padrão em C#, sem dependências externas. Com cabeçalhos de protocolo compactados para melhorar o desempenho, ele simplifica o acesso a todos os tipos S7 e é compatível com uma variedade

de *hardware* que suporta .NET Core. Além disso, pode ser utilizado no *Universal Windows Platform*, incluindo o Win10 IoT para Raspberry (SNAP7HOMEPAGE, s.d.).

A distribuição do Sharp7 é feita como uma biblioteca de código-fonte sob a Licença Pública Geral Reduzida do GNU versão 3.0 (LGPLv3), permitindo a inclusão em *firmwares* comerciais sem a obrigatoriedade de compartilhar o código-fonte da aplicação. No entanto, uma pequena menção de créditos é apreciada quando incorporada em aplicações (SNAP7HOMEPAGE, s.d.).

2.6 Banco de Dados

Um banco de dados é uma coleção estruturada de dados organizados para facilitar a busca e a recuperação eficientes por um computador. Trata-se de uma coleção organizada de itens para permitir maior agilidade no manuseio dos dados (MANOVICH, 2015).

2.6.1 NoSQL e SQL

O termo NoStructured Query Language (SQL), do inglês *Not Only SQL*, representa um tipo de armazenamento de dados alternativo aos bancos de dados tradicionais, desenvolvido com o objetivo de tratar um grande volumes de informações, como o Facebook. O NoSQL é um sistema de gerenciamento de banco de dados não relacional, caracterizado pela rápida recuperação de informações e portatibilidade. A consulta do NoSQL não utiliza a linguagem SQL tradicional, sendo possível acessa-los por meio de formatos como XML (SHARMA; DAVE, 2012).

O embate entre o SQL e NoSQL tem gerado longas discussões, com alguns sugerindo o declínio da era do SQL. No entanto essas categorias de bancos de dados abordam distintos conjuntos de problemas. O NoSQL, exemplificados por HBase, Cassandra e MongoDB, são reconhecidos por sua habilidade em escalar horizontalmente em *clusters* massivos, proporcionando baixa latência e um modelo de programação simplificado, com destaque para armazenamento em formato de chave-valor. Já o SQL, como MySQL e PostgreSQL, oferecem transações ACID cruciais para casos de uso que demandam alta consistência e precisão, como sistemas bancários e de mercado financeiro. Enquanto os bancos de dados NoSQL recebem elogios pela escalabilidade e flexibilidade, os bancos de dados SQL têm a vantagem de uma linguagem comum (SQL) e um modelo relacional testado e comprovado, adequado a diversas necessidades de armazenamento de dados (BARTHOLOMEW, 2010).

2.7 Ambiente de Desenvolvimento Integrado (IDE)

A designação IDE, do inglês: *Integrated Development Environment*, representa um conjunto de ferramentas reunidas em uma interface gráfica para simplificar o processo de criação de aplicações, incluindo elementos como editor de código-fonte, automação de compilação local e debugger. Os desenvolvedores se beneficiam dos IDEs, pois permitem a rápida criação de novas aplicações sem a necessidade de ajustes manuais e integração de utilitários. Recursos como destaque de sintaxe, preenchimento automático e detecção de erros em tempo real são essenciais para economizar tempo, especialmente para aqueles novos em um projeto.

Além de acelerar o desenvolvimento, os IDEs desempenham um papel crucial na transformação digital, facilitando a organização do fluxo de trabalho, identificando erros durante a codificação e promovendo a eficiência. Apesar de ser possível programar sem um IDE, a padronização e a automação oferecidas por essas ferramentas geralmente superam outras abordagens. O mercado oferece diversas opções de IDEs, cada uma com características distintas, como suporte a linguagens específicas, compatibilidade com sistemas operacionais, funcionalidades de automação e impacto no desempenho do sistema. A escolha entre IDEs locais e baseados na nuvem também influencia fatores como padronização, segurança e distribuição eficiente de tarefas prolongadas (HAT, s.d.).

2.7.1 Visual Studio

O Visual Studio é um poderoso Ambiente de Desenvolvimento Integrado (IDE) que abrange todo o ciclo de desenvolvimento. Além de escrever, editar e depurar código, oferece compiladores, ferramentas de preenchimento de código, controle de código-fonte e diversas extensões. Com suporte a várias linguagens, o Visual Studio permite evoluir desde a criação de simples programas até o desenvolvimento e implantação de aplicativos complexos, abrangendo diversas plataformas e cenários, como .NET, C++, ASP.NET, desenvolvimento móvel e de área de trabalho multiplataforma, e criação de interfaces web responsivas em C# (LEE, s.d.).

3 MATERIAIS E MÉTODOS

Este estudo se concentrou na realização de uma pesquisa aplicada no contexto de investigar, analisar e avaliar a aplicação de tecnologias de automação *open source* no setor industrial. O propósito foi descritivo, com o objetivo de demonstrar a aplicação da biblioteca *open source* Sharp7 na construção de um sistema de supervisão SCADA para o ramo alimentício, a fim de controlar e monitorar as autoclaves de uma empresa dos Campos Gerais, visando garantir a qualidade dos alimentos produzidos.

3.1 Materiais

Na seção de materiais deste trabalho, foram empregados exclusivamente um computador e softwares específicos para o desenvolvimento e implementação. Os softwares utilizados incluem o Visual Studio, o Sharp7, o MySQL, entre outros.

As razões para ter escolhido o Sharp7 incluíram sua natureza open source, que proporcionou custo-efetividade por ser gratuito e eliminar a necessidade de licenças caras. Além disso, o código-fonte aberto ofereceu transparência, permitindo auditorias e personalizações conforme necessário. O Sharp7 mostrou-se altamente compatível com PLCs Siemens S7, garantindo uma comunicação robusta e facilitando a leitura e escrita de dados, essencial para a sincronização entre o sistema SCADA e os dispositivos de controle. A biblioteca também foi flexível e customizável, permitindo adaptações específicas para o projeto, e contou com suporte contínuo da comunidade de desenvolvedores, que proporcionou atualizações e correções regulares. Por fim, o Sharp7 integrou-se facilmente com o ambiente de desenvolvimento Visual Studio, simplificando o desenvolvimento e a manutenção do sistema.

A escolha do MySQL para o sistema SCADA baseou-se na vantagem de ser open source, além de contar com uma ampla comunidade de desenvolvedores e suporte técnico disponível, que oferecem recursos e atualizações constantes, assegurando que o sistema permaneça atualizado e eficiente.

O Visual Studio foi escolhido devido à familiaridade da equipe com a ferramenta e sua capacidade de integrar-se facilmente com outras bibliotecas. Isso garantiu velocidade no desenvolvimento e compatibilidade entre os diversos componentes do sistema SCADA, facilitando a implementação e manutenção do software industrial de forma eficiente e eficaz.

3.2 Métodos

O projeto foi conduzido por três equipes, cada uma responsável por uma área específica. A equipe de Tecnologia da informação (TI) ficou encarregada pelo desenvolvimento web e pela criação dos bancos de dados. A equipe de Tecnologia da Automação (TA) foi responsável pelo

desenvolvimento de lógica de controle da autoclave com o PLC. Por fim, a equipe SCADA assumiu a responsabilidade pela aquisição de dados do PLC, desenvolvimento da interface do usuário e integração com o banco de dados.

Este trabalho se concentrou na parte conduzida pela equipe SCADA. A seguir são descritas as etapas desenvolvidas:

- Preparação do ambiente de Desenvolvimento;
- Estruturação do Projeto;
- Construção dos modelos;
- Integração da biblioteca Sharp7;
- Integração com o banco de dados MySQL;
- Interface do usuário.

3.2.1 Preparação do ambiente de Desenvolvimento

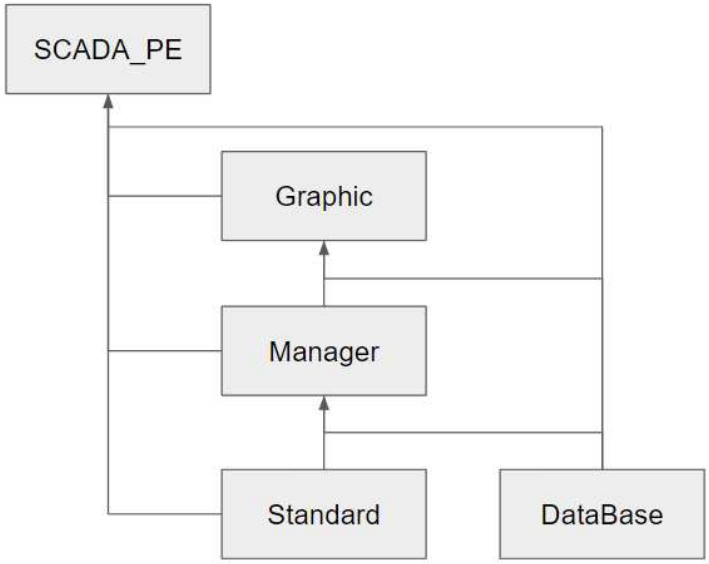
A preparação do ambiente de desenvolvimento envolveu duas etapas distintas que foram fundamentais para o projeto:

- **Configuração do Ambiente de Desenvolvimento:** Nesta etapa, foi necessário configurar o ambiente no qual a aplicação seria desenvolvida. Isso incluiu instalar e configurar ferramentas de desenvolvimento, como o IDE, de modo que foi escolhido o Visual Studio 2017 *Enterprise*. A seguir, foram definidas as configurações de sistema operacional (Windows 10, em Português/Brasil);
- **Definição do Ambiente de Execução:** Após configurar o ambiente de desenvolvimento, foi crucial definir o ambiente no qual a aplicação seria executada. Isso envolveu escolher o sistema operacional, configurar as dependências necessárias, como bibliotecas e *frameworks*, e garantir que o ambiente de execução seria compatível com as especificações do projeto. Esta etapa foi essencial para garantir que a aplicação funcionasse corretamente após o desenvolvimento e durante sua implantação em ambiente real.

3.2.2 Estruturação do Projeto

A estruturação do projeto foi delineada durante o processo de desenvolvimento, sendo organizada através de cinco divisões distintas: *Stardard*, *DataBase*, *Manager*, *Graphic* e Scada_PE, conforme pode-se visualizar na Figura 1, cada uma desempenhando um papel especifico.

Figura 1 – Fluxograma de estruturação do projeto



Fonte: Autoria própria (2024).

3.2.2.1 *Stardard*

O projeto *Stardard* foi realizado como uma biblioteca modular, com a função de definir modelos. Seu objetivo principal foi garantir que, ao acionar essa biblioteca em outros projetos, fosse possível instanciar os tipos de modelos de maneira consistente e padronizada, mantendo sempre o mesmo padrão.

Essa biblioteca proporciona uma estrutura centralizada para a definição e gestão dos tipos de modelos, permitindo que essas definições sejam compartilhadas e utilizadas facilmente em diversos projetos. Ao utilizar o *Stardard*, os desenvolvedores garantem a uniformidade e coerência na manipulação de modelos em diferentes partes de uma aplicação, independentemente do contexto ou projeto que esteja sendo trabalhado.

Desta forma, o *Stardard* desempenhou um papel fundamental na reutilização de código, na simplificação do desenvolvimento e na manutenção da consistência de todos os aspectos que relacionem os modelos.

3.2.2.2 *DataBase*

O Projeto *DataBase* teve como objetivo a manipulação de dados, lidando com todas as interações com o banco de dados. Ele foi projetado para garantir a integridade, segurança e eficiência no armazenamento e recuperação de informações relevantes para o funcionamento da aplicação. Incluiu operações de leitura, gravação, atualização e exclusão de dados.

3.2.2.3 *Manager*

Para o projeto *Manager* concentrou-se na lógica operacional da aplicação, implementando todas as funcionalidades principais e as lógicas necessárias. Coordenou diferentes operações de aplicação, assegurando que as ações do usuário fossem processadas corretamente e que o sistema funcionasse conforme o esperado.

3.2.2.4 Graphic

Esta etapa do projeto foi responsável por toda a interface do usuário, abrangendo o *design* visual e a interação do usuário com a aplicação. Esta etapa engloba a criação de telas, elementos gráficos e fluxos de navegação, garantindo uma experiência intuitiva e agradável para o usuário final.

3.2.2.5 *Scada_*PE

Esta etapa representou a própria aplicação a ser executada. Foram reunidas as todas as partes do projeto e proporcionou a funcionalidade necessária final para o usuário. Neste ponto todas as etapas foram integradas e testadas para garantir um bom funcionamento.

3.2.3 Construção dos modelos

A modelagem envolveu definir classes, atributos, métodos e relações entre objetos, a fim de definir padrões para que a estrutura e a execução do programa sejam coerentes. Todos os modelos criados foram sintetizados em classes na biblioteca *Standard*.

3.2.3.1 Usuário

A gestão de usuários em um sistema SCADA é crucial para a segurança, eficiência e integridade operacional do sistema. Primeiramente, ela garante a segurança e o controle de acesso por meio da autenticação e autorização, assegurando que apenas pessoas autorizadas possam acessar o sistema e prevenindo acessos não autorizados e potenciais ameaças. Além disso, permite a definição de permissões granulares, em que diferentes usuários têm diferentes níveis de acesso de acordo com suas funções e responsabilidades específicas.

A primeira etapa realizada para a gestão de usuário foi definir o modelo de usuário a ser seguido. No quadro Quadro 1 pode-se observar a modelagem para o usuário, com campos e tipos de variáveis definidos.

Quadro 1 – Modelagem de usuário

Campo	Tipo	Descrição
Id_User	INT	Identificação do usuário no banco de dados
Name	STRING	Nome do usuário
Description	STRING	Descrição do usuário
DateTimeCreate	DATETIME	Data e horário de criação do usuário
Id_Attribute	INT	Atributo do usuário
Login	STRING	Login do usuário
Password	STRING	Senha do usuário
Id_UserCreator	INT	Identificação do usuário que o criou
Activate	BOOL	Verifica se o usuário está ativo

Fonte: Autoria própria (2024).

Com o modelo criado foi possível usá-lo para a elaboração das telas de visualizações, da tabela no banco de dados e da classe na logica do programa. Algumas definições deste modelo devem ser mencionadas:

- Nome e identificação devem ser únicos;
- Para criar um novo usuário, um deles já cadastrado deve estar logado no sistema, sendo o inicial o administrador do sistema, criado previamente;
- Nenhum usuário será deletado, para garantir que as informações permaneçam integras. Para rastreabilidade através do campo *Activate* é definido se o usuário está habilitado ou não;
- O campo *Id_Attribute* atribui o nível de acesso do usuário, em ordem crescente e acumulativa, onde o nível mais alto tem todas as permissões dos níveis abaixo, conforme a lista:

    1. AUSENTE

       (Nenhum usuário logado);
    2. OPERADOR

       (Permite alterar visualizações e partir produção);
    3. MANUTENÇÃO

       (Permite ativar em manual válvulas e outros dispositivos);
    4. SUPERVISOR

       (Permite alterar parâmetros de receitas);
    5. ADMINISTRADOR

       (Permite criar usuários).

3.2.3.2 Receitas

O controle de receita no sistema SCADA é essencial para garantir a padronização das produções industriais. Ele envolve a definição e aplicação de parâmetros operacionais específicos, assegurando que cada ciclo de produção siga diretrizes uniformes, resultando em produtos consistentes e de alta qualidade. Os principais benefícios incluem a redução de erros humanos, otimização de recursos, facilidade de reprodução em diferentes locais, rápida adaptação a novas demandas e melhoria contínua dos processos. Em suma, o controle de receitas proporciona consistência, eficiência e flexibilidade, resultando em produtos confiáveis e competitividade no mercado.

Todas as receitas criadas devem seguir o modelo proposto no quadro Quadro 2. As definições de identificação, nome e habilitação seguem as mesmas utilizadas para o modelo de usuário. Na receita temos a necessidade de fazer o versionamento para que seja permitido fazer as alterações mas não se perca a rastreabilidade. O campo *Version* corresponde a um numero inteiro, que se auto incrementará para cada alteração feita na receita.

Quadro 2 – Modelo de receita

Campo	**Tipo**	**Descrição**
Id_Recipe	INT	Identificação da receita
Name	STRING	Nome da receita
Description	STRING	Descrição da receita
List_Segments	LIST	Lista de segmentos
DateTimeCreate	DATETIME	Data e horário de criação
Id_User_Creator	INT	Identificação do usuário que a criou
Version	INT	Versão da receita
Activate	BOOL	Verifica se a receita está ativa

Fonte: Autoria própria (2024).

No modelo temos uma lista de segmento na qual cada um indica o parâmetro de receita que deverá ser executado em determinado momento durante a produção. Os parâmetros são compostos por campo de temperatura, de pressão, de tempo e de identificação de qual atividade está sendo realizada no momento. No quadro Quadro 3 pode-se visualizar a modelagem para um segmento. Abaixo encontra-se uma lista com a descrição de cada *Id_segment*.

- 0 - Saída;
- 1 - Desaeração;
- 2 - Aquecimento;
- 3 - Esterilização;
- 4 - Pré-Resfriamento;
- 5 - Resfriamento;

Quadro 3 – Modelagem de segmento

Campo	Tipo	Descrição
Time	REAL	Tempo em minutos
Temperature	REAL	Patamar de temperatura
Pressure	REAL	Patamar de pressão
Id_Segment	INT	Identificação do segmento

Fonte: Autoria própria (2024).

3.2.3.3 Produtos

A modelagem de produtos foi fundamental para essa estruturação, permitindo a gestão eficaz dos mesmos dentro do sistema, facilitando operações como consulta, atualização e manutenção dos dados dos produtos. Tal modelagem em sistemas de informação traz diversos benefícios, como a padronização das informações, melhorando a qualidade dos dados e a eficiência dos processos operacionais. No Quadro 4 pode-se observar a modelagem para os produtos.

Quadro 4 – Modelo de produto

Campo	Tipo	Descrição
Id_Product	INT	Identificação do produto
Name	STRING	Nome da receita
Description	STRING	Descrição da receita
DateTimeCreate	DATETIME	Data e horário de criação
Id_User_Creator	INT	Identificação do usuário que a criou
Version	INT	Versão da receita
Activate	BOOL	Verifica se a receita está ativo

Fonte: Autoria própria (2024).

3.2.3.4 Produção

Na modelagem da produção (Quadro 5) temos a junção de vários elementos como: qual receita foi usada, qual produto foi produzido e qual autoclave rodou essa produção.

3.2.3.5 PLC

Para salvar as informações de conexão de um PLC, tem-se a modelagem do mesmo no Quadro 6.

3.2.3.6 Módulo

Módulo é a representação de uma autoclave, contendo os campos necessários para as interações. No Quadro 7 tem-se a sua modelagem.

Quadro 5 – Modelo de produção

Campo	Tipo	Descrição
Id_Production	INT	Identificação da produção
Id_AutoClave	INT	Identificação da autoclave
Id_Lot_Cycle	INT	Identificação do lote de produção
Id_User	INT	Identificação do usuário
Id_Product	INT	Identificação do produto
Version_Product	INT	Versão do produto
Id_Recipe	INT	Identificação da receita
Version_Recipe	INT	Versão da receita
DateTime_Start	DATETIME	Data e horário do inicio da produção
DateTime_Stop	DATETIME	Data e horário do final da produção

Fonte: Autoria própria (2024).

Quadro 6 – Modelo de PLC

Campo	Tipo	Descrição
IP	STRING	IP do PLC
Rack	INT	Rack do PLC
Slot	INT	Posição do PLC no rack
Connected	INT	Estado de conexão com o PLC

Fonte: Autoria própria (2024).

Quadro 7 – Modelo de Módulo

Campo	Tipo	Descrição
PLC	PLCModel	Informações do PLC
Id	INT	Identificação do módulo
Id_Production	INT	Identificação da produção atual
Entradas_Analogicas	Entradas_Analogicas	Bloco de dados de entradas analógicas
Entradas_Digitais	Entradas_Digitais	Bloco de dados de entradas digitais
Motores	Motores	Bloco de dados motores
Saídas_Digitais	Saidas_Digitais	Bloco de dados de saídas digitais
Valvulas	Valvulas	Bloco de dados de válvulas
Time_Process	TIME	Tempo de execução da produção
BlockAlarm	BlockAlarm	Lista de alarmes
PID	PID	PID de controle da autoclave

Fonte: Autoria própria (2024).

3.3 Integração da biblioteca Sharp7

3.3.1 Instalação da biblioteca

Primeiramente, foi necessário adicionar a biblioteca Sharp7 ao projeto, o que pode ser feito via *NuGet Package Manager* no Visual Studio, procurando por "Sharp7"e instalando o pacote no projeto *Standard*.

3.3.2 Conexão com o PLC

Depois, criou-se uma instância da classe S7Client para estabelecer uma conexão com o PLC, utilizando o endereço Internet Protocol (IP) do PLC, o *rack* e o *slot* para configurar a conexão conforme o código mostrado na Listagem 1. Para cada módulo foi estabelecida uma conexão para o PLC correspondente da autoclave.

Listagem 1 – Criação de conexão Sharp7

```
1     S7Client client = new S7Client();
2     int result = client.ConnectTo("192.168.0.1", 0, 1); // IP, rack, slot
3     if (result == 0)
4     {
5         Console.WriteLine("Connected to PLC.");
6     }
7     else
8     {
9         Console.WriteLine("Connection failed.");
10    }
```

Fonte: Autoria própria (2024).

3.3.3 Leitura de dados

Para realizar uma leitura após a criação da conexão, conforme a Listagem 2,utiliza-se a função *BlockRead*, como segue:

- DB : Bloco de dados que será lido no PLC;
- *Position* : Posição para iniciar a leitura em *Bytes*;
- *Size* : Quantidade de *bytes* que será lido;
- *Block* : *Array* em *bytes* que receberá os dados da leitura;
- *Result* : Código de erro (Caso seja "0"não há nenhum erro na leitura).

Listagem 2 – Leitura de dados no Sharp7

```
1
2  byte[] BlockRead(int DB, int Position, int Size)
3      {
4          byte[] block = new byte[Size];
5          int result = S7Client.DBRead(DB, Position, Size, block);
6          if (result != 0)
7          {
8              this.stateConnection = StateConnection.Diconnected;
9
10             try
11             {
12                 StreamWriter doc = new StreamWriter("logError.txt", true);
13                 doc.WriteLine(DateTime.Now.ToString() + " - " +
14                             result + " - " + S7Client.ErrorText(result));
15                 doc.Close();
16             }
17             catch (Exception)
18             {
19
20             }
21         }
22         return block;
23     }
```

Fonte: Autoria própria (2024).

3.3.3.1 Tipos de dados

Para uma melhor manipulação das variáveis foram criados os tipos de dados para poder correlacionar com os dados do PLC:

- BOOL : Dados booleanos (verdadeiro/falso);
- BYTE: Valores de 8 bits não assinados.
- INT: Valores de 16 bits com sinal;
- REAL: Valores de ponto flutuante de 32 bits (IEEE 754);
- String: Cadeias de caracteres.

Os tipos de dados aqui definidos são estruturas primordiais nas quais qualquer informação contida no PLC poderão ser definidas por eles ou por um conjunto deles.

3.3.3.2 Estruturas de dados

Alguns elementos precisaram ser definidos por um conjunto de variáveis como a leitura do sensor de temperatura. Para sua representação gráfica não basta a leitura atual do sensor

mas também sua escala de mínimo e máximo. Para esse caso o elemento assim chamado de "Escala_Real"é composto por três valores do tipo "REAL"(Mínimo, Máximo, Valor). Como cada valor em REAL equivale a 32 Bits ou 4 Bytes a variável "Escala_Real"equivale a 12 Bytes.

Essa estruturação dá definição aos elementos como válvulas, motores, saída digitais, entradas digitais, entre outros.

3.3.3.3 Bloco de dados

Nessa etapa são agrupados em blocos as estruturas de dados de mesmos tipos, os quais são correlacionados com os bloco de dados lidos e escritos no PLC.

Neste caso, na modelagem apresentada no Quadro 7, faz-se referência a um bloco de dados do tipo "Entradas_Analogicas". Esse bloco corresponde a composição de todas as entradas analógicas, que por sua vez são representadas por estruturas de dados do tipo "Escala_Real". Ao todo são nove entradas analógicas e, como a "Escala_Real"equivale a 12 Bytes, nesse bloco tem-se ao todos 108 Bytes, ou seja, exatamente o tamanho do Bloco de Dados do PLC para entradas analógicas.

3.3.3.4 Sincronismo com o PLC

Como as variáveis coexistem no sistema SCADA e no PLC foi definido um tempo de atualização de um segundo, sendo a rotina de execução da seguinte forma:

1. Leitura dos bloco de dados do PLC em array de Bytes;
2. Particionamento dos blocos em estruturas de dados;
3. Particionamento das estruturas em tipo de dados;
4. Realização das operações no sistema SCADA;
5. Montagem das estruturas e blocos;
6. Escrita dos blocos no PLC.

3.4 Integração com o banco de dados MySQL

O banco de dados foi desenvolvido conforme a modelagem dos mesmos, em concordância com a equipe de TI, analisando as necessidades da aplicação e elaborando um modelo robusto e eficiente. Após a definição das tabelas, colunas, tipos de dados e relacionamentos, o banco foi implementado seguindo o esquema definido, assegurando a integridade e consistência dos dados.

A integração com MySQL no Visual Studio foi realizada utilizando o MySQL Connector/NET, um *driver* para .NET que permitiu conectar o aplicativo a um banco de dados MySQL. Primeiro, foi realizado o download e instalado o MySQL Connector/NET do site oficial da plataforma. No *Solution Explorer*, foi clicado com o botão direito no nome do projeto e selecionado "Gerenciar Pacotes NuGet". Na aba "Procurar", foi pesquisado por MySql.Data e instalado o pacote MySql.Data da Oracle.

Foram utilizadas funções em SQL para acessar o banco de dados, desenvolvidas conforme a modelagem. Isso facilitou a integração com a aplicação, melhorou o desempenho e garantiu a consistência e integridade das operações. Todas as interações do banco de dados foram inseridas no projeto *DataBase*.

3.4.1 Usuário

Para armazenar os dados de usuários foi criada uma tabela seguindo as definições da modelagem no Quadro 1, de tal maneira que as seguintes funções de manipulação de dados foram criadas:

- Insert_Users - Para inserir um novo usuário (Na Listagem 3 tem-se a instrução em SQL para essa função);
- Fill_Users - Busca todas as informações de todos usuário exceto as senhas;
- Delete_User - Altera o campo *Activate* do usuário para falso;
- Update_User - Faz uma atualização do campos do usuário;
- Load_User - Usado para fazer login no sistema validando o login e senha.

Listagem 3 – Código em SQL para criação de usuário

```
1  INSERT INTO Users
2      (Name, Description, DateTime_Create, Id_Attribute, Login, Password,
3      Id_User_Creator, Activate)
4  VALUES
5      (@Name, @Description, @DateTime_Create, @Id_Attribute, @Login,
6      @Password, @Id_User_Create, @Activate)
```

Fonte: Autoria própria (2024).

3.4.2 Produto

Para manipular os dados da tabelas de produtos tem-se as seguintes funções:

- Insert_Product - Para inserir um novo produto;

- Fill_Product - Busca todas as informações de todos os produtos;
- Delete_Product - Altera o campo *Activate* do produto para falso;
- Update_Product - Faz uma atualização do campos do produto;
- Load_Product - Usado para carregar as informações do produto;
- Recover_Product - Para fazer restauração de um produto ativo.

Para as receitas utiliza-se as funções análogas criadas para os produtos.

De maneira similar, para a produção tem-se apenas três funções: i) *Insert_Producction*, para inserir uma nova produção; ii) *Get_Production*, para buscar dados de uma produção; iii) *Uptade_Production* para inserir os dados finais de uma produção.

3.4.3 Alarmes

No banco de dados há uma tabela com os códigos de alarmes e suas definições para serem consultadas pelo programa. Na ocorrência de um desses alarmes ser ativado, o mesmo será inserido em uma outra tabela, no qual são registrados os alarmes ativos, ficando salvo o horário, a autoclave e o código do alarme que ocorreu.

3.5 Execução e operação

3.5.1 Alarmes

Para que o alarme saia da tabela de ativos, dois eventos devem ocorrer, a condição que gera o alarme deve ser resolvida e o alarme deve ser resetado pelo operador no supervisório, não importando a ordem dos eventos. Este será movido da tabela de alarmes ativos para uma tabela de histórico, para eventuais consultas futuras.

3.5.2 Interface do usuário

3.5.2.1 Layout

No *layout*, foram definidas as divisões da visualização da aplicação, considerando a visualização para o usuário final. Para cada um dos elementos da divisão em tela foram estipulados posição e tamanho.

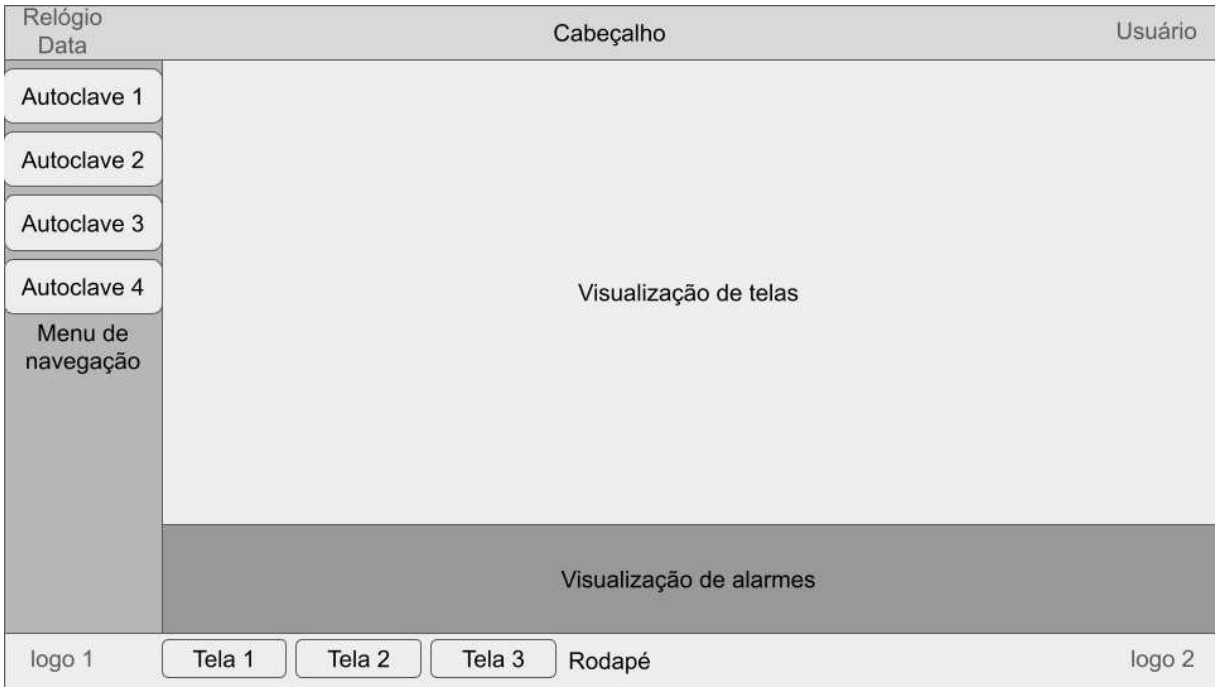
Para definição de *layout* a escolha se fez pensando no uso diário do supervisório durante a produção. O objetivo é fornecer uma interface amigável e de fácil utilização, com uma

disposição lógica de informações e controles, além de elementos visuais claros e compreensíveis. De maneira geral, um bom *layout* para supervisório visa melhorar a eficiência operacional, aumentar a segurança, reduzir o tempo de resposta a eventos e facilitar a tomada de decisões dos operadores.

No *layout*, foram determinados quais elementos da aplicação permaneceriam visíveis constantemente durante o uso, como a barra superior e o menu de navegação. Além disso, foram especificados quais elementos poderiam ser manipulados para se tornarem visíveis, como no caso das telas, podendo ser uma de configuração ou de acompanhamento de produção.

Na Figura 2 pode-se observar o layout esboçado para o inicio da criação do supervisório, sendo a visualização fixa do rodapé, cabeçalho, menu de navegação e visualização de alarmes, mantendo-se as visualizações de telas no centro trocáveis pelo menu de navegação ou rodapé.

Figura 2 – Idealização do Layout



Fonte: Autoria própria (2019).

3.5.2.2 Telas

As telas foram definidas pelas diferentes visualizações acessíveis aos usuários, sendo elas intercambiáveis através do menu de navegação. Estas foram criadas conforme a necessidade da aplicação.

Apenas uma tela ficou visível por vez, e o tamanho e posição delas foram padronizados, seguindo o que foi definido no *layout*.

3.5.2.3 *Pop-Ups*

Diferente do conceito de telas, os *pop-ups*, quando acessados, sobrepõem os demais elementos visuais, com o objetivo de fornecer uma janela para ajustes ou notificações rápidas sem interromper a visibilidade da tela atual.

Tais *pop-ups* foram desenvolvidos para terem diversos tamanhos e movidos para qualquer posição em tela. Por surgirem sobrepondo os elementos do *layout*, não comprometem o mesmo.

3.5.2.4 Menu de navegação, Cabeçalho e Painel de Alarmes

O Menu de navegação é um elemento sempre visível no layout, tem como objetivo a navegação pelo usuário nas diversas telas listadas.

Já o cabeçalho apresenta visualização permanente no layout, e traz informações simples como horário atual, usuário logado e nome da tela atual.

O Painel de Alarmes também é visível a todo momento. Apresenta um breve resumo dos últimos alarmes ativos, a fim de agilizar na tomada de decisão do operador.

4 RESULTADOS

Neste capitulo serão apresentados os resultados obtidos com o desenvolvimento do projeto.

4.1 Estruturação do Projeto

No IDE do Visual Studio todo o conjunto da aplicação é nomeado como uma solução, e dentro dele são criados os projetos. Nesta etapa é definida qual linguagem de programação será utilizada e para qual sua finalidade, como aplicação WEB, serviços do sistema e, para este caso, uma aplicação para o ambiente *Desktop*.

Para cada divisão foi criado um projeto próprio dentro da solução. Os três tipos utilizados foram:

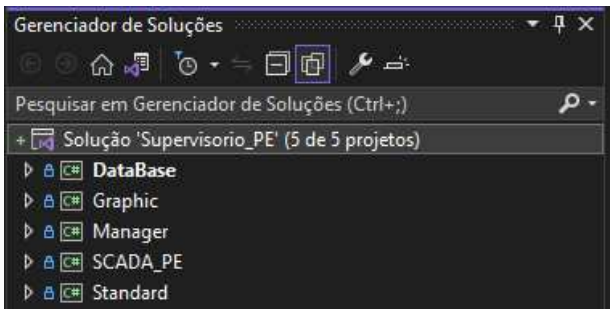
- Bibliotecas gráfica - Podem ser instanciadas por outras aplicações contendo recursos gráficos, como telas e botões;
- Bibliotecas de classes - Podem ser instanciadas por outras aplicações contendo funções e classes que modelam qualquer tipo de objeto;
- Projetos Executáveis - Projeto que pode ser executado pelo usuário final.

Na lista a seguir são apresentados os tipos de projetos que foram definidos para as divisões:

- *Graphic* - Biblioteca gráfica;
- *DataBase* - Biblioteca de classes;
- *Manager* - Biblioteca de classes;
- Scada_PE - Projeto Executável;
- *Stardard* - Biblioteca de classes.

Na Figura 3 pode-se observar a visualização dos projetos no gerenciador de soluções do Visual Studio.

Figura 3 – Estrutura do projeto



Fonte: Autoria própria (2019).

4.2 Interface do usuário

A interface do usuário é composta por uma tela principal e por *pop-ups* e telas que derivam da tela principal.

4.2.1 Layout

Na Figura 4, pode-se observar o *layout* final para o sistema, o qual foi planejado para proporcionar uma experiencia agradável ao usuário. Ele foi projetado com a contenção dos elementos de tela. Isso significa que a interface nunca abrirá mais do que deveria, evitando sobrecargas visuais e distrações desnecessárias.

Este layout utiliza uma paleta de cores específica que inclui o cinza, o azul, o vermelho e o verde. A cor cinza foi escolhida, por ter uma tonalidade mais neutra, não sendo cansativa aos olhos. A cor verde remete a aspectos positivos e de confirmações de ações que foram bem sucedidas. O vermelho foi trazido para indicar itens importantes e emitir alertas, garantindo que informações importantes sejam percebidas rapidamente. Note que os elementos do layout são sempre visíveis, para que o usuário tenha um acesso continuo e rápido a todas as informações e funcionalidades necessárias.

Figura 4 – Layout Final

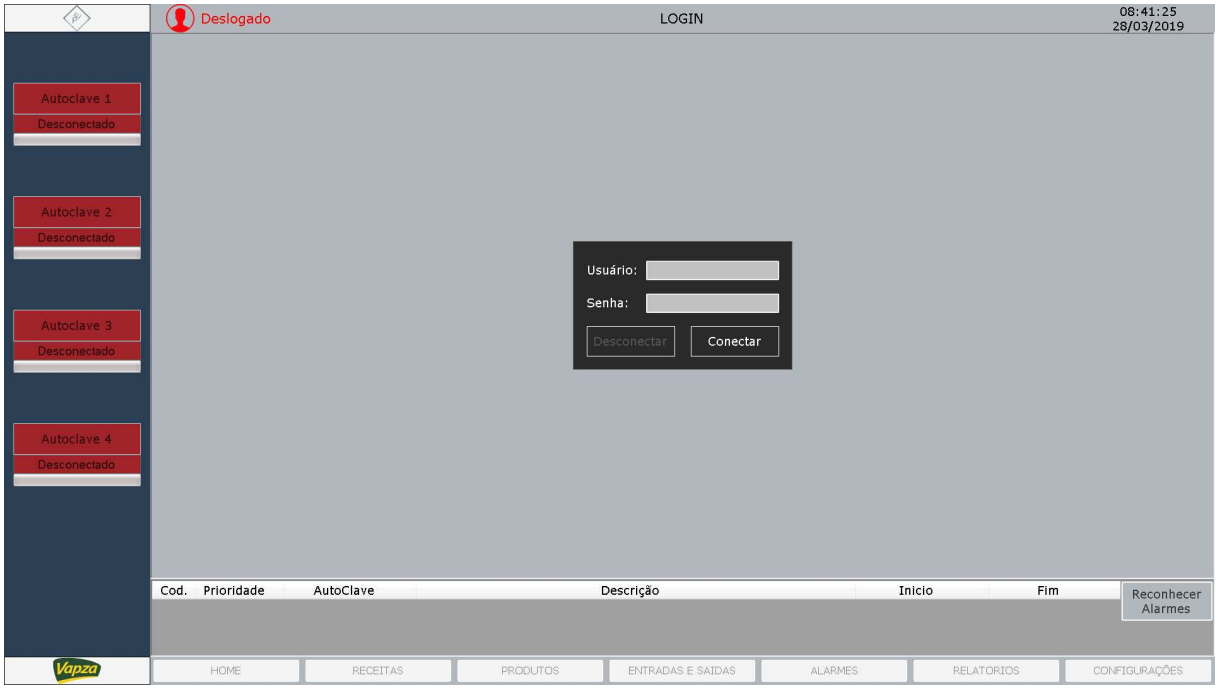
Fonte: Autoria própria (2019).

4.2.2 Inicialização

A tela de inicialização do sistema é de suma importância para garantir a segurança e a integridade dos dados da empresa. Quando o sistema é aberto é esta que o usuário visualiza. A tela atua como uma barreira de proteção, permitindo apenas usuários cadastrados e ativos a terem acesso ao sistema.

Para acessar o sistema, cada usuário deve fornecer suas informações, o que é composto por um "Usuário"e uma "Senha". Quando o usuário está cadastrado, ativo e insere a senha correta, ele é direcionado para a "Tela Principal (*Home*)". Na Figura 5, pode-se observar a tela de inicialização final do sistema.

Figura 5 – Tela de Inicialização



Fonte: Autoria própria (2019).

4.2.3 Tela principal

A tela principal do sistema proporciona ao usuário uma visão completa das possibilidades de ações disponíveis. Nesta é possível ver os menus de navegação e escolher a ação a ser executada, facilitando a interação e a eficiência no uso do sistema. Foi projetada com o objetivo ser a mais utilizada pelos usuários, pois centraliza todas as funcionalidades essenciais.

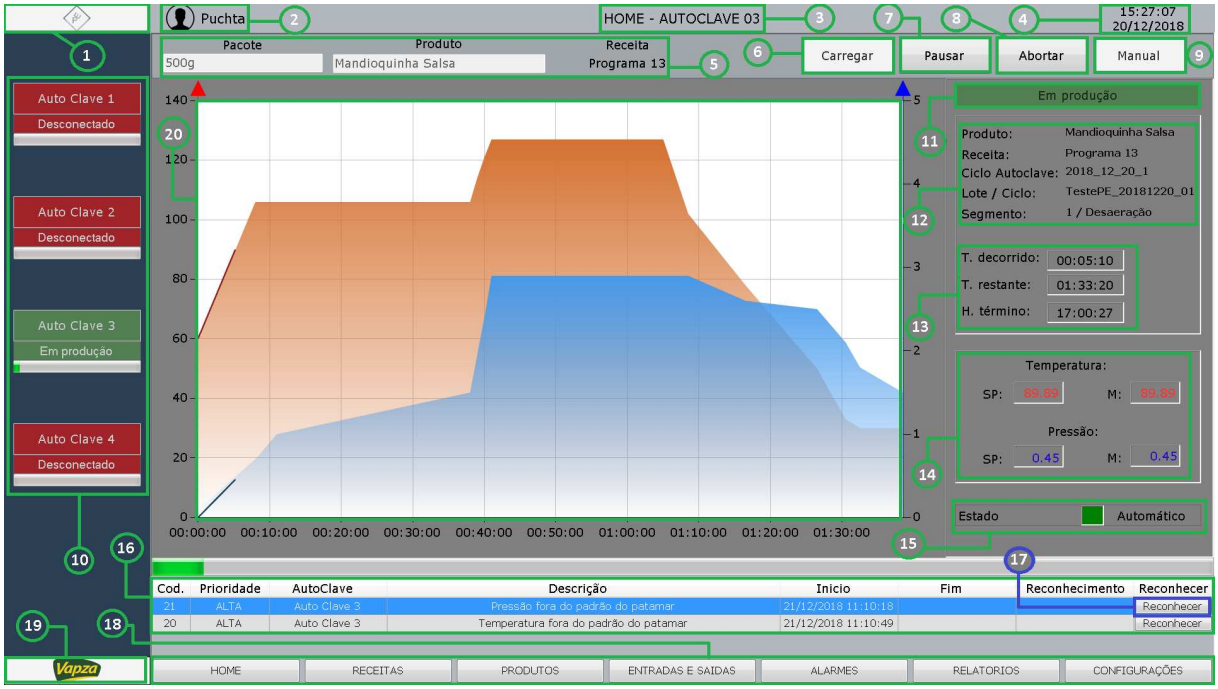
A tela principal possui dois menus de navegação principais. O primeiro permite ao usuário escolher a autoclave que realizara a receita. Já o segundo é utilizado para iniciar e visualizar o andamento da receita na autoclave selecionada. Essa divisão permite que o usuário tenha uma experiencia mais organizada e eficiente.

Uma vez que os produtos e receitas já estejam cadastradas no sistema, o usuário pode dar inicio a produção direto da tela "Home". Isso permite um melhor gerenciamento da produção. Na Figura 6 pode-se observar a Tela principal, com as seguintes divisões (numeradas):

- 1. Logo da empresa desenvolvedora do sistema;
- 2. Nome do usuário logado no sistema;
- 3. Nome da tela que esta em visualização;
- 4. Data e horário atual;
- 5. Seleção do pacote, produto e receita para a produção;

- 6. Comando para carregar a receita que da inicio a produção;
- 7. Comando para pausar uma produção em operação;
- 8. Comando para abortar uma produção em operação;
- 9. Comando para mudar entre manual e automático;
- 10. Menu de navegação das autoclaves;
- 11. Status da autoclave;
- 12. Resumo da produção;
- 13. Informação sobre o tempo da produção;
- 14. Valores de *setpoint* (SP) e valores medidos (M) de Temperatura e Pressão;
- 15. Estado de manual ou automático da autoclave;
- 16. Janela de visualização de alarme;
- 17. Botão para reconhecer alarme;
- 18. Menu de navegação;
- 19. Logo da empresa do ramo alimentício;
- 20. Gráfico com as sombras dos valores desejados, com os valores das penas dos valores medidos, escrevendo sobre a sombra (Eixo x: tempo, Eixo y: Temperatura em vermelho e pressão em azul).

Figura 6 – Tela principal *(Home)*

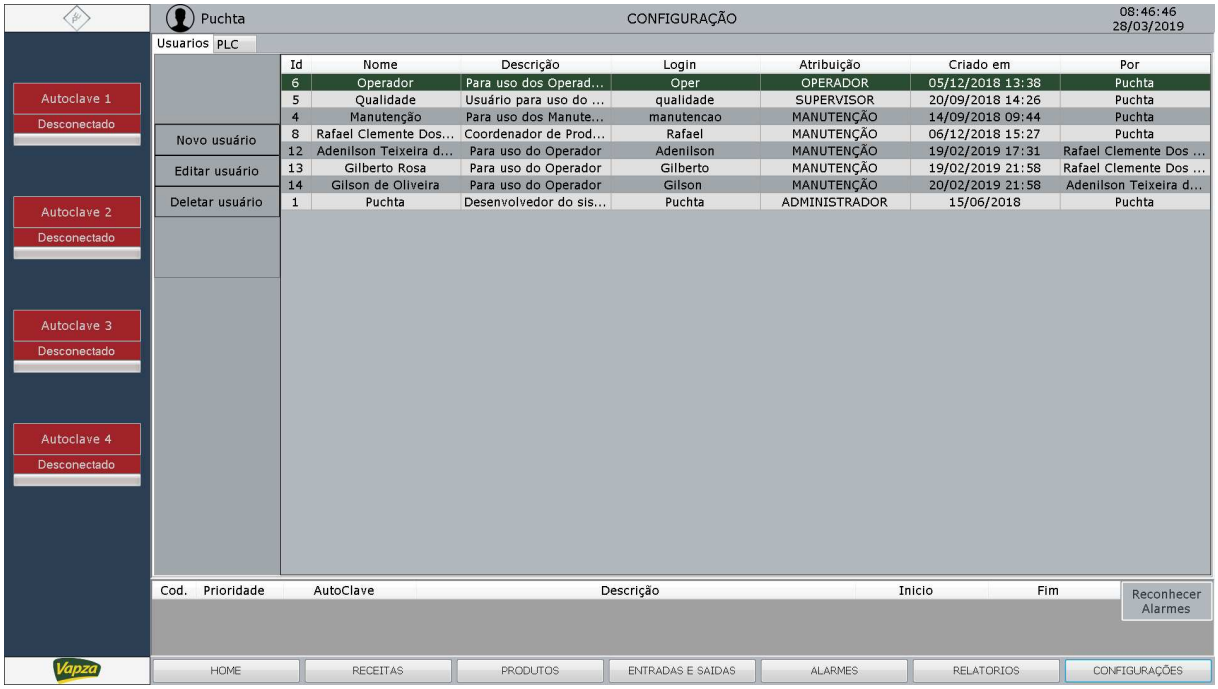


Fonte: Autoria própria (2019).

4.2.4 Usuário

A tela de usuário mostra opções como a criação de um novo usuário, editar um existente ou até mesmo deletar um usuário. Observe que, na realidade, um usuário nunca é deletado em definitivo do sistema, mas apenas é retirado do campo de busca (quando este não esta mais ativo na empresa). Contudo, são mantidos os dados em *standby* para uma possível consulta futura. Na Figura 7 pode-se observar a tela principal final, a qual mostra as opções disponíveis.

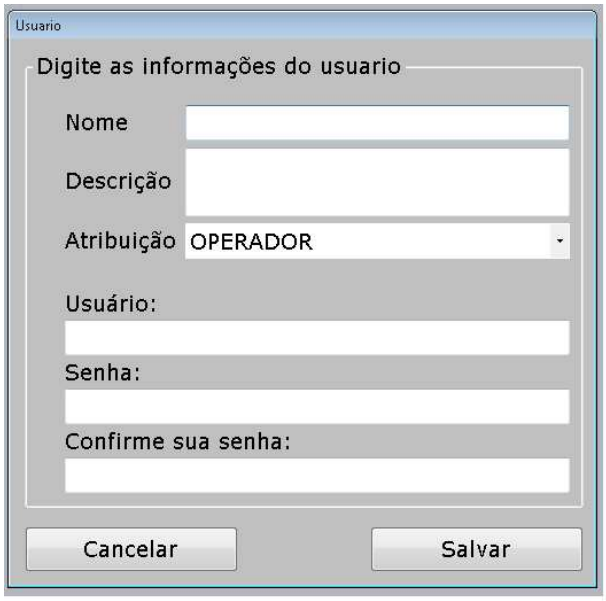
Figura 7 – Tela do usuário



Fonte: Autoria própria (2019).

Os *pop-ups* oferecem janelas mais objetivas, derivadas da tela principal, como pode ser visualizado na Figura 8, a qual mostra o *pop-up* para criar um novo usuário, sendo possível inserir o nome, uma descrição (como a área ou o turno laboral), atribuição e os campos nos quais é cadastrado um usuário e senha de acesso ao sistema.

Figura 8 – Cadastro de novo usuário

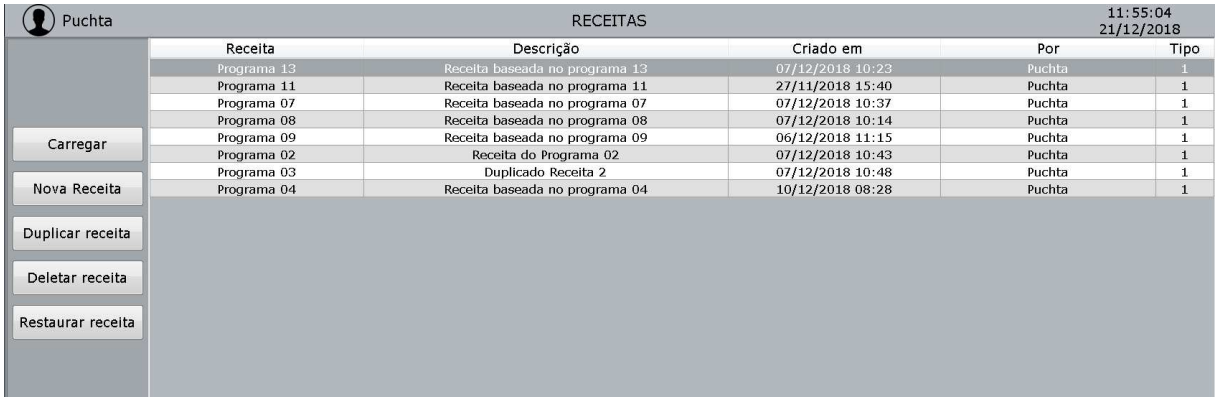


Fonte: Autoria própria (2019).

4.2.5 Receita

A tela de receitas apresenta todas as receitas criadas e ativas. Nela o usuário tem opções como carregar, nova receita, duplicar, deletar e restaurar (Figura 9). Neste caso, assim como no caso do usuário, não é possível deletar uma receita definitivamente, já que é necessário manter no banco de dados todas as receitas já criadas para uma possível consulta futura. De fato, quando o usuário deleta uma receita ele está desativando-a e, se necessário, esta pode ser restaurada.

Figura 9 – Tela de receitas



Receita	Descrição	Criado em	Por	Tipo
Programa 13	Receita baseada no programa 13	07/12/2018 10:23	Puchta	1
Programa 11	Receita baseada no programa 11	27/11/2018 15:40	Puchta	1
Programa 07	Receita baseada no programa 07	07/12/2018 10:37	Puchta	1
Programa 08	Receita baseada no programa 08	07/12/2018 10:14	Puchta	1
Programa 09	Receita baseada no programa 09	06/12/2018 11:15	Puchta	1
Programa 02	Receita do Programa 02	07/12/2018 10:43	Puchta	1
Programa 03	Duplicado Receita 2	07/12/2018 10:48	Puchta	1
Programa 04	Receita baseada no programa 04	10/12/2018 08:28	Puchta	1

Fonte: Autoria própria (2019).

4.2.5.1 Carregar

Quando o usuário seleciona uma receita ativa e clica em carregar, ele é direcionado a tela da Figura 10, que apresenta todo o passo a passo daquela receita, sendo possível visualizar o numero de procedimentos, o segmento, tempo, temperatura e pressão de cada etapa, além de mostrar o tempo total do processo.

Figura 10 – Tela de carregar receita

N°	Segmento	Tempo	Temperatura	Pressão
0	Saída	00:00:00	60	0.1
1	Desaeração	00:08:00	106	0.7
2	Aquecimento	00:03:00	106	1
3	Aquecimento	00:27:00	106	1.5
4	Aquecimento	00:01:00	114	1.95
5	Aquecimento	00:01:00	121	2.4
6	Aquecimento	00:01:00	127	2.9
7	Esterilização	00:24:00	127	2.9
8	Pré-resfriamento	00:03:30	102	2.9
9	Resfriamento	00:08:00	78	2.6
10	Resfriamento	00:10:00	50	2.5
11	Resfriamento	00:04:00	33	2.1
12	Resfriamento	00:02:00	30	1.8
13	Resfriamento	00:06:00	30	1.5

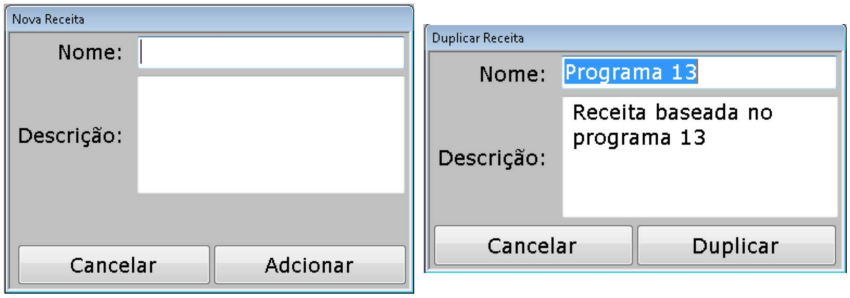
Fonte: Autoria própria (2019).

4.2.5.2 Criar ou duplicar uma receita

Na Figura 11(a) pode-se visualizar o *pop-up* para criar uma nova receita. Nela o usuário dá um nome a receita e uma descrição, quando clicado em adicionar. Em seguida, é direcionado para a tela de criação de receitas (Figura 12), na qual é possível alterar o tipo dos segmentos, o tempo (hora, minuto, segundo), a temperatura (℃) e a pressão (bar).

Já a Figura 11(b), mostra o *pop-up* para duplicar uma receita. Tal função é utilizada quando o usuário deseja alterar alguma variável em uma receita já existente e, com isto não é necessário recriar toda a receita novamente, mas apenas alterar o que é requisitado.

Figura 11 – Criar ou duplicar uma receita



(a) **(b)**

Fonte: Autoria própria (2019).

Figura 12 – Nova receita

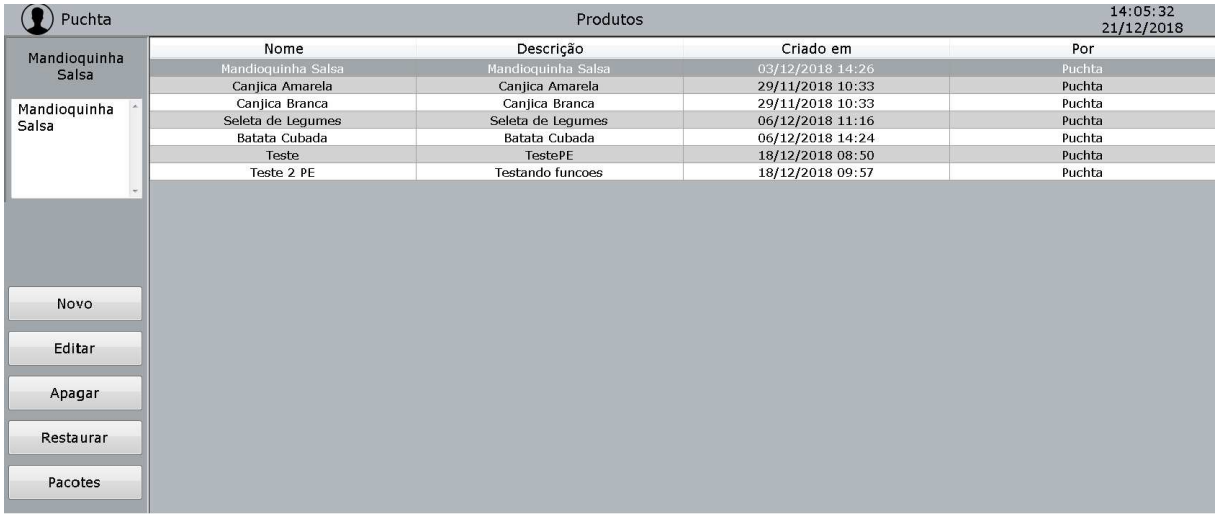
N°	Segmento	Tempo	Temperatura	Pressão
0	Saída	00:00:00	60	0.1
1	Desaeração	00:08:00	106	0.7
2	Aquecimento	00:03:00	106	1
3	Aquecimento	00:27:00	106	1.5
4	Aquecimento	00:01:00	114	1.95
5	Aquecimento	00:01:00	121	2.4
6	Aquecimento	00:01:00	127	2.9
7	Esterilização	00:24:00	127	2.9
8	Pré-resfriamento	00:03:30	102	2.9
9	Resfriamento	00:08:00	78	2.6
10	Resfriamento	00:10:00	50	2.5
11	Resfriamento	00:04:00	33	2.1
12	Resfriamento	00:02:00	30	1.8
13	Resfriamento	00:06:00	30	1.5

Fonte: Autoria própria (2019).

4.2.6 Produtos

A tela de produtos apresenta todos os produtos criados e ativos no momento, assim como a descrição de cada produto, quando e por quem foi criado. Nela o usuário tem opções como criar um novo produto, editar algum já existente, apagar, restaurar e uma opções de pacotes. Na Figura 13 pode-se observar como ficou o resultado da referida tela.

Figura 13 – Tela de produtos



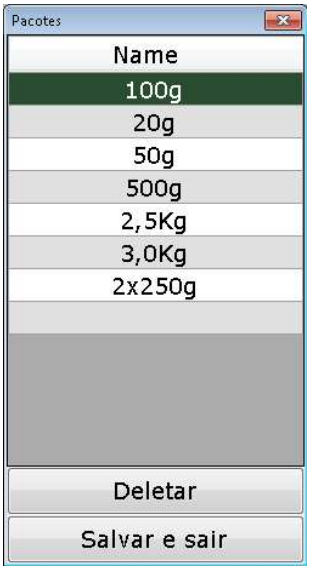
Fonte: Autoria própria (2019).

4.2.6.1 Pacotes

O *pop-up* de pacotes foi implementado no projeto após ser identificada uma necessidade após a conclusão do projeto. Isto surgiu devido a variação nos pesos das embalagens (pacotes) da empresa, pois esta utiliza diferentes pesos para cada produto, o que pode resultar em alterações na receita. Essa função se refere a uma tabela chamada "Gestão de pacotes", que relaciona os diferentes pacotes com as respectivas receitas.

Quando o usuário seleciona a opção "Pacotes"na tela de produtos é redirecionado para o *pop-up* da Figura 14, em que é possível criar, editar ou apagar um pacote.

Figura 14 – Pop-up de pacotes



Fonte: Autoria própria (2019).

4.2.6.2 Criar ou editar um produto

Na Figura 15, pode-se observar o *pop-up* visualizado pelo usuário quando ele clica em "Novo"ou em "Editar"na Tela de produtos. Ao clicar em "Novo", o *pop-up* abre completamente em branco. No entanto, ao selecionar um produto na Tela de produtos e clicar em "Editar", o *pop-up* abre já preenchido, permitindo a edição apenas dos campos necessários. Em ambos os casos o usuário pode habilitar uma receita para determinado pacote.

Figura 15 – Criar ou editar um produto

Fonte: Autoria própria (2019).

Como no usuário e em receitas, não é possível deletar uma produto definitivamente, já que é preciso manter no banco de dados todos os produtos já criados para uma possível consulta futura. Quando o usuário deleta um produto ele está desativando-o e, se necessário, este produto pode ser restaurado. Quando o usuário clica em "Restaurar"o sistema abre um *pop-up* (Figura 16), onde mostra todos os produtos deletados, permitindo a recuperação da informação.

Figura 16 – Restaurar um produto

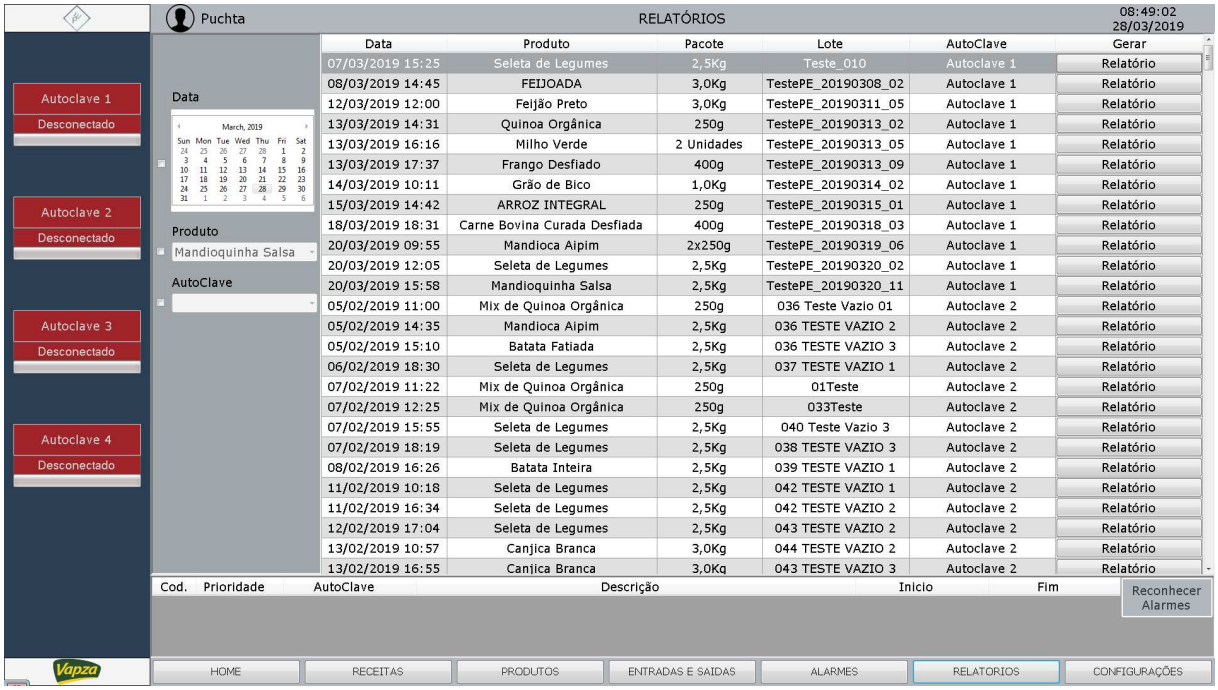
Fonte: Autoria própria (2019).

4.2.6.3 Registro de produção

Na Figura 15, pode-se observar a tela de "Registro de produção". Após o final de uma produção, suas informações ficam salvas em um banco de dados, no qual o usuário pode consultar quando necessário. Clicando em "Relatório", os dados da produção requerida são solicitados ao banco de dados e o relatório é gerado em formato PDF.

O sistema permite que os relatórios sejam filtrados por data, produto e autoclave, facilitando a busca do usuário de forma rápida e eficiente.

Figura 17 – Relatórios de produção



Fonte: Autoria própria (2019).

5 CONCLUSÃO

O presente projeto abrangeu o desenvolvimento e a integração de diversos componentes essenciais para a criação de um sistema SCADA eficiente e seguro. O estudo de caso envolveu a automação de uma empresa alimentícia.

Através da biblioteca modular *Stardard*, foi possível garantir a consistência e a padronização na definição de modelos, facilitando a reutilização de código e a manutenção do sistema. O projeto *DataBase* assegurou a manipulação eficaz dos dados, enquanto o *Manager* coordenou a lógica operacional, e o projeto *Graphic* proporcionou uma interface de usuário intuitiva e agradável.

A utilização de bibliotecas *open source* destacou-se por proporcionar uma solução custo-efetiva, transparente, flexível e com um suporte comunitário robusto. Além disso, a modelagem precisa de dados assegurou a integridade e a rastreabilidade das informações, facilitando a gestão dos diferentes elementos do sistema.

A integração da biblioteca Sharp7 permitiu uma comunicação robusta com PLCs Siemens S7, essencial para o controle dos processos industriais. Tal procedimento facilitou a leitura e a escrita de dados, mantendo a sincronização entre o sistema SCADA e os dispositivos de controle.

Ao longo do desenvolvimento, a modelagem de dados foi fundamental para assegurar a integridade e a rastreabilidade das informações, além de proporcionar uma estrutura clara e organizada para a aplicação. A definição precisa de modelos para usuários, receitas, produtos, produção, PLCs e módulos permitiu uma gestão eficaz e padronizada dos diferentes elementos do sistema.

Como trabalhos futuros, sugere-se a implementação das diretivas e soluções desenvolvidas em outros processos industriais e em manufaturas de diferentes naturezas. Isso permitirá expandir o alcance e a aplicabilidade do sistema SCADA, promovendo a automação eficiente e segura em um espectro mais amplo de indústrias.

REFERÊNCIAS

BARTHOLOMEW, D. Sql vs. nosql. **Linux Journal**, Belltown Media Houston, TX, v. 2010, n. 195, p. 4, 2010.

DANEELS, A.; SALTER, W. What is scada? 1999. Disponível em: https://cds.cern.ch/record/532624/files/mc1i01.pdf. Acesso em: 01 out. 2023.

ERICKSON, K. T. Programmable logic controllers. **IEEE potentials**, IEEE, v. 15, n. 1, p. 14–17, 1996.

FARNHAM, B.; BARILLERE, R. **MIGRATION FROM OPC-DA TO OPC-UA**. [*S.l.*], 2011.

FOUNDATION, O. **What is OPC?** 2023. OPC Foundation. Disponível em: https://opcfoundation.org/about/what-is-opc/. Acesso em: 29 dez. 2023.

FUGGETTA, A. Open source software—-an evaluation. **Journal of Systems and software**, Elsevier, v. 66, n. 1, p. 77–90, 2003.

GONZÁLEZ, I. *et al.* A literature survey on open platform communications (opc) applied to advanced industrial environments. **Electronics**, MDPI, v. 8, n. 5, p. 510, 2019.

HAT, R. **O que é IDE? - Ambiente de Desenvolvimento Integrado.** s.d. RedHat. Disponível em: https://www.redhat.com/pt-br/topics/middleware/what-is-ide. Acesso em: 04 fev. 2024.

LEE, T. G. **O que é o Visual Studio?** s.d. Microsoft. Disponível em: https://learn.microsoft.com/pt-br/visualstudio/get-started/visual-studio-ide?view=vs-2022. Acesso em: 04 fev. 2024.

MANOVICH, L. Banco de dados. **Revista ECO-Pós**, v. 18, n. 1, p. 7–26, 2015.

OFOEDA, J.; BOATENG, R.; EFFAH, J. Application programming interface (api) research: A review of the past to inform the future. **International Journal of Enterprise Information Systems (IJEIS)**, IGI Global, v. 15, n. 3, p. 76–95, 2019.

SCHWAB, K. **A quarta revolução industrial**. [*S.l.*]: Edipro, 2019.

SEHR, M. A. *et al.* Programmable logic controllers in the context of industry 4.0. **IEEE Transactions on Industrial Informatics**, IEEE, v. 17, n. 5, p. 3523–3533, 2020.

SHARMA, V.; DAVE, M. Sql and nosql databases. **International Journal of Advanced Research in Computer Science and Software Engineering**, v. 2, n. 8, 2012.

SNAP7HOMEPAGE. s.d. Snap7 Homepage. Disponível em: https://snap7.sourceforge.net/. Acesso em: 04 fev. 2024.

STALLMAN, R. **Why Open Source Misses the Point of Free Software**. 2023. GNU Project - Free Software Foundation. Disponível em: https://opcfoundation.org/about/what-is-opc/. Acesso em: 29 jan. 2024.

GLOSSÁRIO

BOOL Pode ser verdadeiro ou falso. 25–27

DATETIME Representa um momento no tempo, geralmente expresso como uma data e hora do dia. 25–28

INT O tipo inteiro é usado para representar números inteiros, positivos ou negativos, sem parte decimal. 25–28

LIST Armazena uma coleção ordenada de elementos. 26

REAL Representa números racionais com parte decimal. 27

STRING Representa sequências de caracteres. 25–28