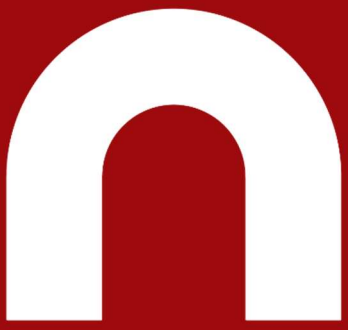


isec
Engenharia

MESTRADO EM ENGENHARIA ELETROTÉCNICA

Desenvolvimento de aplicações de automação usando Ignition

Autor

Tomás Verdade Ribeiro

Orientador

João Paulo Morais Ferreira

Coimbra, março 2023

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

isec Engenharia

DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA

Desenvolvimento de aplicações de automação usando Ignition

Relatório de Estágio de Natureza Profissional para a obtenção do grau de Mestre em Engenharia Eletrotécnica

Especialização em Automação e Comunicações em Sistemas Industriais

Autor

Tomás Verdade Ribeiro

Orientador

João Paulo Morais Ferreira

Supervisor na empresa SA – Soluções em Automação, S.A.

Pedro Miguel Figueiredo Pedrosa

INSTITUTO POLITÉCNICO DE COIMBRA

INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

Coimbra, março 2023

AGRADECIMENTOS

Este relatório de estágio resulta da conjugação de diversos fatores ao longo de um vasto período, pelo que contou com a colaboração de várias pessoas às quais não posso deixar de agradecer.

Em primeiro lugar quero agradecer a todos os colaboradores da SA – Soluções em Automação, S.A., pela forma como me acolheram e me fizeram sentir em casa, nomeadamente ao Engenheiro Pedro Pedrosa e ao Engenheiro André Costa por todo o apoio e paciência ao longo do período de estágio.

Quero também agradecer ao meu orientador, Professor Doutor João Paulo Morais Ferreira, por toda a disponibilidade demonstrada e por todo o auxílio prestado na elaboração deste relatório.

Como não poderia faltar, um agradecimento especial a todos os meus familiares por todo o apoio nas mais diversas ocasiões. Em especial aos meus pais e avós que, além de contribuírem para a minha educação, contribuíram de forma especial no fator motivacional.

Por último, quero deixar o meu obrigado aos amigos de sempre e aos amigos de Coimbra. Por todos os cafés, por todos os trabalhos de grupo e por todos os momentos partilhados que me motivaram e apoiaram.

RESUMO

O presente documento tem como finalidade retratar todo o envolvimento do estagiário nas atividades e equipas da SA – Soluções em Automação, mais especificamente no departamento de programação, no qual foi inserido.

No decorrer do estágio, foi dada a oportunidade de participar ativamente no desenvolvimento e otimização de vários projetos inovadores que a empresa tem em curso. Contudo, o principal foco de trabalho foi na elaboração de uma aplicação para a simulação e controlo de equipamentos de palco usado em teatros e óperas (varas, palcos, elevadores, cortinas, sistemas de iluminação, plataformas, etc..).

Numa fase inicial do estágio, a aplicação estava a ser desenvolvida no software *Unity* da *Unity Technologies*. A forte componente de simulação gráfica deste software, recorrendo à linguagem C#, tinha bastante interesse para o resultado pretendido. Porém, foram encontrados alguns obstáculos em diferentes componentes e foi necessário repensar o rumo a dar ao projeto. Após a tomada de decisão de não continuar a desenvolver a aplicação no *Unity* e apenas usar este como simulador, optou-se por desenvolver a aplicação num software com uma linguagem de programação mais adaptada à solução pretendida, surgindo assim a opção do *Ignition*, da *Inductive Automation*.

Antes de recomeçar a aplicação, foi realizada uma formação online, com a duração de uma semana, na *Inductive University* de modo a adquirir um maior conhecimento sobre o software e facilitar assim a transição. O software utilizado para o desenvolvimento da aplicação foi um software de SCADA com inúmeras vantagens não só pela facilidade de programação em *Python*, mas também na comunicação com bases de dados, com autómatos e com o simulador.

Ao longo do período do estágio foi possível captar novos conhecimentos, assim como aprofundar conteúdos já lecionados, bem como ganhar uma nova perspetiva do funcionamento interno de uma empresa na área da automação industrial.

Palavras-Chave: Teatros, Simulação, Controlo, *Unity*, SCADA, Base de Dados, *Ignition*.

ABSTRACT

The purpose of the present document is to portray all the involvement of the intern on the activities and teams of SA – Soluções em Automação, more specifically in the programming department in which he was included.

Over the course of the internship, it was given the opportunity to be actively involved in the process of development and optimization of several innovative projects that the company currently has in progress. However, the focus was the development of an application to simulate and control the equipment of a stage that can be used in theatres and operas (fly bars, stages, elevators, curtains, lights, platforms, etc…).

Initially, the application it was being developed using the software Unity from Unity Technologies. This software graphic simulation component, using C# as is main language, was very promising to the intended result. However, some obstacles were encountered in different stages of the process, so it was necessary to take a step back and rethink the strategy in which the application was moving towards. After the decision was made to stop developing the application on Unity and only using it as a Simulator, opted for a software more compatible with programming languages more suited with the intended solution, therefore appearing Ignition form Inductive Automation.

Before the developing process, an online course was taken in the Inductive University to better prepare and to get familiar with the software. The selected software was a SCADA software with numerous vantages, not only by the Pyhton language that's used in it, but also for the multiple tools to communicate with the databases, the PLC's and with the Simulator.

During the internship it was possible to acquire new knowledge, enhance insights already thought and gain a new perspective of the workflow of a company in the field of industrial automation.

Key Words: Theatres, Simulation, Control, Unity, SCADA, Database, Ignition

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE QUADROS

ABREVIATURAS

BD – Base de Dados

HMI - *Human-Machine Interface*

IDE - *Integrated Development Environment*

IIOT – *Industrial Internet of Things*

MES - *Manufacturing Execution System*

MQTT - *Message Queuing Telemetry Transport*

MSSQL – *Microsoft SQL Server*

OEM - *Original Equipment Manufacturer*

OPC – *Open Platform Communications*

OPC UA - *Open Platform Communications Unified Architecture*

PLC - *Programmable Logic Controller*

RDBMS - *Relational Database Management System*

RTU – *Remote Terminal Unit*

SA – Soluções em Automação

SCADA - *Supervisory Control and Data Acquisition*

SQL - *Structured Query Language*

1 - INTRODUÇÃO

O presente relatório enquadra-se no âmbito da unidade curricular de Estágio Curricular para a obtenção do grau de Mestre em Engenharia Eletrotécnica, na área de especialização em Automação e Comunicações em Sistemas Industriais. Deste modo, pretende-se que o aluno, ao longo do seu percurso na empresa, adquira competências de trabalho que não são possíveis de obter em contexto académico, bem como experimentar a aplicação de conceitos teóricos no mercado de trabalho.

1.1 – Motivação

A principal vantagem na escolha da realização de um estágio é o contacto com o mercado de trabalho, que prepara o aluno da melhor forma para o futuro próximo. A empresa escolhida para a realização do mesmo, foi a SA – Soluções em Automação, S.A..

Figura 1 - Logotipo SA Automação

Esta empresa tem como atividade principal a representação e distribuição de produtos, destinados ao mercado de automação industrial, bem como o apoio ao cliente. As marcas com as quais trabalha são de grande prestígio em todo o mundo, e na maioria dos casos, líderes no seu segmento. Todas estas valências apresentadas pela empresa, aliadas ao facto de estar recetiva para receber o aluno como seu estagiário tiveram um grande fator motivacional.

1.2 – Objetivos

Este estágio tem como finalidade formar o estagiário em contexto profissional, através da integração nas atividades da SA, com particular foco nas seguintes áreas:

- Desenvolvimento de aplicações de simulação e controlo baseadas em C# / *Unity3D*;
- Desenvolvimento de uma aplicação em *Ignition*, software utilizado em aplicações SCADA que utiliza bastantes propriedades de SQL.
- Interligação do software SCADA, *Ignition*, com o software de simulação 3D, *Unity3D*.

De modo a cumprir com os objetivos estabelecidos foram adquiridos e aprofundados conhecimentos em várias linguagens de programação (C# e *Python*) e também em gestão de base de dados utilizando o *Microsoft* SQL.

Estes conteúdos obtidos permitiram o desenvolvimento de uma aplicação para a simulação e controlo de equipamentos de palco usado em teatros e óperas (varas, palcos, elevadores, cortinas, etc..).

1.3 – Organização do trabalho

Para a realização deste estágio foi realizado um planeamento que consistiu em três fases, descritas em mais detalhe nos capítulos seguintes, sendo essas fases: a parte de simulação tendo por base o *Unity*, a construção e desenvolvimento da aplicação recorrendo ao Ignition e o envio da informação para os equipamentos de campo recorrendo a PLC's. Para facilitar o processo de ambientação aos softwares, numa fase inicial, o estagiário acompanhou o desenvolvimento de projetos em *Unity* e em MSSQL com elementos da empresa. Para adquirir conhecimentos no software Ignition, uma vez que era uma novidade na empresa, foi realizada uma formação online, no site da *Inductive University*.

O relatório realizado com base no estágio apresenta cinco capítulos, sendo que o primeiro é focado na motivação e contextualização do problema, bem como nos objetivos propostos para o estágio.

No segundo capítulo é feita uma revisão bibliográfica, focada nos conceitos de SCADA, apresentando soluções existentes no mercado mundial.

O terceiro capítulo aborda o software *Ignition*, da *Inductive Automation*, referindo as suas principais características que lhe confere uma forte competitividade no mercado mundial.

Como foco principal, surge o quarto capítulo, onde, de uma forma geral, é apresentada a aplicação desenvolvida para a gestão e controlo de equipamentos de palco. Neste capítulo será exposto o processo de construção da aplicação, assim como as relações e dependências com outros softwares utilizados em todo o processo.

Por último, no quinto capítulo serão apresentadas conclusões ao trabalho apresentado, bem como sugestões de trabalho.

Na tabela 1 podemos observar a forma como ficou dividido tempo útil de estágio.

Tabela 1 – Tempo útil de estágio

Unity	X	X		X					
SQL		X		X	X				
Ignition			X	X	X	X	X	X	X
Relatório									X
	Out	Nov	Dez	Jan	Fev	Mar	Abr	Mai	Jun

2. REVISÃO DA LITERATURA

Historicamente, nos meados das décadas de 70 e 80, com o avanço da tecnologia e desenvolvimento dos primeiros microprocessadores, o computador tornou-se cada vez mais numa peça chave nos diversos setores industriais. A sua utilização teve parte no planeamento de atividades de manutenção, controlo de stock, armazenamento de dados e de informação em histórico. [1]

Com o crescimento da exigência de mercado nos níveis de produtividade, foi necessário um acompanhamento tecnológico dos equipamentos, transitando de equipamentos puramente mecânicos para equipamentos eletromecânicos e eletrónicos que apresentam ferramentas que possibilitam o controlo por computador, chamados de sistemas de supervisão.

Como forma de se destacarem da competição existente no mercado cada vez mais exigente, as empresas procuram otimizar os seus processos produtivos, gerindo os seus recurso e melhorando a performance operacional, de forma a obter os melhores níveis de eficiência possíveis.

Para tal, a inclusão de um sistema de gestão integrado tornou-se a melhor solução, uma vez que permite otimizar os processos, assim como assegurar o cumprimento de normas e legislações.

Após mais de 50 anos em constante evolução, a arquitetura dos sistemas SCADA fixou-se numa estrutura sólida, contendo vários componentes, cada um com a sua função especifica. Alguns sistemas possuem componentes adicionais, mas os componentes básicos são os seguintes: RTU, PLC, Sistema de telemetria, Servidor de aquisição de dados, HMI, Serviço de histórico e Sistema de supervisão (computador central). [2]

- **RTU:** A principal função de um *Remote Terminal Unit* é recolher informação de sensores de campo e converter a informação em dados digitais, para depois serem enviados ao sistema de supervisão. Além disso, um RTU deve ser capaz de receber comandos do sistema de supervisão. Em alguns casos, o RTU pode conter microcontroladores básicos capazes de executar instruções booleanas simples.
- **PLC:** Apresenta semelhanças com o RTU, mas as funções de controlo são muito mais desenvolvidas, podendo controlar um processo localmente e excetuar instruções de logica simples ou complexa. Assim como o RTU, o PLC recolhe informação dos sensores e atuadores e é capaz de trocar dados com o sistema de supervisão. É possível conectar um HMI diretamente ao PLC, permitindo assim controlar localmente um sistema controlado por um PLC.
- **Sistema de telemetria:** Refere-se aos protocolos de comunicação e canais físicos entre estações e sistema de supervisão.

- **Servidor de aquisição de dados:** Baseado numa arquitetura Cliente-Servidor e protocolos industriais, permite aos Clientes aceder a dados armazenados em RTUs ou PLCs.
- **HMI:** o Interface Homem-Máquina é um dispositivo no qual são apresentados os dados recolhidos no RTU e no PLC, e é possível ao operador interagir com o processo. O HMI é o interface gráfico do sistema SCADA. Recolhe informação, faz o seu processamento e apresenta em relatórios, gráficos, *trends*, alarmes, notificações, etc.
- **Serviço de histórico:** Esta funcionalidade do SCADA pode guardar informação numa base de dados com registo de data e hora de eventos *booleanos*, alarmes ou qualquer tipo de informação recolhida pelo sistema. Deste modo o cliente pode consultar estes dados, analisar estatísticas e *trends* através do HMI.
- **Sistema de supervisão:** O principal componente de um sistema SCADA. Geralmente um computador que contem o software instalado e no qual é possível controlar todos os restantes equipamentos, executando algoritmos, enviando comandos, etc.

A implementação de um sistema SCADA pode trazer inúmeros benéficos, dos quais se destacam os seguintes: [3]

- Um sistema com vastos anos de desenvolvimento que conta com muitas iterações e melhorias em todo o seu processo;
- A quantidade de desenvolvimento requerida ao utilizador é limitada, principalmente com os equipamento de engenharia adequados;
- Este sistema apresenta uma estrutura bem estabelecida, que confere robustez à solução e aumenta a sua confiabilidade;
- Permite uma visão global do processo em que encontra inserido;
- Fornece ao seu utilizador várias ferramentas de diagnóstico e de controlo;
- A grande maioria de fornecedores de software SCADA fornece opções de suporte técnico e de manutenção.

Alguns sistemas SCADA com presença no mercado são:

- **AVEVA *Edge*:** É uma HMI customizável e projetada especialmente para aplicações integradas e OEM, nas quais são fundamentais o tamanho compacto e a eficiência em termos de custo. O seu tamanho compacto e o conjunto de recursos possibilitam que utilizadores criem aplicações HMI intuitivas, seguras e com elevado nível de sustentabilidade. [4]
- **SIMATIC WinCC:** Com uma vasta opção de sistemas de visualização e com variadas funcionalidades de alto desempenho para monitorizar processos automatizados. Compatível com múltiplos utilizadores, distribuídos por servidores de redundância, o utilizador beneficiará de um sistema aberto que oferece funcionalidades para os diversos setores e para aplicações SCADA. [5]

- ***Wonderware InTouch*:** Com uma série de ferramentas disponíveis, o *Wonderware InTouch* permite monitorizar e operar qualquer sistema parcialmente ou completamente automatizado, por meio de uma ou mais estações de operação (computadores ou via Web). A monitorização ocorre por meio de telas gráficas de alta qualidade, com indicações dinâmicas do estado dos equipamentos e grandezas analisadas. Possui também ferramentas para realizar *trends*, relatórios, histórico de eventos e de processo, entre outras. [6]
- ***Wonderware System Platform*:** O *Wonderware System Platform* é uma plataforma de supervisão integrada e unificada essencial para os processos e sistemas corporativos. O *Wonderware System Platform* serve como uma fonte única de informações úteis para utilizadores operacionais, de engenharia e cooperativos. O *Wonderware System Platform* fornece aos utilizadores uma visualização rica em informações tais como: alarmes inteligentes, ferramentas de análise e relatórios flexíveis, extensões *plug-ins*, como o MES, *Workflow*, *Quality*, *Batch* e *Enterprise Integration* para produção sustentável e melhoria do desempenho operacional. [7]
- ***Ignition*:** Conjuga um modelo de licenciamento ilimitado com implementação instantânea web-based e possui um conjunto de ferramentas SCADA. [8]

Como considerações finais, a integração de um sistema de SCADA é, cada vez mais, uma ferramenta com inúmeras vantagens para as empresas. Permite um maior controlo e análise de dados em todos os procedimentos, o que leva a uma melhoria na produtividade.

No Capítulo 3 é aprofundando, em maior detalhe, o *sofware Ignition* e as suas características, que permitem uma forte presença no mercado mundial. Este *software* SCADA foi escolhido para o desenvolvimento da aplicação que fundamenta o presente relatório, devido às condições de licenciamento que oferece, juntamente com os benefícios na simplicidade de comunicação com BD e PLCs.

3. Solução *IGNITION*, da *Inductive Automation*

O *Ignition* é uma plataforma de software fornecida pela *Inductive Automation*, uma empresa fundada em 2003 por Steve Hechtman e que tem a sua sede em Folsom, no estado da Califórnia, EUA. A sua criação teve por base a frustração do seu fundador por não conseguir corresponder às necessidades dos clientes com as soluções que existiam. Graças à sua visão, nasceu o *Ignition* com a finalidade de possibilitar os seus clientes de tornarem ideias realidade ao remover obstáculos, quer tecnológicos quer económicos. [9]

O *Ignition* abrange várias modalidades de funcionamento. Pode operar como Interface Homem Máquina, permitindo ao utilizador monitorizar, em tempo real equipamentos de campo e ter *feedback* do seu estado. Identicamente, pode operar como SCADA, possibilitando adquirir informação e armazenar num base de dados, assim como ter um sistema de alarmes. Além disso, também funciona como MES, que designa em inglês *Manufacturing Execution System*, contudo esta funcionalidade do software não foi explorada. [10]

Para a aplicação desenvolvida no decorrer do estágio curricular, o *Ignition* operou mais como um sistema SCADA, tirando assim proveito das variadas funcionalidades deste software.

Esta solução em estudo, apresenta uma forte presença no mercado, estando presente nos mais diversos países e sendo confiada por milhares de empresas. Na figura 2 (retirada de [9]) estão expostas algumas das empresas que usam o *Ignition*.

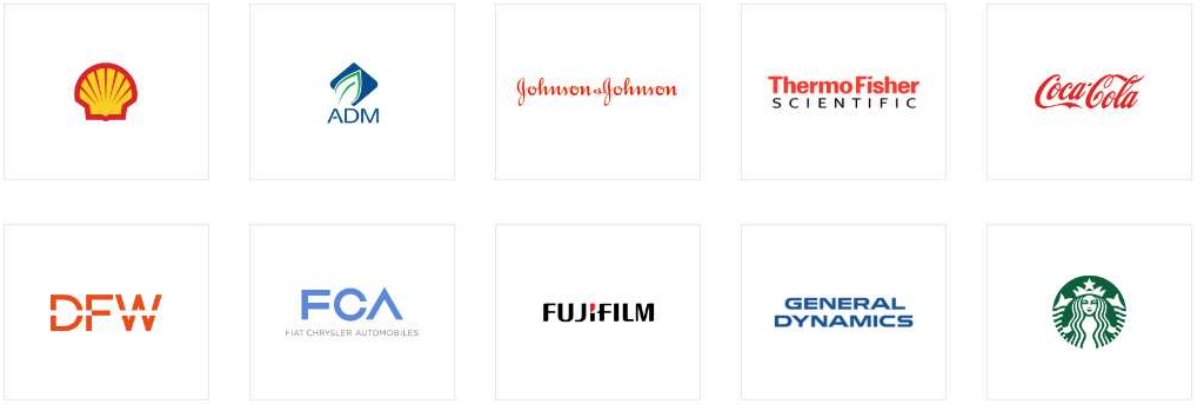


Figura 2 - Clientes Ignition

3.1 – Arquitetura *Ignition*

Esta solução apresenta diferentes arquiteturas, possibilitando uma maior flexibilidade de opção para desenvolver soluções. É possível optar entre a arquitetura *Standard*, *Scale-Out*, *Hub & Spoke*, *Enterprise* e IIOT.

A arquitetura *Standard* consiste num único servidor *Ignition* conectado a um DB e aos vários PLCs e dispositivos do cliente. A arquitetura *Scale-Out* conecta várias *Gateways Ignition* de forma a criar um sistema descentralizado. A arquitetura *Hub & Spoke* permite interligar vários locais remotamente a uma *Gateway* central. Por sua vez, a arquitetura *Enterprise* possibilita enviar dados de forma independente para uma *Gateway* centralizada e para a cloud. Por último, a arquitetura IIOT, é possível interligar uma estrutura MQTT na cloud, ou numa *network* privada, ou até num sistema híbrido. [11]

Na Figura 3 (retirada de [11]), é possível observar um sistema de arquitetura Standard, que foi o tipo de arquitetura usada no desenvolvimento da aplicação que fundamenta este relatório.

Figura 3 - Sistema de arquitetura Standard

Seguindo o planeamento realizado numa fase inicial do estágio, é expectável que o *Ignition* comunique com o *Microsoft* SQL *Server* com BD da aplicação, com PLCs da conceituada marca *Siemens* por OPC-UA e com, pelo menos, um computador Cliente. O número de computador pode variar por cliente.

3.2 – Vantagens *Ignition*

Este software apresenta benefícios que o destaca dos seus concorrentes [12], benefícios esses que serão abordados em maior detalhe:

- **Solução *server based*:** Esta vantagem significa que o *Ignition* precisa de apenas um computador, para funcionar como servidor. Apenas um servidor e tudo o resto será conectado ao mesmo. Deste modo, uma solução pode ter quantos clientes e quantos *designers* forem necessários sem necessidade de instalação;
- **Licenciamento simples:** é necessária apenas uma licença por servidor. Com apenas uma licença, é possível ter um número ilimitado de *Clients*, um número ilimitado de *Tags* e um número ilimitado de projetos, sem custos adicionais.
- ***Web-based deployment*:** Este software é instalado, implementado e gerido usando a web. É possível conectar, alterar definições, atualizar projetos e criar *Tags* a partir de qualquer computador da rede.
- ***Web-launched* de *Designer* e *Clients*:** As ferramentas *Designer* e *Client* podem ser executadas a partir da web.
- **Segurança incorporada:** A segurança está incorporada em todos os aspetos do *Ignition*. É possível gerir utilizadores, editar permissões de acesso a determinadas páginas ou até proteger determinados componentes do projeto.
- **Arquitertura modular:** Como já referido anteriormente, o *Ignition* apresenta vários tipos de arquiteturas permitindo uma vasta variedade de conexões entre PLCs, BD, *Clients*, etc. Deste modo cada aplicação pode optar pela solução que melhor se adequa ao seu caso.
- ***Cross platform*:** O *Ignition* foi desenvolvido em Java, o que permite a sua instalação em qualquer sistema operativo. Pode ser executado no *Windows*, *Mac* OS X, *Linux*, iOs e *Andorid*.

A Figura 4 (retirada de [8]) ilustra alguns benefícios da utilização do *Ignition*.

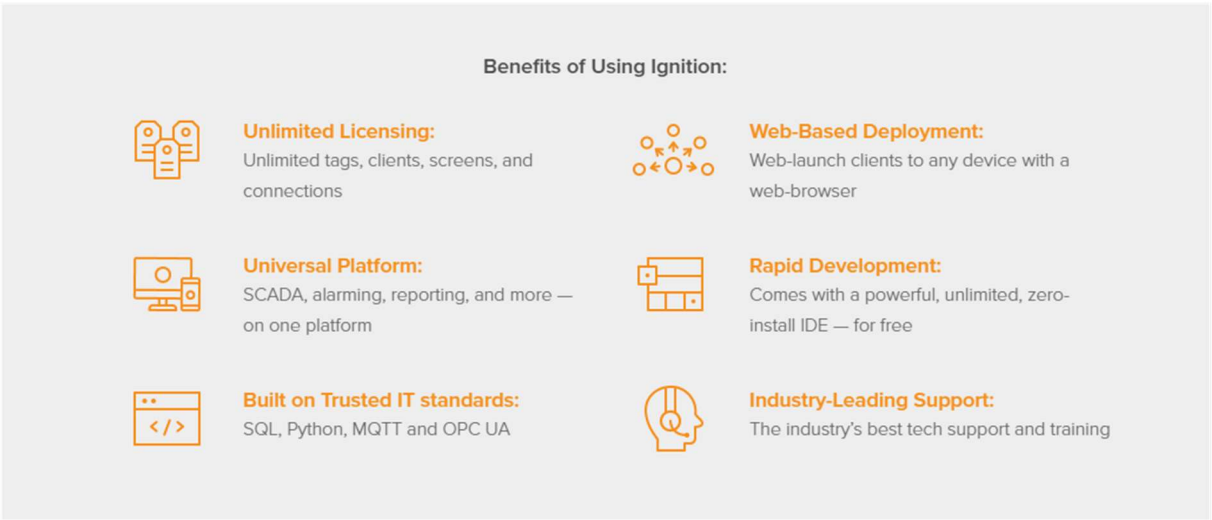


Figura 4 - Benefícios Ignition

3.3 – Formação *Inductive University*

Além das vantagens referidas anteriormente, o *Ignition* apresenta uma valência para atrair novos utilizador, um vasto curso de vídeos formativos totalmente gratuitos acompanhado de testes para creditação do progresso *online*. Esta formação foi realizada pelo estagiário com o objetivo de ganhar conhecimento no mesmo e facilitar assim o processo que se iria iniciar.

A Figura 5 mostra os cursos realizados ao longo da formação, que teve um tempo aproximado de quarenta a sessenta horas.

Figura 5 - Modulo formação Inductive University

Os vários cursos lecionados, continham informação bastante útil sobre tópicos como funcionamento da *gateway*, conceção ao PLC por OPC-UA, funcionamento de BD no *Ignition*, comportamento de objetos no *Designer* e como os controlar e código *Python* no *Ignition*. Todos estes conteúdos foram aplicados no desenvolvimento da aplicação.

Como mencionado, estes vídeos e testes estão disponíveis *online*, no site da *Inductive University*, e podem ser realizados por qualquer utilizador, apenas sendo necessário uma conta de *login*. Os conteúdos lecionados também estão agrupados por versão do software, sendo que a formação foi efetuada para a versão mais recente disponível na altura e, na qual, foi desenvolvida a aplicação.

4. Sistema de gestão e controlo desenvolvido

Após analisadas as variadas soluções para desenvolvimento de uma aplicação que permita a gestão e o controlo de equipamentos de placo, chegou-se à conclusão de que o *Ignition* seria a melhor plataforma para o fazer, nomeadamente devido à facilidade de comunicação com bases de dados e com PLCs.

No planeamento desta aplicação foi estabelecido como objetivo que deveriam existir dois modos de funcionamento: um modo de Simulação e um modo *On-line*. O modo de Simulação deve permitir aos operadores testar a movimentação de motores e observar o seu comportamento expectado num ambiente virtual e sem risco de avarias mecânicas, e um modo *On-line* que permita a operação de motores físicos conectados a um PLC.

Deste modo, optou-se por utilizar o software *Unity* para a componente de simulação e o TIA *Portal* para a programação do PLC. Toda a interface com o utilizador foi realizada no software SCADA, *Ignition*, recorrendo também a Bases de Dados para armazenar e transmitir informação. Na figura 6 encontra-se um diagrama funcional da solução pretendida, retratando todas as interligações entre os softwares utilizados na aplicação, assim como o fluxo de dados a movimentar.

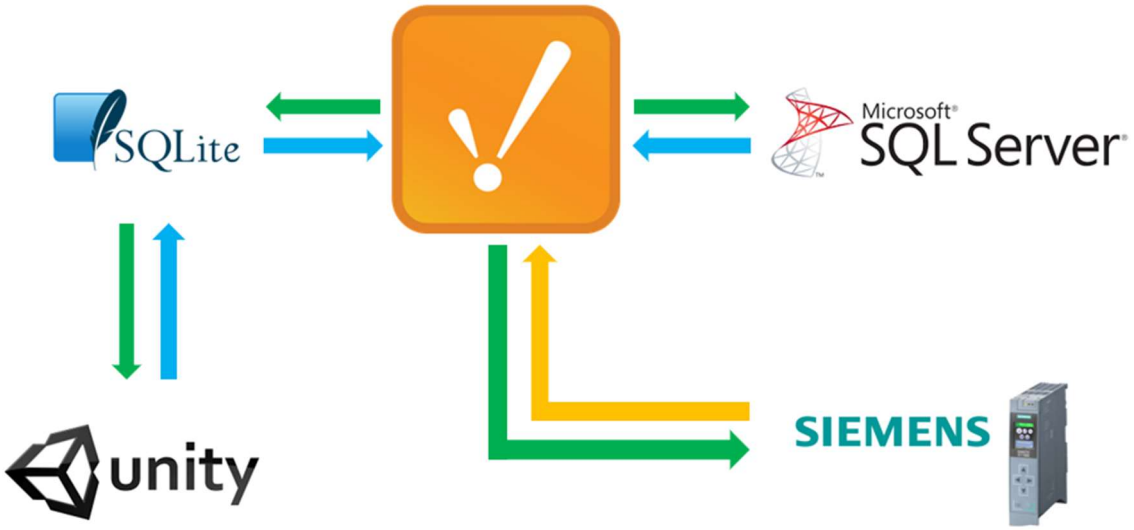


Figura 6 - Diagrama funcional das interligações entre softwares utilizados

De modo sucinto, o *Igntion* utiliza o MSSQL como base de dados principal da aplicação, enviando e recebendo dados conforme solicitado. Nesta BD estão todas as tabelas referentes à configuração de espetáculos, configuração de motores, listas de motores, lista de grupos, movimentação de motores, entre outros tabelas. O *Ignition* sempre que precisa de atualizar, ler ou enviar dados para a BD utiliza comandos SQL tais como *UPDATE, SELECT, INSERT* entre outros.

Em modo de simulação, o *Igntion* envia e recebe dados para o *SQLite*. Esta DB envia dados para o *Unity* que simula a movimentação dos motores e devolve os dados para o SQLite que, por sua vez, devolve a informação ao *Ignition*.

Em modo *on-line*, o *Ignition* envia os comandos necessários para a movimentação dos motores para o PLC, que dá as instruções necessárias aos equipamentos de campo para se movimentarem. O feedback dos movimentos é devolvido ao *Igntion* para que o operador tenha acesso a informação atualizada. Neste modo a comunicação é feita por OPC-UA. É de salientar, como ilustra a Figura 6, que não foi possível concluir em tempo útil de estágio a programação do PLC, logo o *Ignition* não recebe feedback do autómato. No seguimento do estágio e com o continuar do desenvolvimento da aplicação, este processo foi concluído.

Nos subcapítulos seguintes serão descritos os vários processos de desenvolvimento e construção do sistema de gestão e controlo desenvolvido. Para tal, serão explicadas a metodologia usada em todos os processos envolventes base de dados (SQLite e MSSQL) e os processos usados com o software *Unity* para a solução de simulação da aplicação, e a interface criada em *Ignition* assim como uma descrição detalhada das suas funcionalidades.

4.1 – Bases de Dados

Uma base de dados é uma ferramenta de recolha e organização de informações. As bases de dados podem armazenar informações sobre pessoas, produtos, encomendas ou qualquer outro assunto. Uma base de dados informatizada é um contentor de objetos, que pode conter mais do que uma tabela. [13]

No decorrer do estágio e para o desenvolvimento da aplicação foram utilizadas duas bases de dados distintas, o SQLite e o MSSQL, as quais serão abordadas de forma distinta.

4.1.1- SQLite

O SQLite é uma biblioteca de linguagem C que implementa código SQL de forma rápida, compacta, independente e de alta confiabilidade. SQLite é o motor de base de dados mais usado a nível mundial e está presente em telemóveis, na maioria dos computadores e em inúmeras aplicações utilizadas diariamente. [14]

Na figura 7 está representado um diagrama com as dependências das interligações nas quais o SQLite está presente.

Figura 7 - Diagrama funcional das interligações do SQLite

Como se pode constatar, o SQLite serve de ponto de interligação entre dois softwares: o *Igntion* e o *Unity*. Tem como propósito para a aplicação desenvolvida, receber valores do sistema de SCADA e enviar os dados para o simulador. Após o processamento de dados no *Unity*, a informação volta a ser enviada para a BD que devolve ao SCADA.

Dentro da BD existe uma estrutura de *arrays* para cada eixo/motor. Dentro de cada um desses eixos, estão contidos os bits de controlo (*Control Word*) necessários para o *Unity* simular o movimento, e os bits que refletem o estado do eixo (*Status Word*), usados do lado do SCADA.

4.1.2 – Microsoft SQL Server

Microsoft SQL Server é um Sistema de gestão de base de dados (RDBMS) que suporta uma vasta variedade de processamento de transações e analise em ambientes informáticos. [15]

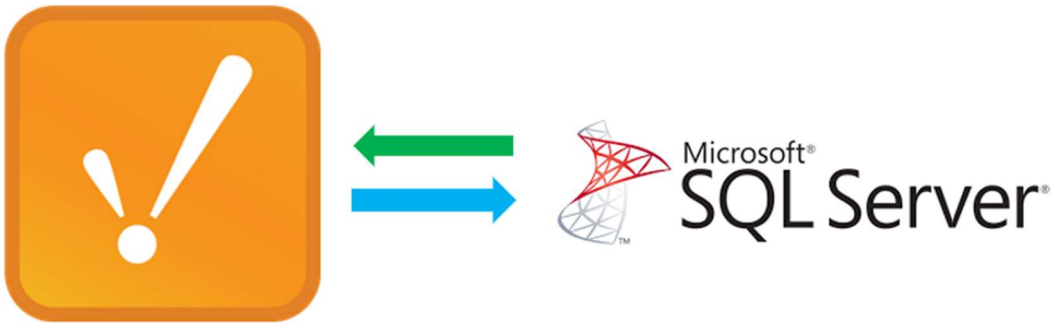


Figura 8 - Diagrama funcional das interligações do MSSQL

Pela análise da figura 8, podemos constatar que a funcionalidade desta BD é unicamente comunicar com o SCADA.

Contudo, apesar de apresentar um diagrama funcional mais simplificado, esta DB possui um maior grau de importância, pois armazena todo o tipo de dados para a

aplicação em desenvolvimento. É nela que estão armazenadas todas Tabelas existentes na aplicação, todas as *Views* e todos os *Stored Procedures*.

4.1.2.1 - Tabelas SQL

As tabelas criadas foram uma parte preponderante em todo o processo da aplicação, pois permitiu uma melhor organização e acessibilidade de dados. À data da conclusão do estágio curricular, a aplicação que tinha vindo a ser desenvolvida contava com um total de dezanove tabelas, como se pode visualizar na Figura 9.

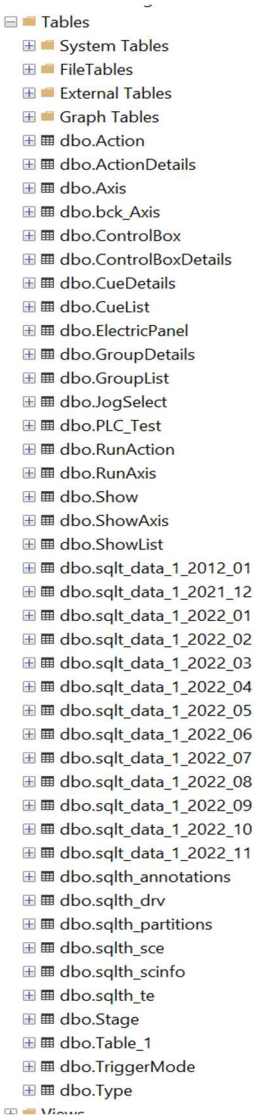


Figura 9 - Lista de Tabelas MSSQL

As tabelas, serviram para armazenar informação sobre os mais variados componentes da aplicação, como por exemplo, informação detalhada sobre cada eixo, informação sobre cada lista de eixo, grupos criados, cada espetáculo, cada zona do placo, cada tipo de ação e sobre o comportamento de cada eixo dentro dessa mesma

movimentação numa ação. Estes dados e outros formam armazenados de forma eficiente dentro de tabelas.

É de salientar a importância da tabela *dbo.Axis,* que foi a primeira a ser criada. Na Figura 10, podemos analisar todas as colunas que constituem esta tabela e os respetivos *data types* usados.

GNITION.STAGE - dbo.Axis

Column Name	Data Type	Allow Nulls
[Data.ID]	int	☐
[Data.SystemID]	int	☐
[Data.Name]	nvarchar(50)	☑
[Data.SystemName]	nvarchar(50)	☑
[Data.Selected]	bit	☑
[Data.Allowed]	bit	☑
[Data.StageID]	int	☑
[Data.Type]	int	☑
[Data.Notes]	nvarchar(50)	☑
[Position.Actual]	real	☑
[Position.Target]	real	☑
[Position.HiScene]	real	☑
[Position.HiRange]	real	☑
[Position.HiHiLimit]	real	☑
[Position.HiLimit]	real	☑
[Position.HiOperate]	real	☑
[Position.LoOperate]	real	☑
[Position.LoLimit]	real	☑
[Position.LoLoLimit]	real	☑
[Position.LoRange]	real	☑
[Position.LoScene]	real	☑
[Position.Home]	real	☑
[Position.Unit]	int	☑
[Move.Acceleration]	real	☑
[Move.Deceleration]	real	☑
[Move.HomeVelocity]	real	☑
[Move.JogVelocity]	real	☑
[Move.Velocity]	real	☑
[Motor.Power]	real	☑
[Motor.Voltage]	real	☑
[Motor.Current]	real	☑
[Motor.GearRatio]	real	☑
[Motor.EncoderRatio]	real	☑
[Load.Bypass]	bit	☑
[Load.Actual]	real	☑
[Load.Delay]	real	☑
[Load.Max]	real	☑
[Load.Min]	real	☑
[Load.Monitor]	bit	☑
[Log.Modified]	datetime	☑
[Log.ModifiedBy]	nvarchar(50)	☑
[Positioning.Location]	int	☑
[Group.Connected]	int	☑
[Color.R]	int	☑
[Color.G]	int	☑
[Color.B]	int	☑
[Controller.ID]	int	☑
[ElectricPanel.ID]	int	☑

Figura 10 - Estrutura tabela Axis

Dada a complexidade desta tabela e do seu elevado número de colunas, a sua organização teve de ser bem planeada. Nela estão contidos dados sobre o próprio eixo nas colunas com o prefixo *Data*, valores de posição nas colunas com o prefixo

Position, valores e limites referentes à velocidade em colunas com o prefixo *Move*, configuração elétrica nas colunas com o prefixo *Motor*, dados referentes à carga permitida nas colunas com o prefixo *Load* e algumas colunas com dados adicionais.

4.1.2.3 - *Views* SQL

Uma *View* em SQL consiste numa tabela virtual, na qual o seu conteúdo é definido através de um *query*. Assim como nas tabelas, uma *View* abrange numa série de colunas e linhas com dados elencados. [16] Dados esses que são recolhidos através de uma instrução *SELECT*. O resultado deste comando, forma a tabela virtual pretendida.

Na Figura 11 estão elencadas as *Views* usadas na aplicação desenvolvida.

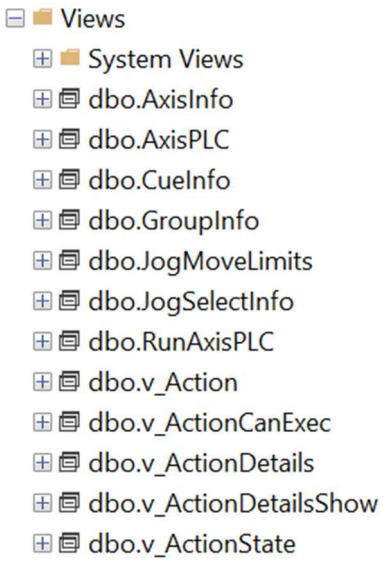


Figura 11 - Lista de Views MSSQL

A utilização desta ferramenta do MSSQL facilita a consulta de resultados, pois a informação já foi filtrada e analisada, tornando assim a visualização para o utilizador muito mais clara.

Uma *View* ao ser criada pode selecionar informação de várias tabelas já existentes. Como mostra a Figura 12, a *View dbo.GroupInfo* tem informação de quatro tabelas distintas: *dbo.GroupList, dbo.GroupDetails, dbo.Axis* e *dbo.Stage*. Além de selecionar de que tabelas é recolhida a informação, também são selecionadas de que colunas os dados são extraídos.

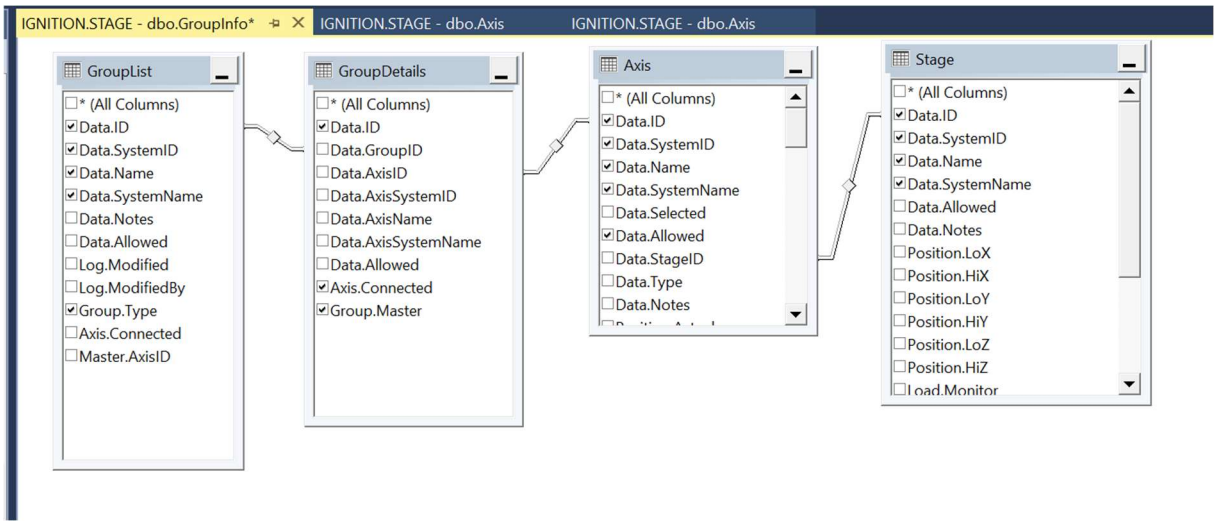


Figura 12 - View dbo.GroupInfo

4.1.2.3 - *Stored Procedures* SQL

Um *Stored Procedure* é um código SQL que pode ser executado as vezes necessárias, conforme seja chamado. Pode conter rotinas cíclicas, validação de parâmetros, devolver valores e indicar a sua execução ou falha. Esta ferramenta tem a grande vantagem de melhorar o desempenho da solução, uma vez que o código executado se torna parte da execução única do servidor, tornando assim todo o processo mais rápido.

Para a aplicação desenvolvida, foram criados quatro *Stored Procedures*, como retrata a Figura 13.

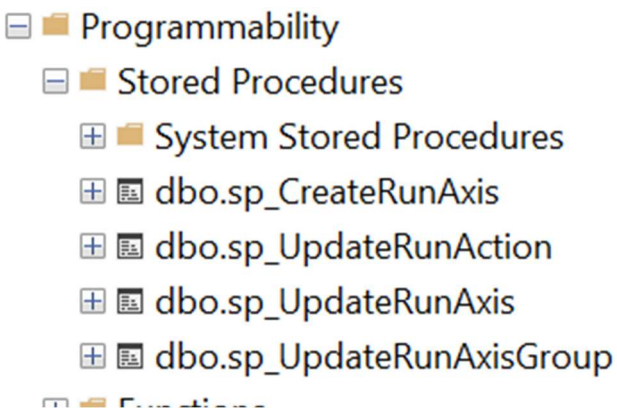


Figura 13 - Lista Stored Procedures SQL

Os procedimentos criados, possuem finalidades distintas, as quais serão descritas com maior pormenor:

- **dbo.sp_CreateRunAxis:** Sempre que executado, este procedimento cria a tabela RunAxis;

- **dbo.sp_UpdateRunAction:** Este procedimento é executado sempre que existe um comando de movimentação em modo Automático. É atualizada a lista de ações que pode executar conforme as suas prioridades. Todos os eixos que pertençam a uma ação que teve o seu estado atualizado, são também atualizados;
- **dbo.sp_UpdateRunAxis:** Assim como o procedimento anterior, este é executado no mesmo momento. Quando um eixo tem condições para se movimentar, este *Stored Procedure* analisa os seus valores de posição atual, posição de destino, velocidade e tempo de movimento configurados previamente pelo utilizador. Os cálculos de velocidade, aceleração e desaceleração são refeitos de modo a verificar se os valores estão dentro dos limites;
- **dbo.sp_UpdateRunAxisGroup:** Estes procedimento serve exclusivamente para atualizar os dados dos grupo, na Tabela RunAxis. Uma vez que a configuração de grupos pode ser configurada em vários momentos, requer um procedimento dedicado.

Por motivos de confidencialidade a nível contractual com a entidade onde se realizou o estágio, não é possível mostrar o código desenvolvido.

4.2 – *Unity*

O *Unity* é um motor de jogos e IDE (ambiente de desenvolvimento integrado) para criação de ambiente gráfico virtual interativo e vídeo jogos. Nas palavras de David Helgason, o *Unity* é um conjunto de ferramentas usado para construir jogos, e é a tecnologia usada para a criação de gráficos, de áudio e as interações com a rede. *Unity* é famoso pela sua rápida capacidade de criar protótipos, e pelo elevado número de publicações. [17]

4.2.1 - Linguagem de Programação

O *Unity* oferece aos seus utilizadores três linguagens de programação para o desenvolvimento de aplicações: *Boo*, *Unity JavaScript*, também designado como *UnityScript*, e C#. Apesar de diferentes, estas três linguagens permitem a implementação do mesmo conteúdo. [17]

Boo apresenta uma sintaxe semelhante a *Python* e uma estrutura idêntica a *UnityScript*. É uma linguagem de programação com uma taxa de utilização mais baixa dentro dos utilizadores de *Unity*. *UnityScript* é uma linguagem que apresenta algumas semelhanças com *Javascript* e costuma ser uma melhor escolha para novos utilizadores. Das três, é a linguagem que apresenta uma maior taxa de utilização dentro dos utilizadores do *Unity*, o que facilita a assistência e a resolução de problemas. *UnityScript* suporta escrita dinâmica, o que acaba por ser menos eficiente

do que C#, uma vez que o compilador tem de ter em consideração se o tipo de objeto sofre alterações. C#, é uma linguagem que apresenta um maior grau de complexidade, quando comparada com *UnityScript*. Contudo permite ao utilizador ter um maior controlo sobre as ações a executar. [17] A principal desvantagem da utilização de C# com o *Unity* é a falta de compatibilidade e integração com os componentes da aplicação, o que obriga a configuração dos mesmos manualmente.

Uma das características que sobressaem no *Unity* são as suas propriedades visuais. Estas propriedades visuais permitem alterar, diretamente no editor, variáveis públicas, sem ter de modificar o código. [17]

4.2.2 - Integração do Unity na aplicação

Uma das principais valências da aplicação que fundamenta este relatório, é a sua componente de simulação que permite aos seus utilizadores testarem e configurarem a movimentação dos motores que serão utilizados nos espetáculos. Para tal, optou-se por desenvolver este simulador no *Unity*. Outro fator que levou à escolha deste software, foi o facto de haver antecedentes na empresa da sua utilização em projetos diferentes.

Como já referido anteriormente para simular a movimentação, o Unity recebe e envia dados através do SQLite. Esta BD possui um *array* com o número de motores a simular. Dentro dessa estrutura estão organizados os valores por *Control Word*, *Control Position, Move Position* e *Status Word*. Todas estas *Tags* estão mapeadas, do lado do SCADA, para ler ou escrever os valores, referentes ao motor a que estão atribuídas, na BD do SQLite. Esta BD, por sua vez, mapeia para o Unity.

Para controlar a simulação, temos uma estrutura designada aos comandos de controlo, a *Control Word*. Visível na Tabela 2, temos os bits pertencentes à *Control Word* juntamente com os seus *data types*. Estas *Tags* são lidas do lado do simulador e, conforme o seu estado, são simulados os movimentos do motor em questão.

As *Tags* da *Control Word* são lidas do lado do simulador e, conforme o seu estado, são simulados os movimentos do motor em questão.

Tabela 2 - Control Word Unity

Control.Enable	Bool
Control.GoPos	Bool
Control.Home	Bool
Control.JogMinus	Bool
Control.JogPlus	Bool
Control.Power	Bool
Control.Reset	Bool
Control.Mode	Int
Control.Velocity	Real
Control.TargetPosition	Real

Além da *Control Word*, o Simulador também precisa de analisar os dados da *Control Position* e da *Control Move*, apresentados na Tabela 3 e na Tabela 4, respetivamente. Estas duas tabelas possuem valores a verificar, dentro do simulador, antes de realizar qualquer movimento.

Na *Control Position* são enviados dados referentes aos limites operacionais e de segurança nos quais o motor pode realizar movimentos. Se algum dos limites for ultrapassado é gerado um alarme na *Status Word*. Se, porventura, houver um erro no valor mapeado pelo *Ignition* não for válido o próprio simulador corrige e só permite a movimentação dentro do range operacional previamente configurado.

Tabela 3 - Control Position Unity

Position.HiHiLimit	Real
Position.HiLimit	Real
Position.HiOperate	Real
Position.LoOperate	Real
Position.LoLimit	Real
Position.LoLoLimit	Real

Já na *Control Move* encontram-se os valores que limitam a velocidade de operação dos motores. Independentemente do tipo de movimento que estamos a simular, os valores configurados não podem ser ultrapassados, por motivos de segurança. Nesse cenário, se for solicitada uma maior velocidade ou aceleração do que a permitida, o simulador limita conforme o valor previamente configurado. Deste modo, os valores obtidos em simulação serão idênticos aos replicados no campo uma vez que o motor físico também se encontra limitado.

Tabela 4 - Control Move Unity

Move.MaxAcceleration	Real
Move.MaxDeceleration	Real
Move.MaxHomeVelocity	Real
Move.MaxJogVelocity	Real
Move.MaxMoveVelocity	Real

Após simulado o movimento, são atualizados as *Tags* da *Status Word* que serão mapeados para a BD e de novo para o *Ignition*. Como podemos constatar na Tabela 5, para cada motor existe informação referente ao seu estado, se está em movimento, vários alarmes para diferentes situações e a sua posição.

Tabela 5 - Status Word Unity

Status. Enabled	Bool
Status.Fault	Bool
Status.Homed	Bool
Status.MoveMinus	Bool
Status.MovePlus	Bool
Status.OTMinus	Bool
Status.OTPlus	Bool
Status.OTSMinus	Bool
Status.OTSPlus	Bool
Status.Powered	Bool
Status.Ready	Bool
Status.InPosition	Bool
Status.ActualPosition	Bool

Deste modo, a informação enviada para o *Ignition* do Simulador retrata de uma forma realista o comportamento dos motores físicos.

Relativamente aos modos de funcionamento configurados no simulado, cada motor pode exercer três tipos de movimentos, que são determinados pela *Control Word*, com a variável *Control.Mode*. Este inteiro, definido pelo utilizador no *Ignition*, tem o seu valor de *default* a zero e pode variar até três, sendo que o *Control.Mode* zero não executa nenhuma ação.

O *Control.Mode* (modo) um, representa um movimento em posição que ignora os fins de curso, permitindo ao motor mover-se sem limitações de posição. Conforme a variável que vem do *Ignition* do *Control.JogPlus* e do *Control.JogMinus* o motor terá um incremento ou decremento na sua posição, respetivamente.

Por sua vez, o *Control.Mode* dois simula um movimento em posição dentro dos limites operacionais. Conforme o comando *Control.JogPlus* e do *Control.JogMinus* o motor terá um incremento ou decremento na sua posição até atingir os limites estabelecidos na *Control Position.*

Já o *Control.Mode* três simula um movimento em posição. Na *Control Word* é estabelecido um *setpoint* de posição para o motor selecionado. O *setpoint* configurado tem de respeitar os limites operacionais do motor. Caso isto não aconteça, o simulador estabelece como posição de destino para o movimento a posição limite no sentido do movimento.

Para os três modos, os valores de velocidade, aceleração e desaceleração estão configurados na *Control Word* e respeitam o limites da *Control Move*.

4.3 – Estado da aplicação

A última componente desta aplicação, desenvolvida em tempo útil de estágio, o *Ignition*. É, com base neste software SCADA e nas suas variadas características, que se vai moldar e criar o aspeto final do sistema de gestão e controlo para equipamentos de palco.

Desde o início, que a finalidade era desenvolver uma aplicação intuitiva para o operador, que fosse eficiente na sua utilização.

Os seguintes *prints* retratam o estado da aplicação, salvaguardando que com a continuidade do desenvolvimento da aplicação, podem não representar o seu estado mais atualizado. Esta aplicação irá correr num monitor de 27 polegadas, *touchscreen*, com uma resolução 4K. É de salientar que a elevada resolução pode não representar da melhor forma a visualização no presente relatório. As imagens expostas terão algum nível de detalhe e atenção ao tópico abordado e, em anexo, seguem *prints* que retratam um estado global das páginas abordadas.

Figura 14 - Janela inicial da aplicação

A Figura 14 mostra a página inicial desta aplicação. Este *layout* inicial foi criado com a intenção de introduzir o operador a esta solução de uma forma simples, informativa e que fornece opção de escolha ao operador das opções disponíveis.

Na parte superior do ecrã, temos o *Top Banner,* uma zona de visualização sempre visível em qualquer momento. Nela pode ser configurado o idioma, o operador, selecionar espetáculo a configurar, operar o relógio, analisar LEDs indicativos de estados gerais e informação da data e hora.

Na lateral esquerda do ecrã, temos uma zona designada aos painéis de navegação. Esta zona está sempre visível, contudo os painéis vão variando conforme o operador

vai navegando pela aplicação. Na figura 15 estão visíveis todos os cinco painéis de navegação existentes nesta solução.

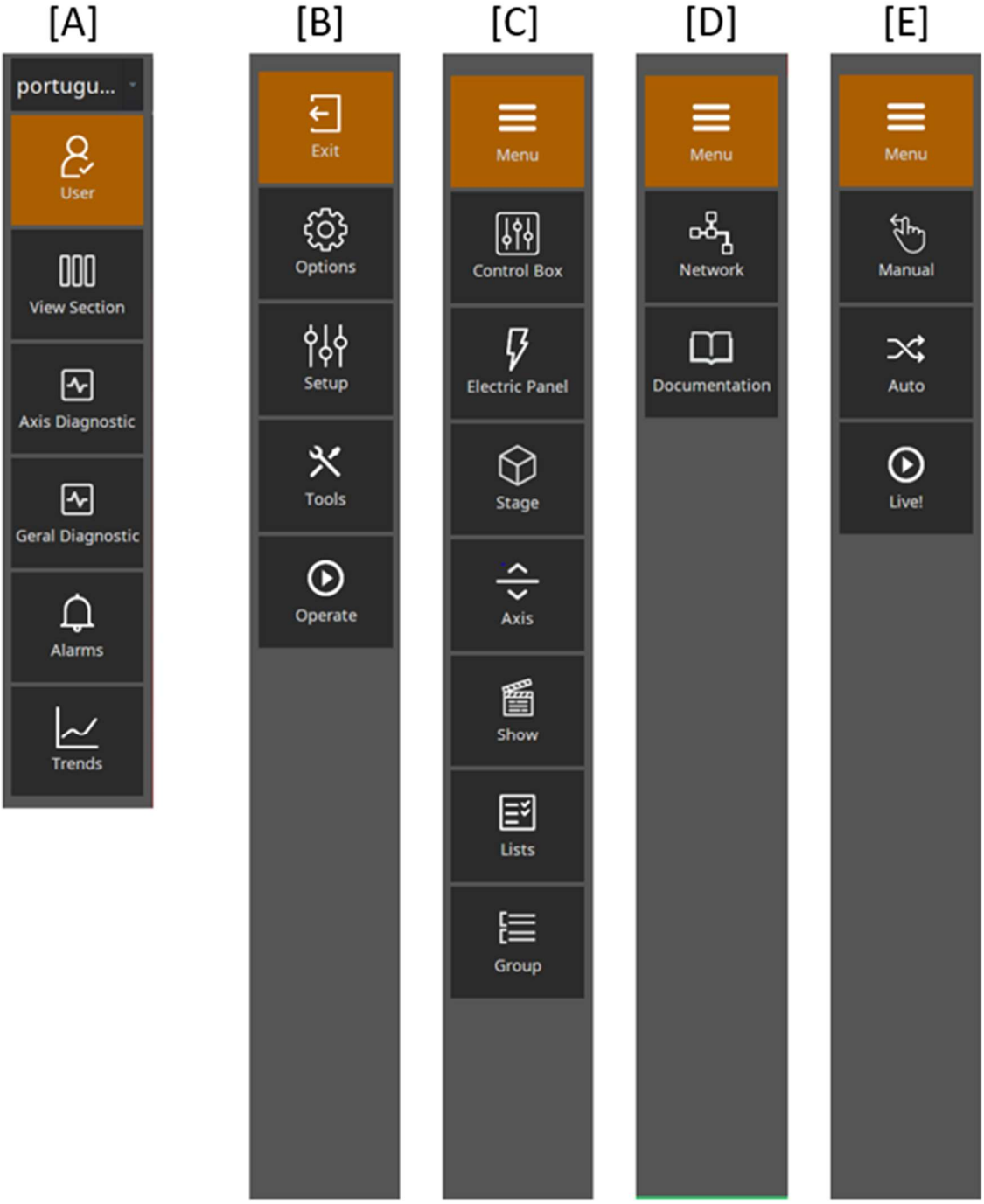


Figura 15 - Painéis de navegação

O painel [A], encontra-se sempre visível em qualquer situação da aplicação. Nele é possível selecionar seis botões com configurações distintas.

- **'User':** Abre um *popup* de *login*;
- **'View Section':** Abre uma janela onde é possível visualizar a posição atual de todos os eixos configurados;

- **‘Axis Diagnostic’:** Abre uma janela na qual é possível analisar o estado dos *bits* da *Status Word* e da *Control Word* para cada eixo;
- **‘Geral Diagnostic’:** Abre uma janela de visualização dos estados das *Control Box’s* existentes e dos estados dos painéis elétricos configurados;
- **‘Alarms’:** Abre a janela de alarmes, na qual estão presentes todos os alarmes na aplicação, assim como o seu histórico;
- **‘Trends’:** Abre a janela dos *trends*, na qual podemos analisar um gráfico gerado com a posição atual de cada motor.

O painel [B] apresenta cinco opções. Este painel está visível no início da aplicação e pode variar conforme as opções escolhidas.

- **‘Exit’:** Abre um *popup* no qual é possível fechar a aplicação;
- **‘Options’:** Abre uma janela de opções na qual é possível ao operador configurar alguns parâmetros da aplicação. De momento, apenas existe a opção de alterar entre os modos de simulação e *online*;
- **‘Setup’:** Abre o painel de navegação [C];
- **‘Tools’:** Abre o painel de navegação [D];
- **‘Operate’:** Abre o painel de navegação [E].

O painel [C] contem as várias opções disponíveis para o operador fazer a configuração da aplicação, dos mais variados elementos.

- **‘Menu’:** Volta para o painel de navegação principal([B]);
- **‘Control Box’:** Abre uma janela onde se realiza a configuração das *Control Box´*s existentes para cada solução. Cada *Control Box* usa o MAC *Adress* do seu computador associado como forma identificativa.
- **‘Electric Panel’:** Abre uma janela onde se pode parametrizar os quadros elétricos que integram a solução;
- **‘Stage’:** Abre uma janela onde se configuram os limites físicos do palco e zonas de operação existentes na sala de espetáculos em que a aplicação será usada;
- **‘Axis’:** Abre uma janela de configuração dos motores usados;
- **‘Show’:** Abre uma janela onde se configuram e criam os vários espetáculos existentes;
- **‘Lists’:** Abre uma janela onde o operador pode agrupar os eixos existentes em Listas, para deste modo, facilitar a sua operação;
- **‘Group’:** Abre uma janela de criação e configuração de grupos.

No painel [D] dá ao operador opções de ferramentas que podem ser uteis no manuseamento da aplicação.

- **‘Menu’:** Volta para o painel de navegação principal ([B]);
- **‘Network’:** Abre uma janela informativa, na qual está retratada a tipologia de network usada;
- **‘Documentation’:** Abre uma janela que contem informação referente a esquemas elétricos, manuais de utilizador e manuais dos variadores usados.

O painel [E], contem as opções responsáveis pela movimentação dos motores da aplicação.

- **'Menu':** Volta para o painel de navegação principal([B]);
- **'Manual':** Abre a janela referente ao modo Manual da aplicação;
- **'Auto':** Abre a janela configurar as ações a executar em modo Automático;
- **'Live':** Abre a janela do modo Automático.

Na Figura 16, podemos observar a janela *popup* de *Login*. Nela pode ser feito o *login*, *log out* e fazer a gestão de contas. A opção de gestão de contas, com vários graus de permissões já é uma função nativa do próprio *Ignition*, sendo que foi aproveitada.

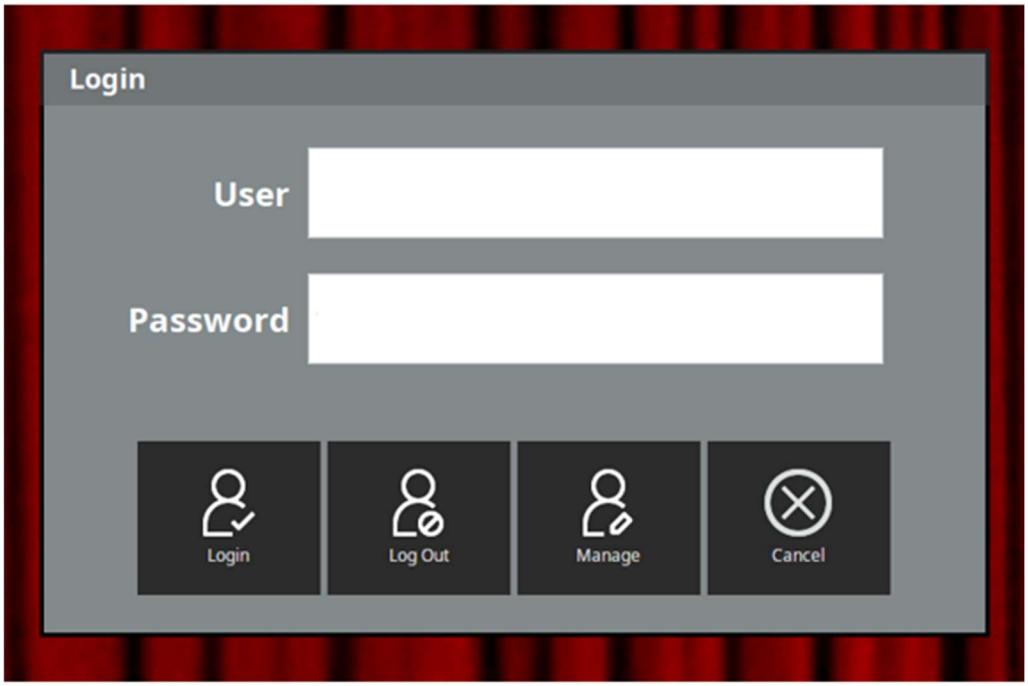


Figura 16 - Pop-Up Login

4.3.1 – Janelas de Visualização

De modo a fornecer ao utilizador informação clara sobre o estado da aplicação, foram criadas um conjunto de páginas designadas a componentes de visualização, as quais serão abordadas de seguida.

A janela *'View Section'* foi criada com o intuito de fornecer ao operador uma visão global do estado e posição de todos os motores, de uma forma rápida e esclarecedora. Para tal, optou-se por criar três *templates* que permitam ao operador que tipo de visualização sobre os motores gostaria de ter. Para cada uma das opções, o operador pode escolher através da lista de eixos, que motores gostaria de visualizar. Além disso, existem três opções de visualização que permite ao operador escolher a opção de visualização que lhe é mais favorável para a situação em que se encontra.

Na Figura 17, encontra-se uma vista mais detalhada sobre as posições que cada motor ocupa, os seus limites operacionais, os seus limites de fim de curso, assim como os seus estados.

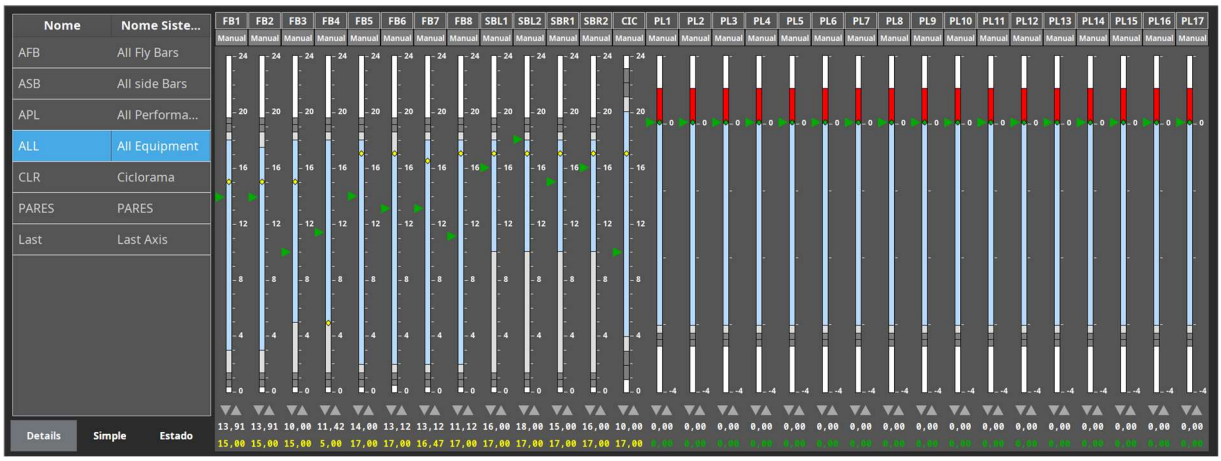


Figura 17 – Vista detalhada ViewSection

Na Figura 18, apresenta uma vista mais simplificada do estado e da posição de cada motor. Apenas indica os limites operacionais, contudo tem a vantagem de poder indicar ao operador de forma visual que tipo de componente e as suas dimensões se encontra agregado a cada motor.

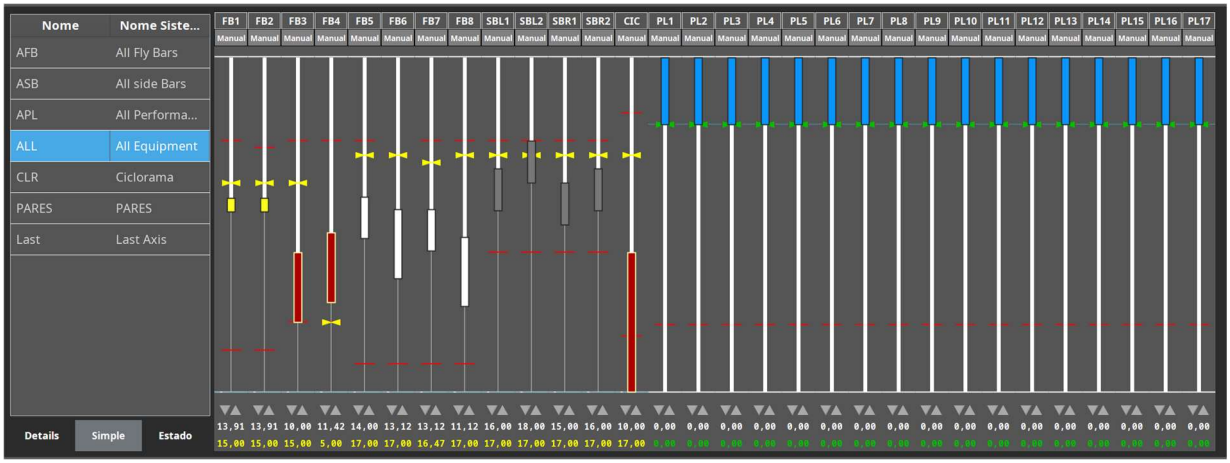


Figura 18 - Vista simplificada ViewSection

A Figura 19, mostra os mesmos dados de uma forma distinta e ainda mais simplificada, focando-se mais no estado de cada motor. Nesta vista é indicada a posição atual do motor, a carga atual que suporta, a sua corrente, o modo de operação, se possui algum acessório e qual a sua dimensão, e ainda, indicação do sentido do movimento.

Figura 19 - Vista estados ViewSection

A aplicação foi preparada para quando ocorre um erro num determinado motor, o seu diagnóstico seja o mais simples possivel. Para tal, foi desenvolvida uma página para o diagnóstico individual de cada motor, como ilustra a Figura 20. Esta página encontra-se dividida em três secções. Uma, na qual o operador pode selecionar o eixo desejado, numa tabela que elenca todos os eixos da aplicação. Esta tabela é obtida através de um *SELECT* ao SQL. Uma segunda secção que mostra os alarmes do eixo selecionado. Por último, temos uma secção mais detalhada sobre o estado do eixo selecionado. Os *bits* apresentam informação em tempo real da *Control Word, Status Word* e de posição, velocidade e carga atual de cada motor.

Figura 20 - Vista de Diagnósticos por motor

Além de uma janela dedicada ao diagnóstico individual de cada motor, foi também criada uma página na qual se pode analisar todos os alarmes gerados, que podemos ver na Figura 21. O operado, nesta janela pode fazer o reconhecimento dos alarmes, fazer um *reset* geral e consultar o histórico de alarmes.

Figura 21 - Vista alarmes

A ferramenta de alarmes é nativa do *Ignition*. Para configurar alarmes para uma *Tags* em específico, basta aceder à sua configuração e ativar a definição de alarmes. Um alarme pode ser ativado se ultrapassar um *setpoint* configurado, ou pode ter um valor

em específico. Neste caso, com os valores a tratar são *booleanos*, o seu alarme é gerado, sempre que o seu estado é '*True*' (um).

Em adição à análise que pode ser feita ao estado dos motores nas janelas mencionadas anteriormente, o operador pode também consultar uma janela de diagnósticos gerais, como mostra a Figura 22.

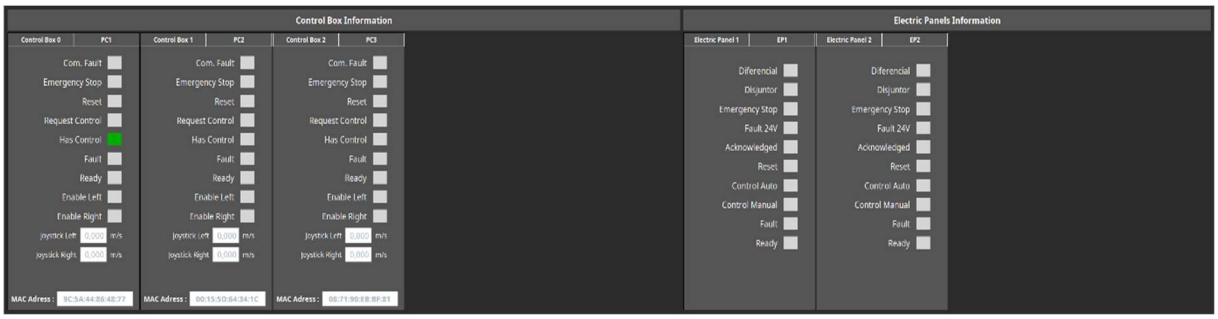


Figura 22 - Vista Diagnósticos gerais

Aqui pode-se observar de forma rápida e esclarecedora o estado dos vários equipamentos de controlo que a solução possui configurados, assim como, dos quadros elétricos existentes. Esta janela foi pensada para facilitar o diagnóstico em caso de alarmes que não sejam relacionados com falhas ou avarias em motores.

Na vista focada dos comandos (*Control Box*) podemos observar os estados de vários estados, como por exemplo falhas de comunicação ou paragens de emergência, mas também o estado dos *joysticks* e os valores de velocidade por eles recebidos. Por sua vez, a vista dedicada aos quadros elétricos (*Electric Panel*) é possível analisar vários estados de erro, remotamente.

Outra forma de fornecer ao operador a informação detalha sobre a posição dos eixos, é a utilização de *Trends*. Esta ferramenta nativa do *Ignition*, permite guardar em histórico *tags* e gerar com os seus valores, gráficos em função do tempo. A Figura 23, ilustra esta situação, na qual é visível a posição atual de certos eixos num espaço de tempo determinado pelo operador.

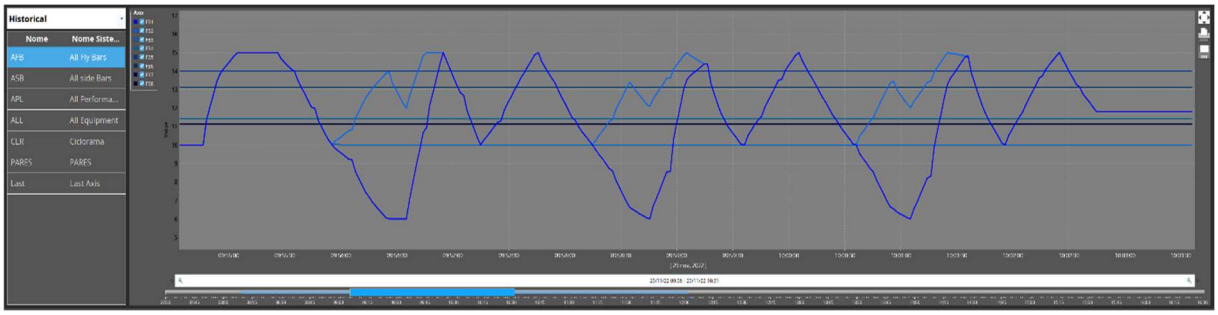


Figura 23 - Vista Trends

Esta função funciona em modo histórico ou em tempo real, opção que pode ser alterada pelo operador no canto superior da página 'View Section'.

4.3.2 - Janelas de Setup

Esta aplicação conta com janelas designadas à configuração de elementos importantes para a segurança na criação e elaboração de espetáculos, as quais serão abordadas com mais detalhe de seguida.

As janelas de configuração, ou *setup*, embora abordem tópico diferentes, apresentam uma estrutura semelhantes. Todas possuem uma tabela, por norma no canto superior esquerdo, que retrata o estado geral do componente a configurar. Esta informação contém o número de sistema, o nome dado pelo utilizador e informação especifica para cada tipo de cenário. De salvaguardar, que esta tabela é obtida através de um *SELECT* ao SQL. Do lado direito de cada tabela, foi colocado um componente *Ignition* chamado, *Tab Strip* onde o operador pode selecionar de que forma deseja configura a tabela, possuindo as operações 'Editar', 'Duplicar', 'Adicionar', 'Eliminar', 'Atualizar' e 'Gravar'. Este componente é dinâmico, sendo que a opção 'Gravar' apenas aparece visível quando são feitas alteração a um componente. O componente *Tab Strip* possui código *Python* com instruções para cada operação selecionada.

Existem também campos de edição onde o operador pode alterar a informação da Tabela. Estes componentes podem sofrer alterações sempre que o operador selecione uma linha da tabela e carregue 'Editar'. No final das alterações feitas, o operador precisa apenas de selecionar 'Gravar' e as alterações feitas serão atualizadas na tabela SQL através de um *UPDATE*.

Como já referido anteriormente, esta solução está pensada para ter um ou mais comando de controlo, designado por *Control Box*, na aplicação. Para tal existe uma página designada a configurar os vários comandos possíveis, como se pode ver na figura 24.

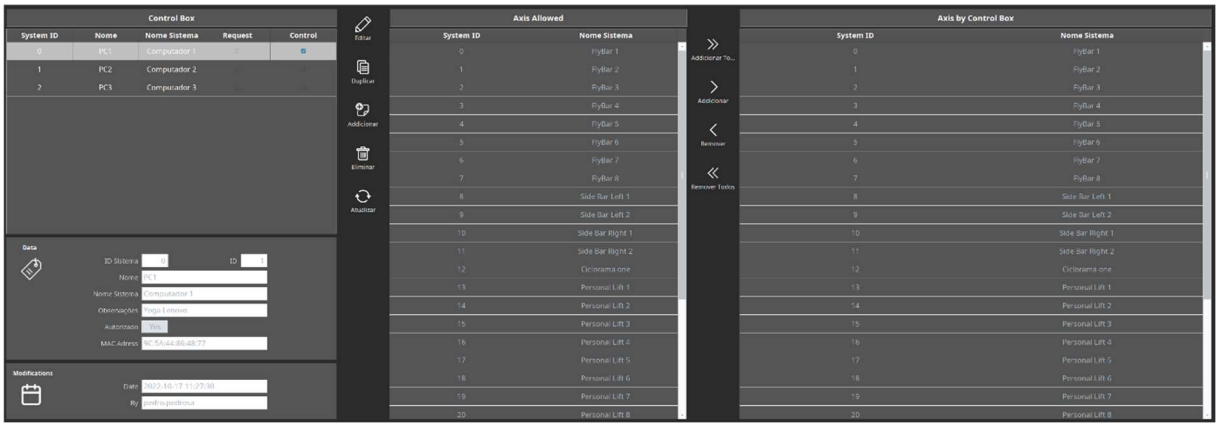


Figura 24 - Vista setup comando de controlo

Para cada *Control Box* é possível configurar o seu nome, atribuir um MAC *Address*, que serve como forma de identificação.

Outro dos componentes cuja configuração é de extrema importância, são quadros elétricos. Cada instalação pode conter um, ou mais quadros que necessitam de ser configurados do lado da aplicação, na página visível na figura 25.

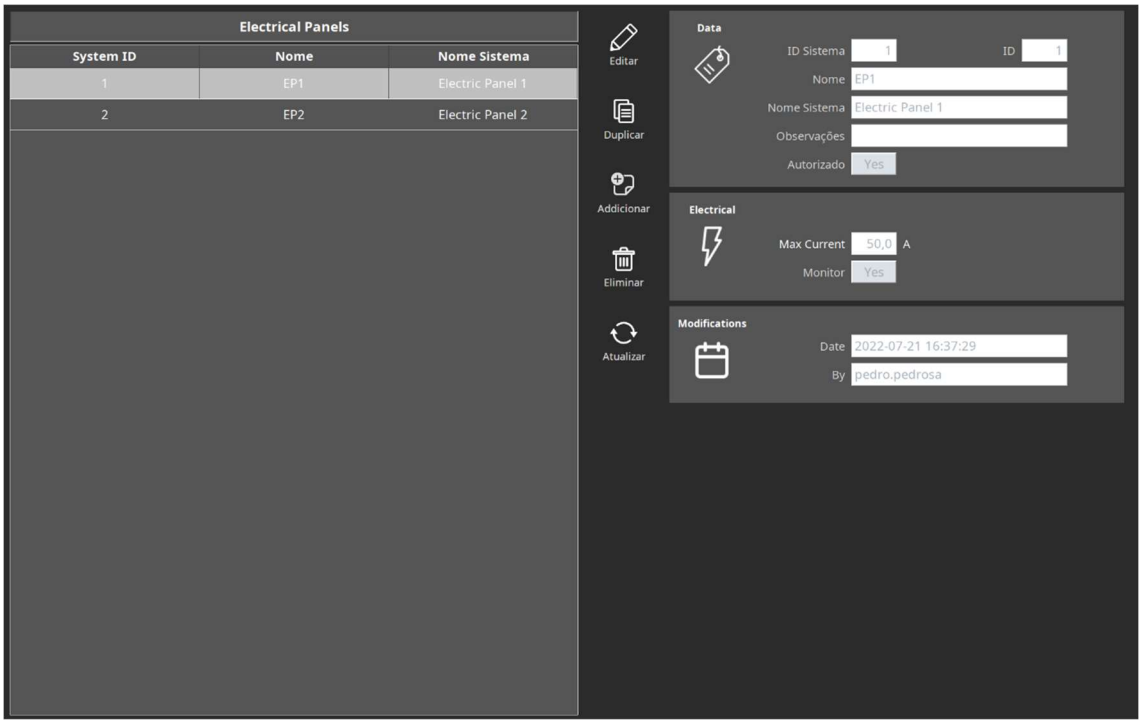


Figura 25 - Vista setup quadros elétricos

Para cada quadro, o operador pode definir um valor de corrente máxima e que motores serão alimentados por esse quadro.

As salas de espetáculos para as quais esta aplicação é pensada, variam em dimensões e podem ser divididas em várias zonas. Assim, necessitam de uma secção na aplicação onde essas zonas possam ser configuradas. Na Figura 26 encontra-se a página que permite realizar essa operação.

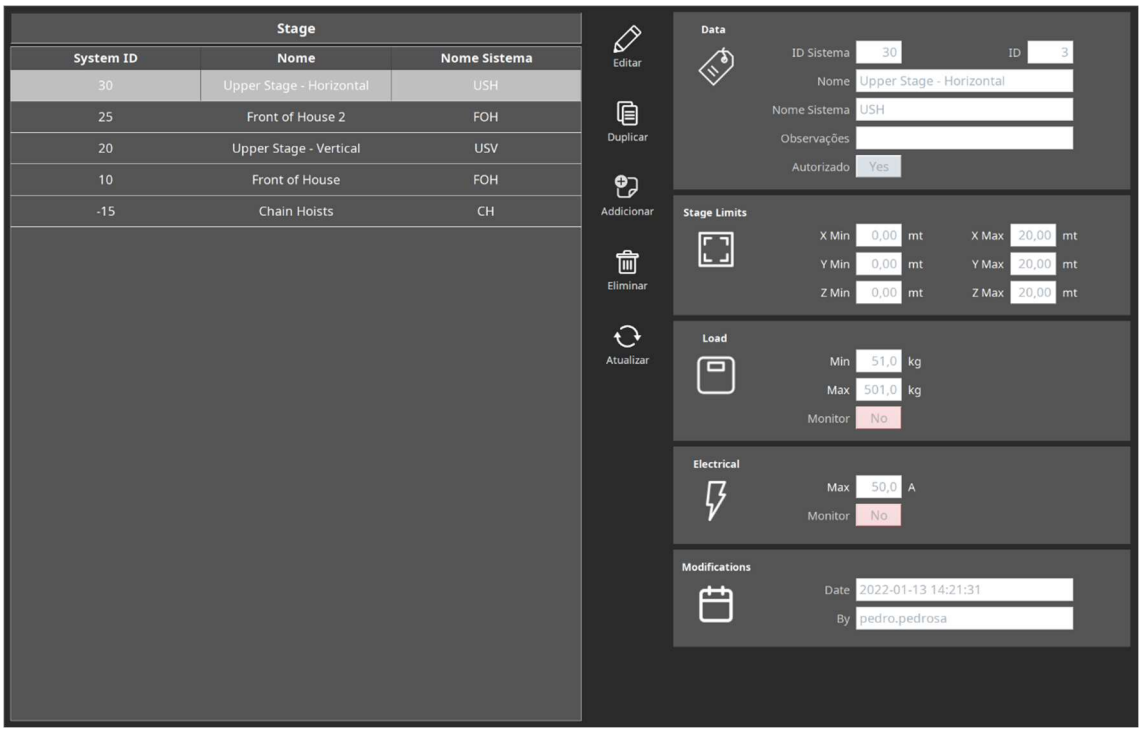


Figura 26 - Vista setup zonas de palco

O operador pode criar zonas novas, definir as suas dimensões, definir valores mínimos e máximos de carga por zona e definir um limite de corrente máximo para a zona em questão.

Contudo, os principais elementos a configurar são os motores a utilizar. A janela criada, visível na Figura 27, apresenta uma tabela onde estão listados todos os motores configurados, juntamente com os vários campos de preenchimento. Cada motor pode ser atribuído a uma zona de placo, configurada previamente, indicando o local da sua instalação e se está autorizado para ser utilizado ou não. Além disso, são estabelecidos pelo operador limites de posição, velocidade, carga e ainda limites de corrente. Além disso, pode ainda escolher-se uma cor a associar a cada motor para ser usada na visualização nos *Trends*.

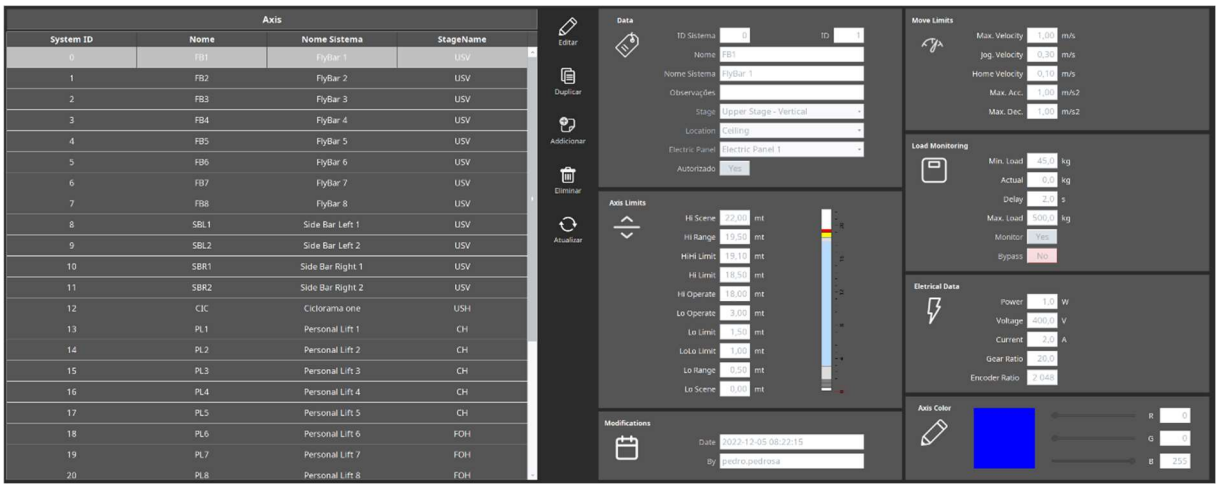


Figura 27 - Vista setup Eixos

Em adição, é também possível configurar os espetáculos existentes. Na Figura 28, podemos analisar o processo configuração.

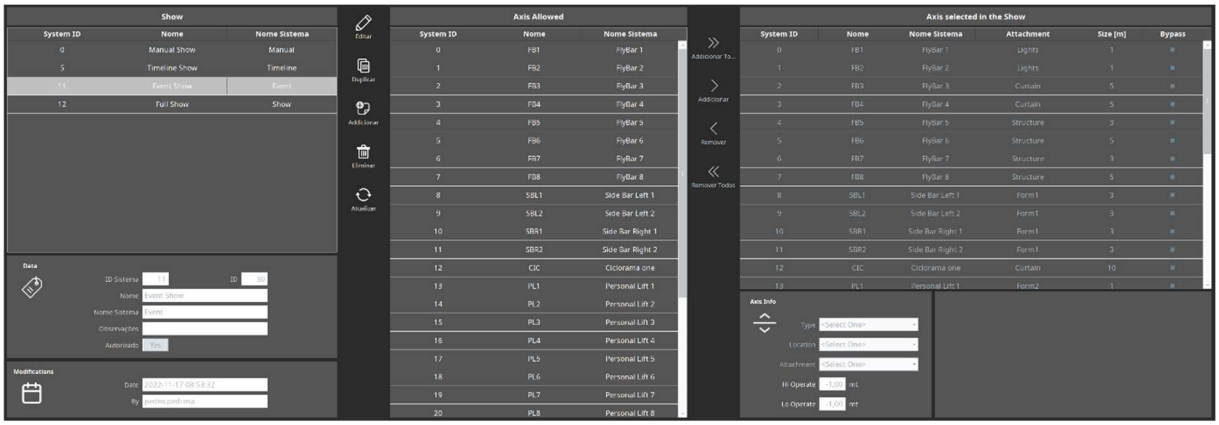


Figura 28 - Vista setup espetáculos

Para cada espetáculo criado, podem ser configurados que motores estão envolvidos, e ainda, se possuem acessório e as suas dimensões. Na figura 29 pode-se observar essa configuração. É dada a opção ao utilizador que tipo de acessório (luzes, cortinas, cenários, estruturas, etc.) quer associar ao motor que está a ser configurado, e a sua altura. Além disso, existe uma opção para fazer um *bypass* a este componente. Caso o *bypass* esteja ativo, opção de *default*, o movimento não terá em conta a dimensão do acessório e poderá deslocar-se até ao seus limites operacionais previamente configurados. Se o *bypass* estiver desligado, o tamanho do acessório passará a ser considerado na movimentação.

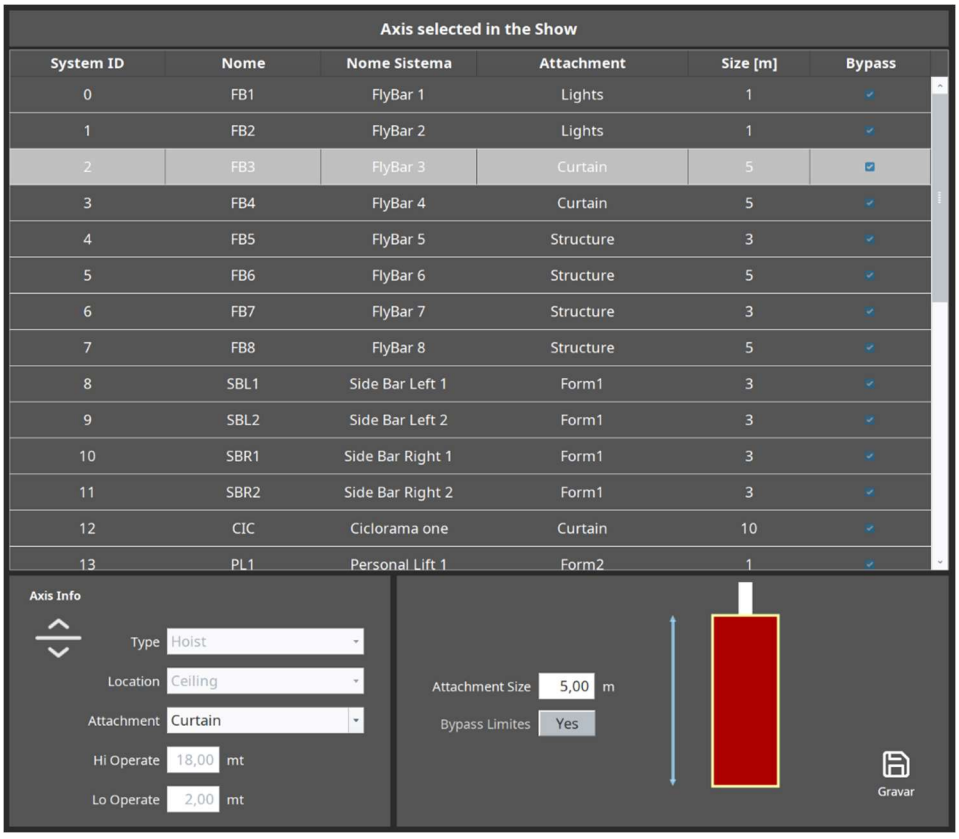


Figura 29 - Configuração acessórios por eixo

Em reuniões com possíveis clientes da aplicação, que possuem o *know-how* neste tópico, e do conhecimento existente na empresa de projetos anteriores surgiu também, no processo de desenvolvimento, a ideia de pode criar listas de motores, para facilitar a seleção de motores, e de poder agrupar motores para movimentos mais específicos. A Figura 30 ilustra a janela de configuração das listas e a Figura 31, a janela de configuração de grupos.

Na janela da configuração das listas, o processo é bastante simples. O operador apenas necessita de criar uma lista nova, ou editar um já existente e adicionar ou remover motores do modo que achar mais conveniente para a sua utilização.

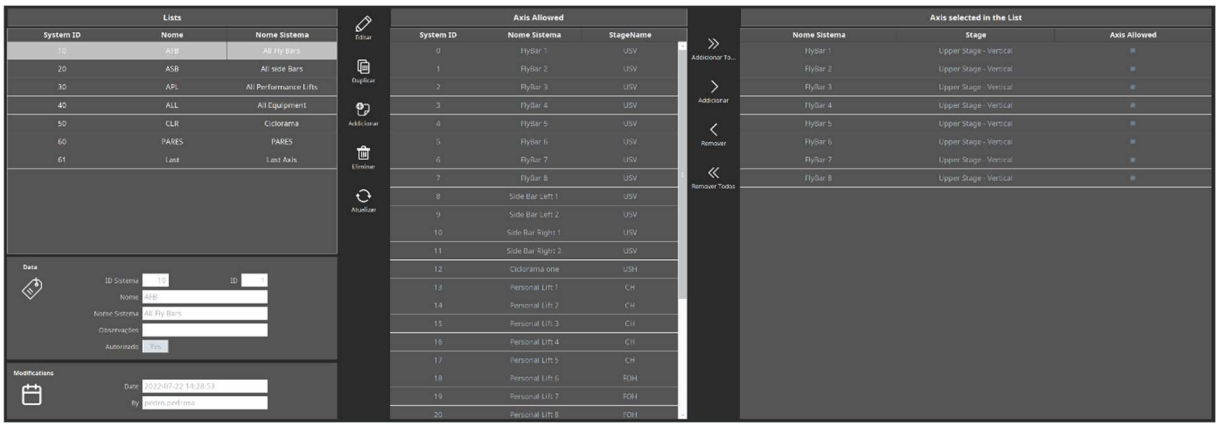


Figura 30 - Vista setup listas

O processo para a janela de grupos, já apresenta algumas alterações. No processo da configuração de um grupo, é necessário identificar que tipo de grupo quer, se síncrono se assíncrono, e se deseja conectar fisicamente os motores. Se o grupo for síncrono, todos os seus elementos têm de cumprir com o movimento em execução. Caso ocorra um erro num dos motores, todos têm de parar. Este erro pode ser causado por avaria mecânica ou se um dos motores atingir os limites operacionais ou fins de curso. No caso dos grupos assíncronos, isto já não se verifica. Desde que não seja o mestre em falha, o movimento do grupo pode continuar, contudo o erro terá de ser corrigido posteriormente.

Além disso, ao escolher os motores pertencentes a um grupo, é necessário escolher qual o mestre. Os restantes motores irão assumir o papel de escravos, seguindo assim a movimentação do mestre.

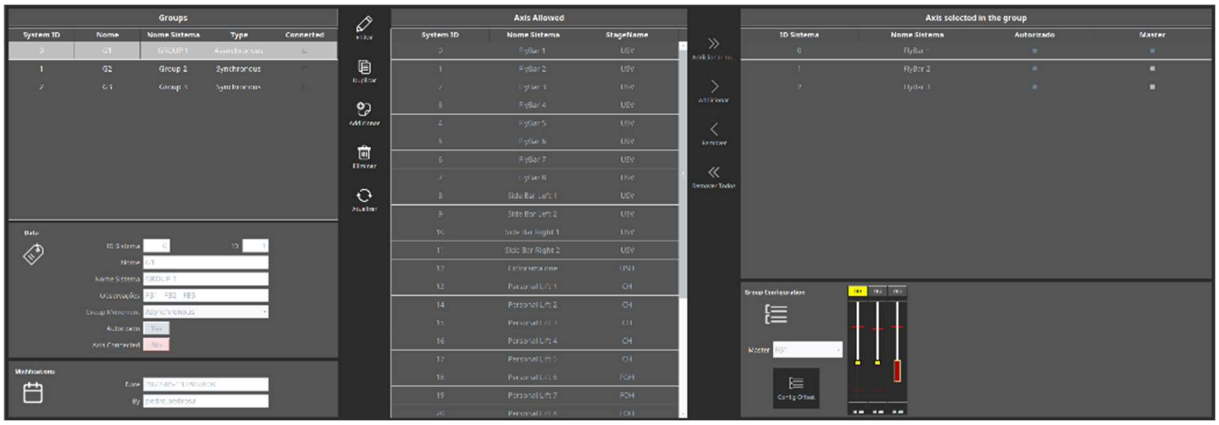


Figura 31 - Vista setup grupos

Adicionalmente, na página de configuração dos grupos, existe também uma opção onde é possível movimentar individualmente cada motor para uma posição pretendida. Esta operação acontece num *popup*, visível na Figura 32. Assim, o grupo será formado com a posição desejada pelo operador sem ter de sair desta página, agilizando todo o processo.

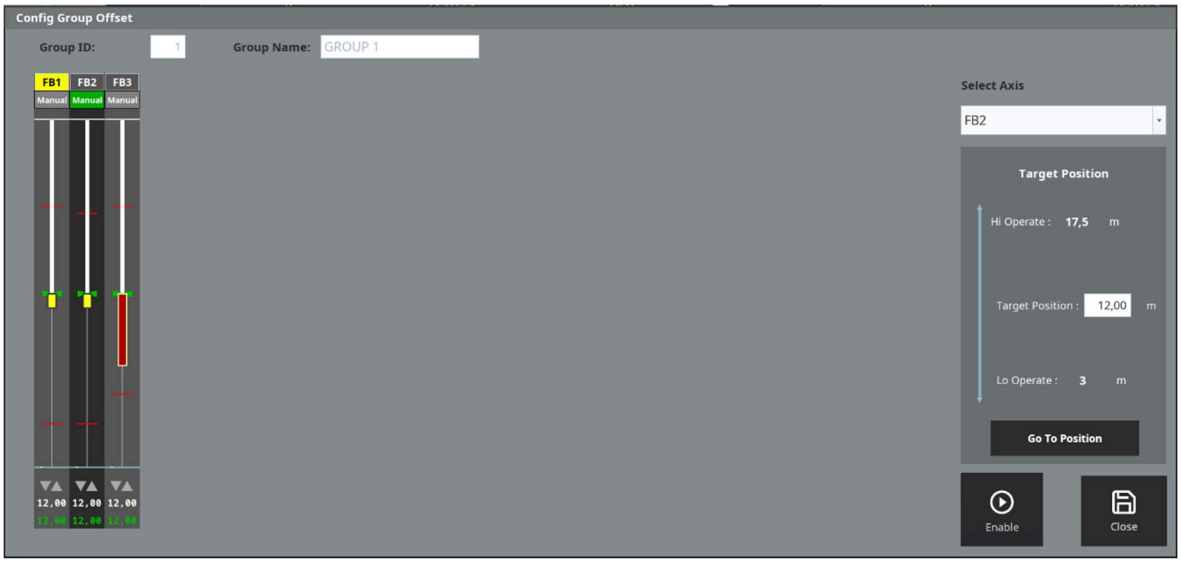


Figura 32 - Configuração de grupos

4.3.3 - Janelas de ferramentas

Como qualquer aplicação desta dimensão, é necessário facultar ao utilizador informação adicional para uma melhor experiência. Para tal foram criadas duas páginas que resolvem esta situação.

Na Figura 33, encontra-se exposto o *layout* da página *'Network'*. Esta página possui dois modos, um designado a vista da Network usada na aplicação em questão, e outro designado a vista topológica da aplicação. Ambos os esquemas expostos são retirados do *TIA Portal* e retratam o estado da interligação entre o PLC e os motores usados.

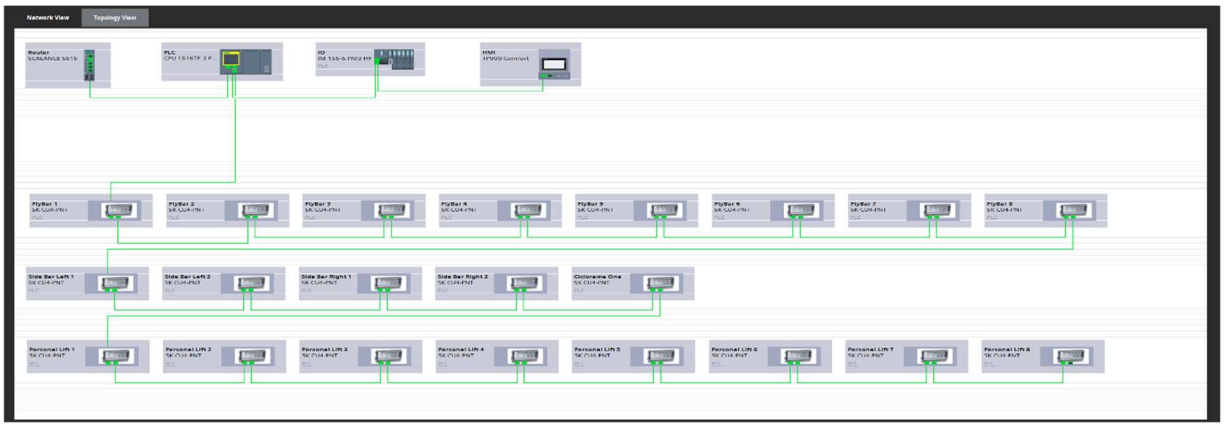


Figura 33 - Vista página Network

Foi também criada uma página, como mostra a Figura 34 designada para armazenar toda a documentação necessária. De momento, existem três documentos que podem ser acedidos: os esquemas elétricos usados, o manual de funcionamento da aplicação e o manual dos variadores usados.

Figura 34 - Vista página de documentação

O *Igntion* possui uma ferramenta para visualização de documentos PDF, sendo que apenas é necessário nas definições do componente indicar o ficheiro que desejamos visualizar.

4.3.4 - Janelas de operação

Com maior destaque da aplicação, foram criadas páginas que permitem levar a cabo o propósito desta aplicação que é comandar e controlar equipamentos do palco usado em teatros e óperas. Assim, foram criadas três páginas: Uma para o modo Manual, outra para o configurar e testar o modo Automático e uma para operar o modo automático, apelidada de modo Live.

4.3.4.1 – Modo Manual

O modo Manual foi criado com a finalidade de permitir ao operador desta aplicação movimentar, em qualquer momento, um ou vários motores conforme seja necessário. Esta movimentação pode ser de motores individuais, listas de motores, ou de grupos previamente configurados. Além disso, foi também decidido previamente que a aplicação iria contar com dois *joysticks* físicos para auxiliar a movimentação.

Para tal, chegou-se à estrutura visível na Figura 35. Este modo conta com três secções. A secção central na qual é possível selecionar que motores irão ser movimentados. E duas secções laterias, a secção esquerda que será controlada pelo *joystick* esquerdo e a secção direita que será controlada pelo *joystick* direito. Estas secções apresentam o mesmo funcionamento, contendo o mesmo código, logo apenas será abordada uma delas.

É importante salientar que, conforme se a aplicação estiver em modo de Simulação, os *joysticks* em funcionamento são virtuais, podendo ser comandado através da aplicação que comunica com o simulador. Se estiver em modo de *On-line* os *joysticks* são físicos e encontram-se conectados diretamente ao PLC.

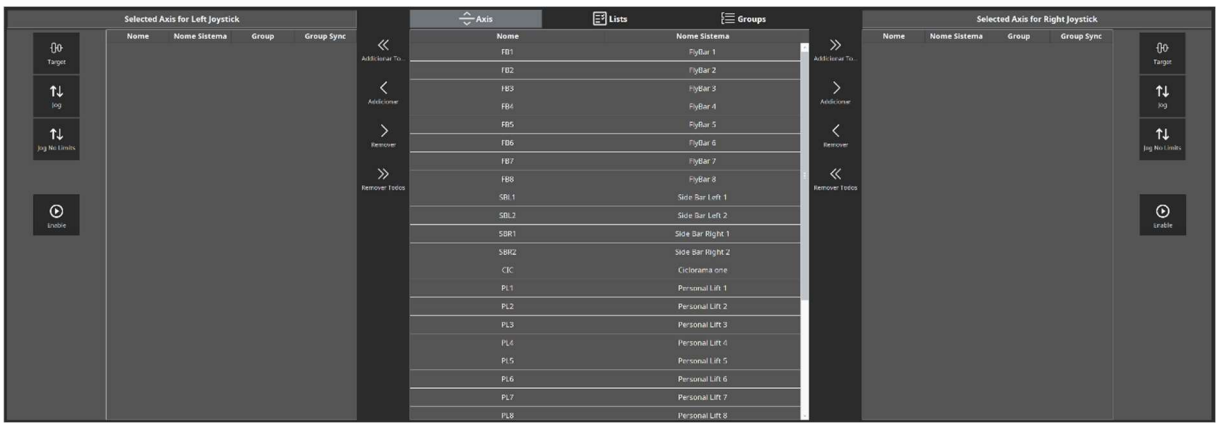


Figura 35 - Vista página modo Manual

Na secção central, o operador pode optar pela vista que lhe é mais útil para a tabela de seleção de eixos. Estas opções são as seguintes: ‘Axis’, que mostra uma tabela contendo todos os eixos pré configurados, que não estejam agrupados; ‘Lists’ que mostra uma tabela das listas de eixos previamente configuradas; ‘Groups’ que expõe no formato de tabela quais os grupos disponíveis para utilização. Todas estas tabelas são obtidas através de um *SELECT* ao SQL.

Após a seleção, existem dois *Tab Strips*, um de cada lado que permite ao operador adicionar ou remover motores das tabelas de movimentação. Esta operação consiste em inserir ou remover dados numa tabela designada à movimentação manual. Esta tabela, possui uma coluna identificativa para distinguir os eixos adicionado à tabela da esquerda e os adicionados à tabela da direita. Os elementos adicionados à esquerda terão um ‘L’ nesta coluna, e os adicionados à direita um ‘R’.

As tabelas visíveis à esquerda e à direita são uma *View* da tabela geral, filtradas pela condição 'L' e 'R', respetivamente.

Tendo escolhido os motores, é necessário ainda escolher o tipo de movimentação. No modo Manual, existem 3 tipos de movimentos possíveis: O modo 'Jog No Limits', o modo 'Jog' e ainda o modo 'Target'.

O modo 'Jog No Limits' retrata um movimento em velocidade. Contudo, apresenta a particularidade de não olhar a limites operacionais do eixo, previamente configurados. Por este motivo, a manobra possui alguns riscos. Como forma de fornecer alguma segurança, esta manobra apenas pode ser realizada para um motor de cada vez. Se o operador tiver mais do que um motor selecionado na tabela, esta opção não está disponível. Como apenas se pode movimentar um motor de cada vez neste modo, a seleção de uma lista ou de um grupo não é valida.

Após selecionar o eixo pretendido para o movimento, o operador necessita de carregar no botão de '*Enable'* e alterar a posição do *joystick* para iniciar o movimento. O botão de '*Enable'* e o *joystick* podem ser virtuais ou físicos, dependendo se estamos em modo de simulação ou *on-line*, respetivamente. Os valores do *joystick* representam valores de velocidade, em m/s, e podem ser positivos ou negativos. Para valores positivos do *joystick*, o motor terá um incremento de posição à velocidade indicada, e para valores negativos o motor terá um decremento de posição à velocidade indicada. A Figura 36, representa todo este processo descrito.

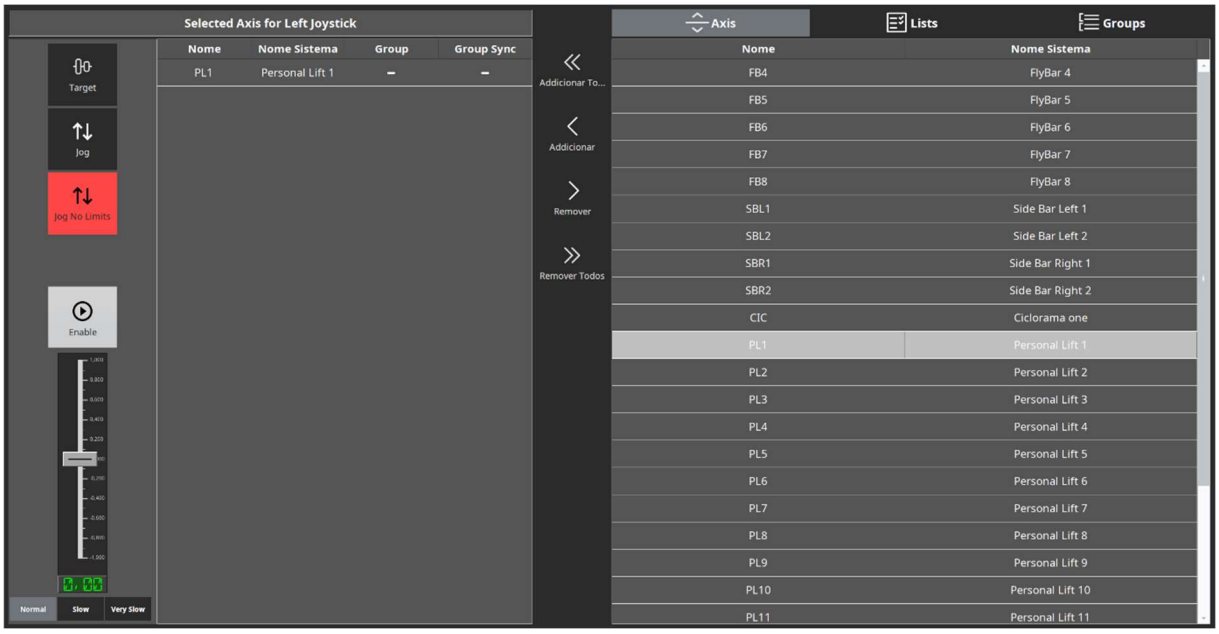


Figura 36 - Movimentação Manual de um eixo no modo 'Jog No Limits'

O modo 'Jog', como o nome indica, também retrata um movimento em velocidade. Contudo, neste modo já não existe limite ao número de motores a movimentar, uma vez que os limites operacionais de cada eixo são tomados em consideração. Assim, é possível ao operador selecionar vários eixos individualmente, selecionar listas de eixos previamente configuradas ou até selecionar grupos.

O funcionamento do '*Enable'* e do *joystick* é igual ao do modo 'Jog No Limits'. Para valores positivos do *joystick*, o motor terá um incremento de posição à velocidade indicada, e para valores negativos o motor terá um decremento de posição à velocidade indicada. No caso de um motor atingir o seu limite operacional, o motor irá perder as condições para continuar o seu movimento. Neste cenário, os motores que ainda tiverem curso para andar irão continuar o seu movimento e o motor que atingiu o seu limite irá ficar parado nessa posição. Se o operador inverter a velocidade do *joystick*, o motor irá afastar-se lo limite operacional em que se encontra. A Figura 37 mostra o processo da movimentação de uma lista de eixos em modo 'Jog'.

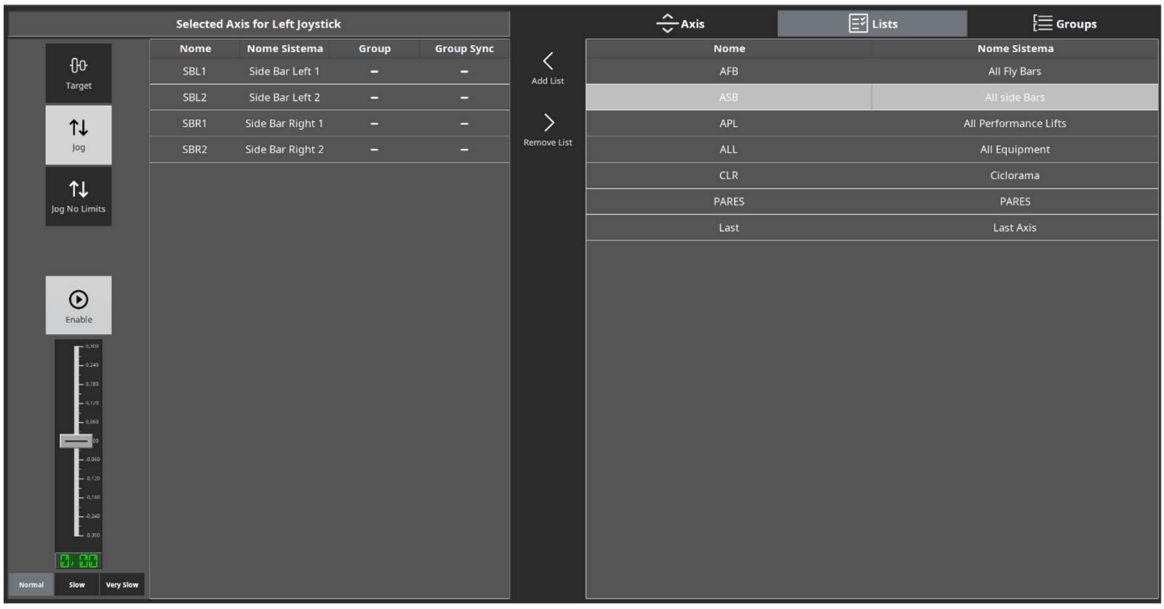


Figura 37 – Movimentação Manual de uma lista de eixos no modo 'Jog'

O modo 'Target', apresenta um funcionamento diferente dos anteriores, e retrata um movimento em posição. O operador necessita de introduzir posição de destino válida, dentro dos limites operacionais, para iniciar o movimento. Este modo é possível de operar com eixos individuais, listas de eixos e grupos. No caso da movimentação de eixos desagrupados, todos os eixos selecionados irão cumprir com a posição destino inserida. No caso dos grupos, o mestre vai para a posição destino e os escravos cumprem o *offset* com que o grupo foi configurado. A introdução da posição destino é forçada a estar dentro dos limites operacionais. Para o caso de existir mais do que um eixo selecionado, os valores dos limites são calculados no *background* e devolvidos para informação do operador.

Após a introdução de uma posição válida, o operador pode pressionar o botão de 'Enable' e alterar o valor da velocidade no *joystick*. A Figura 38 retrata este modo.

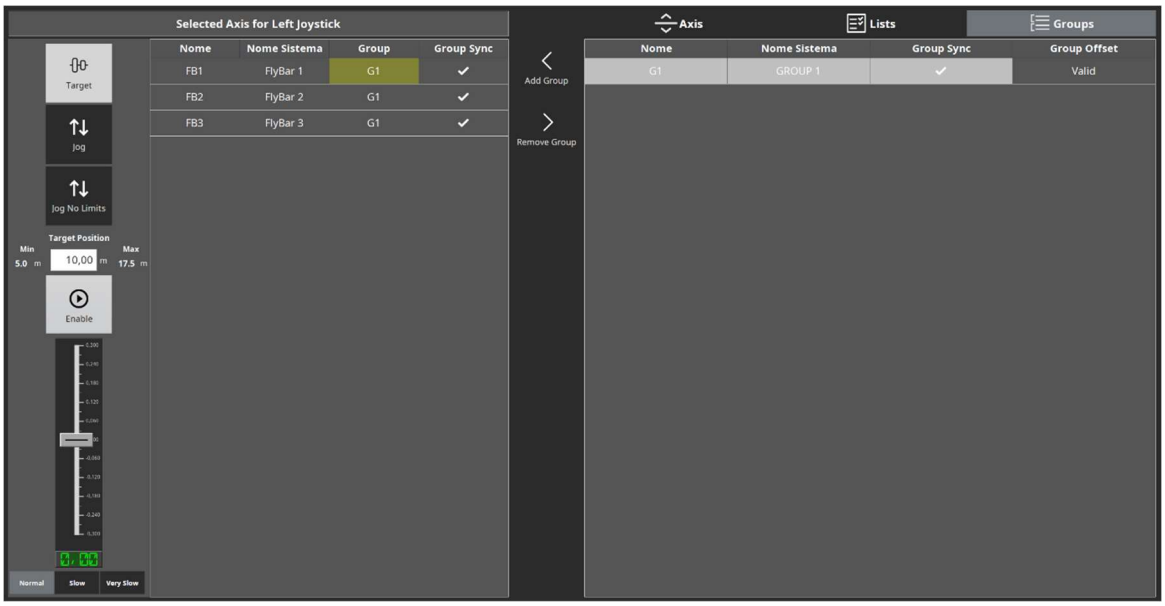


Figura 38 - Movimentação Manual de um grupo no modo 'Target'

Em qualquer um dos modos selecionados pelo operador, o botão de 'Enable' enquanto tiver o seu estado positivo (botão pressionado) ativa os *bits* da *Control Word* necessários para iniciar o movimento, faltando apenas a velocidade do movimento. Conforme a velocidade indicada pelo *joystick* o motor irá cumprir com a manobra desejada.

Em adição, sempre que o *joystick* é operado, é dada a opção de controlar a velocidade do movimento, tendo três opções: A primeira opção 'Normal', definida por *default*, que tem como velocidade máxima disponível a velocidade configurada no *setup* do eixo selecionado. Se forem vários eixos, é a menor velocidade configurada para esses eixos; A opção 'Slow', representa 10% do valor 'Normal'; A opção 'Very Slow', representa 1% do valor 'Normal'. Esta variedade de opção é útil para o operador realizar movimentos precisos, com maior exatidão a uma velocidade mais controlada.

Por motivos de confidencialidade a nível contractual com a entidade onde se realizou o estágio, não possível mostrar o código desenvolvido.

4.3.4.2 - Modo Auto

O modo Automático, surge como um dos principais focos desta aplicação. Nele, é possível criar ações e configurar a movimentação de motores, permitindo assim um maior controlo sobre o cenário do espetáculo criado ao seu operador. O operador pode usar este modo para editar espetáculos assim como testar os seus movimentos. Nesta página, todos os movimentos configurados, são movimentos em posição.

Dada a importancia deste modo, a página criada também reflete a sua maior complexidade. Tentando simplificar ao máximo esta página, chegou-se à estrutura apresentada na Figura 39.

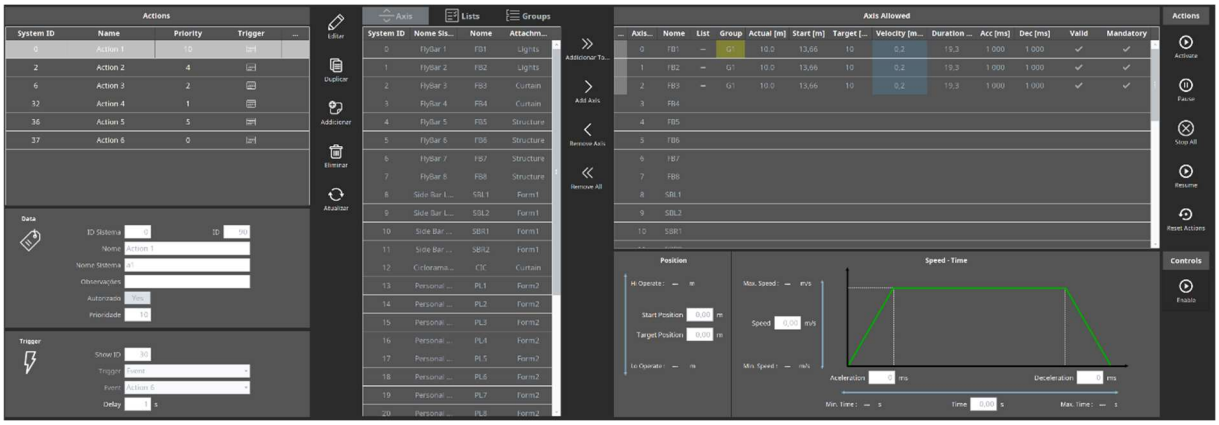


Figura 39 - Vista página modo Automático

Esta página conta com cinco secções, as quais serão descritas em maior detalhe. Estas secções apresentam um encadeamento lógico dos passos que devem ser seguidos desde criar uma ação, passando por configurar os seus movimentos e testar os mesmos.

Em primeiro plano, do lado esquerdo da página, e de modo a manter a coerência que tem vindo a ser aplicada na solução desenvolvida, temos a primeira secção. Esta secção, visível em maior detalhe na Figura 40, apresenta uma tabela identificativa das ações existentes no espetáculo que o operador está a configurar. Além dessa tabela, existem também dois campos onde o utilizador pode ver e editar uma determinada ação. Do lado direito encontra-se um *Tab Strip* que permite criar, editar ou apagar ações para o espetáculo.

Figura 40 – Configuração tabela de ações modo Automático

No primeiro campo de edição, pode-se editar a informação da ação criada, com principal foco para a sua prioridade. O sistema de prioridades foi introduzido como forma de gerir conflito entre múltiplas ação a decorrer ao mesmo tempo que utilizem os mesmos eixos. Se duas ações forem executadas ao mesmo tempo e os eixos usados não coincidam, ambas podem realizar o seu percurso normalmente; caso ambas as ações necessitem do mesmo eixo, apenas a ação com maior prioridade será executada. A ação com prioridade menor ficará à espera de que se reúnam condições para poder entrar em execução. Se, por erro de operação, ambas as ações tiverem uma prioridade igual, nenhuma delas será executada, e ambas ficarão *on hold*.

O segundo campo de edição permite configurar o tipo de *trigger* da ação. O *trigger* funciona como modo de ativação, das quais existem três opções: *Manual*, *Event* e *Timeline*. Conforme a opção escolhida pelo operador, o formato deste campo de edição pode sofrer alterações.

Para um *trigger manual*, o operador não necessita de configurar mais nada para a ação, como mostra a Figura 41.

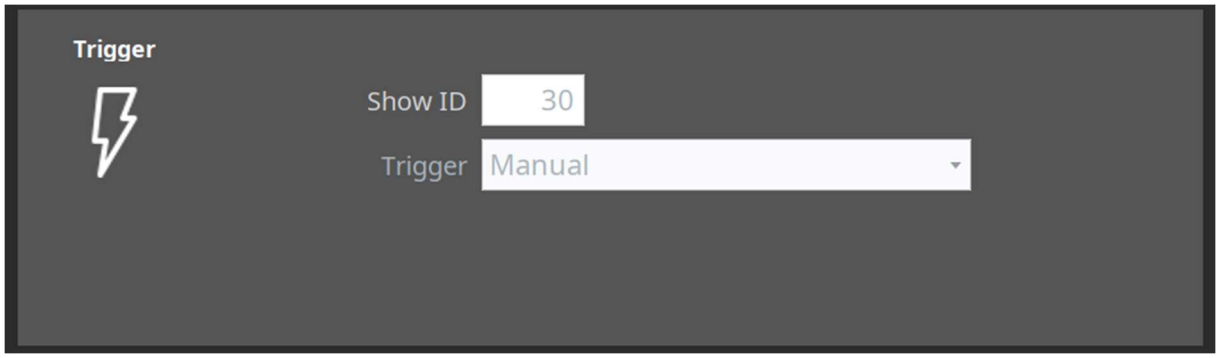


Figura 41 – Trigger Manual

Para um *trigger event,* é necessário escolher que ação precede a ação a ser configurada e pode também ser definido um *delay* entre a conclusão de uma ação e o início da nova ação, como mostra a Figura 42.

Figura 42 – Trigger Event

Para um *trigger timeline*, o operador necessita de introduzir a que tempo, no formato HH:mm:ss, deseja iniciar a ação.

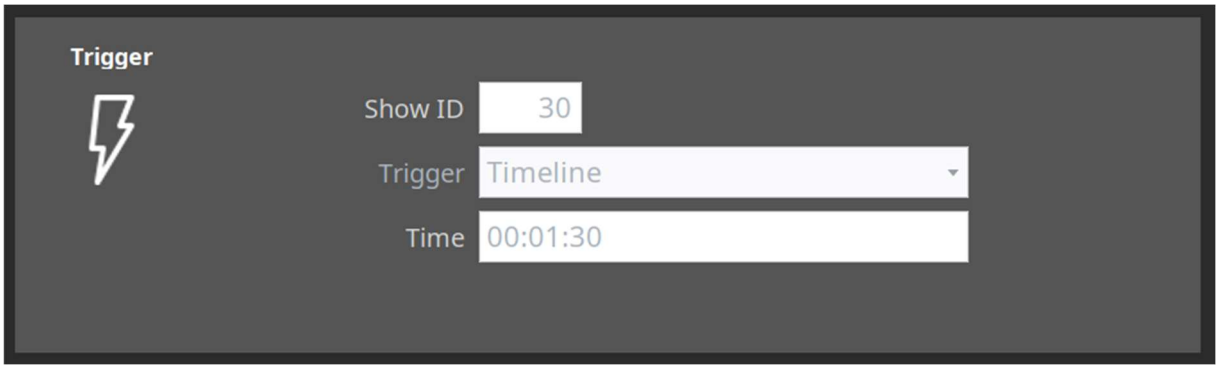


Figura 43 – Trigger Timeline

Como segunda secção desta página temos, como ilustra a Figura 44, uma zona onde é possível adicionar ou remover eixos individualmente, listas de eixos ou grupos à ação que está a ser configurada. Deste modo, o operador pode personalizar cada ação do modo mais conveniente ao adicionar ou remover eixos a cada ação do seu espetáculo. Esta tabela possui também uma coluna contendo a informação sobre que acessório possui cada eixo, fornecendo assim mais informação ao operador. Para aceder a informação de várias tabelas, foi utilizada a função *INNER JOIN* para realizar o *query* ao SQL.

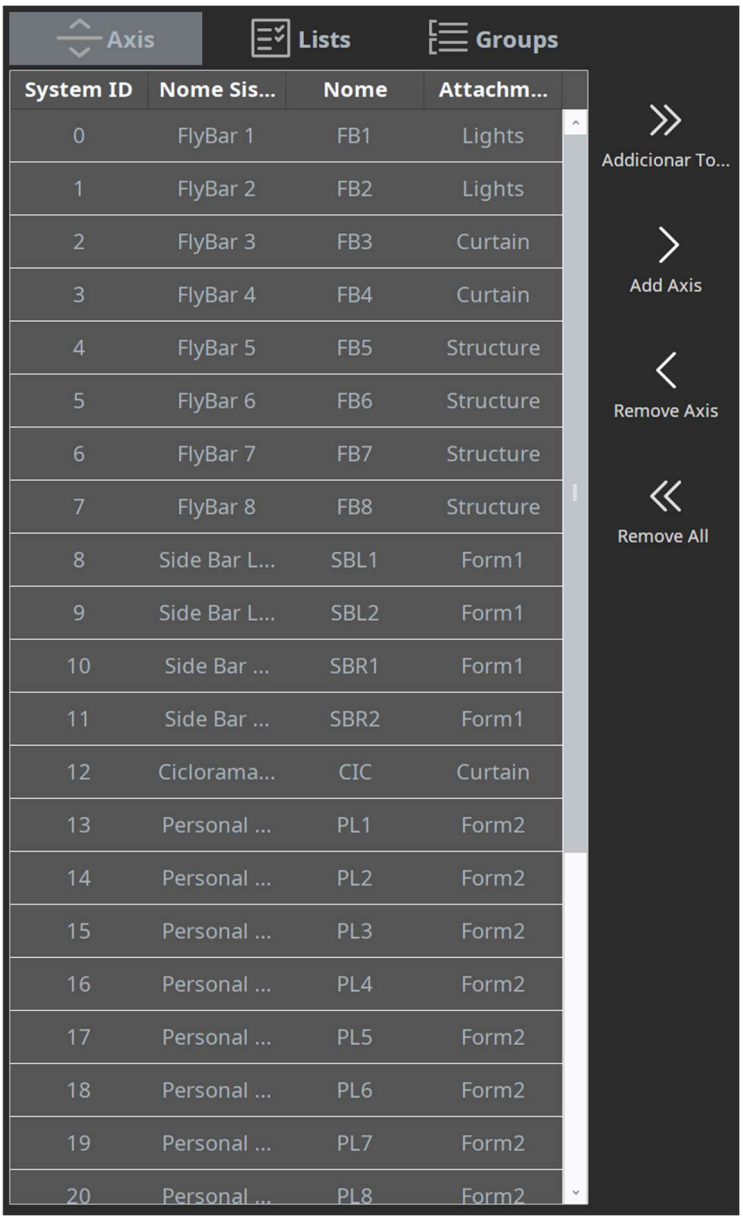


Figura 44 – Tabela eixos modo Automático

Como terceira secção, temos onde podemos visualizar que eixos foram adicionados à ação e os detalhes da sua movimentação. Esta tabela contém informação individual sobre a posição atual, a posição onde se inicia o movimento, a posição de destino, a velocidade de movimentação, a duração do movimento, o tempo de aceleração e o tempo de desaceleração, se o movimento já foi validado e ainda se o movimento é obrigatório.

Contudo, quando um eixo é adicionado a uma ação, todos estes valores não estão configurados, e o movimento ainda não foi validado. Para tal, o operador necessita de selecionar, individualmente, o eixo que quer configurar, editar os seus valores numa zona de configuração que se apresenta debaixo da tabela dos eixos e validar o movimento.

A figura 45 mostra, a secção quatro e secção cinco, na parte superior e inferior da imagem, respetivamente.

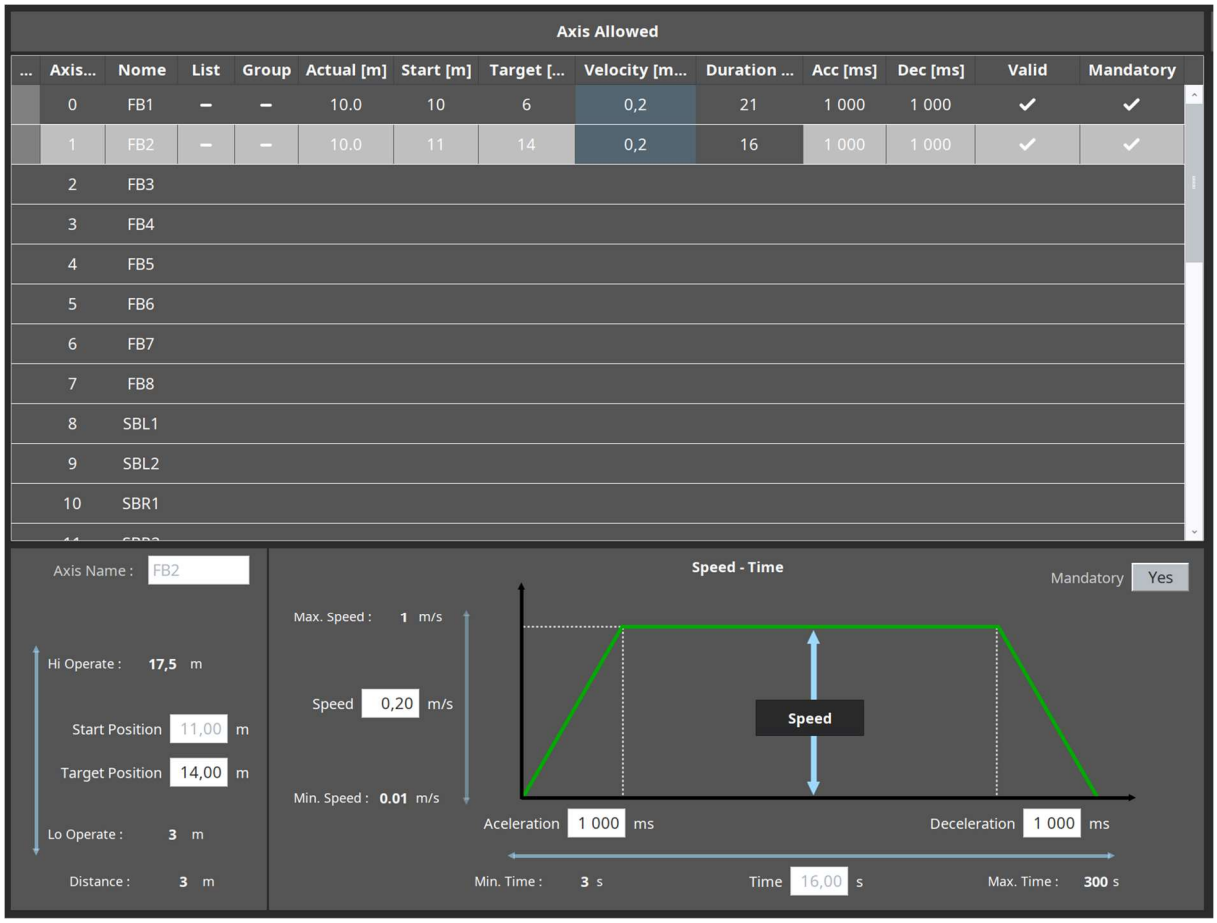


Figura 45 – Detalhes sobre informação do movimento de cada eixo no modo Automático

Na zona de configuração de cada movimento (secção cinco), o operador pode configurar o movimento que o motor irá realizar. Para tal, há certos parâmetros que têm de ser preenchidos.

O primeiro é a posição de destino, sendo que a posição onde se inicia o movimento é a posição em que se encontra o motor quando o mesmo é adicionado à ação, o operador apenas necessita de introduzir um valor de destino, valido dentro dos limites operacionais previamente configurados. O valor da distância é calculado automaticamente.

De seguida o operador necessita de escolher se deseja realizar o seu movimento em função da velocidade, ou em função do tempo. Se optar por configurar em relação à velocidade, a zona de configuração terá o *layout* da Figura 46, se optar por configurar em relação ao tempo, terá o *layout* da Figura 47. Contudo, para ambos os casos é necessário preencher o valor do tempo de aceleração, em milissegundos, e o valor do tempo de desaceleração, em milissegundos.

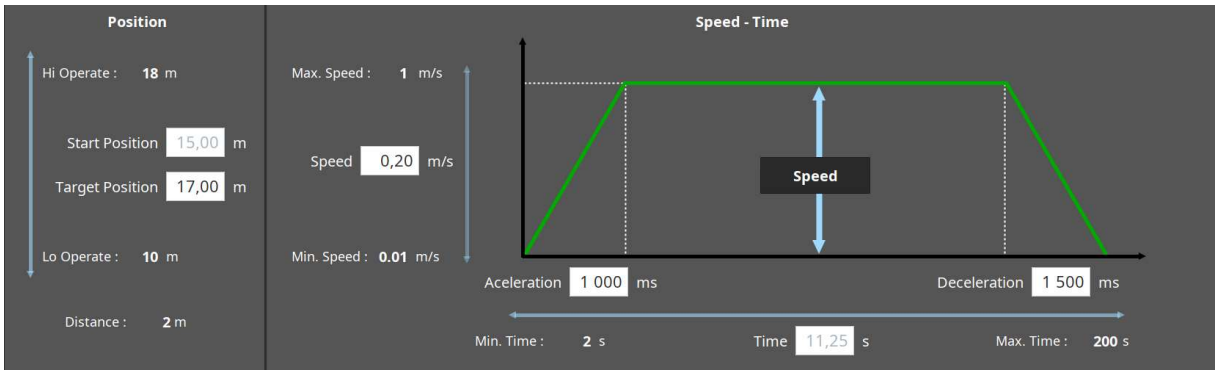


Figura 46 – Configuração modo Automático (Velocidade)

Quando um movimento é configurado em relação à velocidade, o operador já tendo preenchido o campo referente à posição de destino, determinando assim a distância do movimento, e os valores dos tempos de aceleração e desaceleração, resta apenas inserir o valor da velocidade requerida. Este valor tem de ser inferior à velocidade máxima permitida pelo motor.

Foram utilizadas para os cálculos as seguintes variáveis:

$$d = \text{distância } (m)$$

$$x1 = tempo\ de \text{ aceleração (s)}$$

$$x2 = \frac{d}{velocidade\ máxima} - \frac{x1}{2} - \frac{x3}{2}$$

$$x3 = \ tempo\ de \text{ desaceleração (s)}$$

Sabendo todos estes valores, é possível determinar o tempo de movimento (t), através da seguinte formula:

$$t = x1 + x2 + x3\ (s)$$

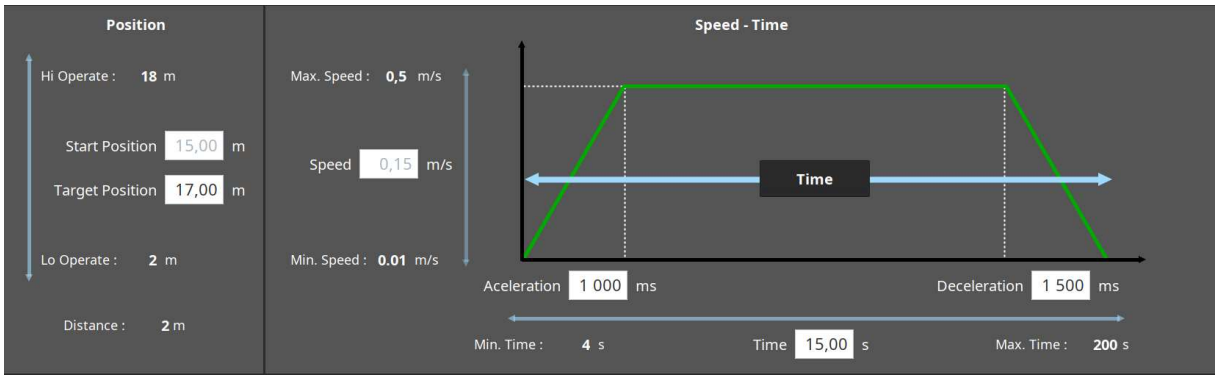


Figura 47 – Configuração modo Automático (Tempo)

Quando um movimento é configurado em relação ao tempo de movimento total, já tendo preenchido o campo referente à posição de destino, determinando assim a distância do movimento, e os valores dos tempos de aceleração e desaceleração, resta apenas preencher o valor do tempo do movimento. É importante salientar, que

como retrata a Figura 47, o tempo total de movimento engloba o tempo de aceleração, o tempo à velocidade máxima e o tempo de desaceleração. Este valor tem de estar contido entre o tempo de duração mínimo e máximo que o motor consegue executar por limitação da velocidade máxima do motor.

Foram utilizadas para os cálculos as seguintes variáveis:

$$d = \text{distância } (m)$$

$$x1 = tempo\ de\ \text{aceleração } (s)$$

$$x2' = tempo\ total\ do\ movimento - tempo\ de\ desaceleração\ (s)$$

$$x3' = \ tempo\ total\ do\ movimento\ (s)$$

Sabendo todos estes valores, é possível determinar a velocidade máxima de movimento (v), através da seguinte formula:

$$v = \frac{\mathrm{d}}{\frac{x1}{2} + (x2' \ - \ x1) + \frac{x3' - x2'}{2}} \ (m/s)$$

Contudo, esta página além de ter a funcionalidade editar e criar ações e o movimento dos seus eixos, também dá a possibilidade de teste no processo de desenvolvimento, sem ter de sair da página. Para tal, foi implementada, do lado direito da página, uma secção designada a comandar estas ações. Foram criados um *Tab Strips* e um botão de 'Enable', que apenas é visível no modo de simulação. Em modo *On-line*, o botão físico do *joystick* substituirá o botão virtual.

A primeira *Tab Strip*, tem a função de comandar as ações, e possui cinco opções.

- A opção de 'Run' na qual o operador pode selecionar uma ação, independentemente do seu *trigger* (não necessita de ser uma ação com *trigger* Manual), e todos os eixos pertencentes a esta ação irão realizar os seus movimentos configurados. Após todos os movimentos serem concluídos a ação é dada por concluída;
- A opção 'Pause', onde o operador pode colocar em pausa uma ação selecionada. Todos os eixos pertencentes a esta ação irão ficar em pausa.
- A opção 'StopAll', coloca as ações, que estão em movimento, em pausa;
- A opção 'Resume', retoma todas as ações que estiverem em pausa;
- A opção 'Clean', limpa o estado de todas as ações, voltando todas ao seu estado inicial.

Por motivos de confidencialidade a nível contractual com a entidade onde se realizou o estágio, não possível mostrar o código desenvolvido.

4.3.4.3 - Modo Live

O modo ‘Live’ é uma versão mais simples e pragmática, do ponto de vista de operação, do modo Automático, já mencionado. Nesta página, visível na Figura 48 é possível visualizar e comandar todas as ações previamente configuradas, para um determinado espetáculo.

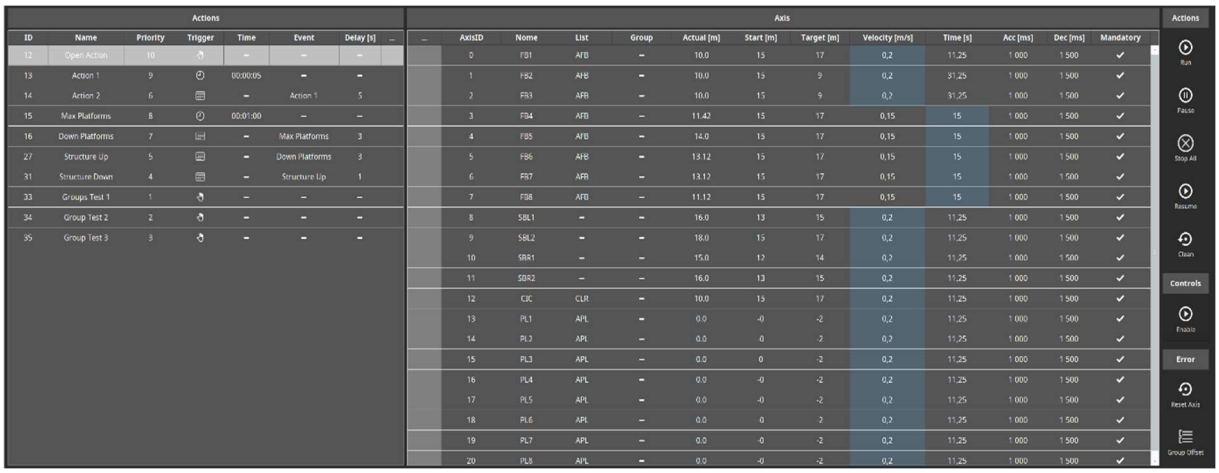
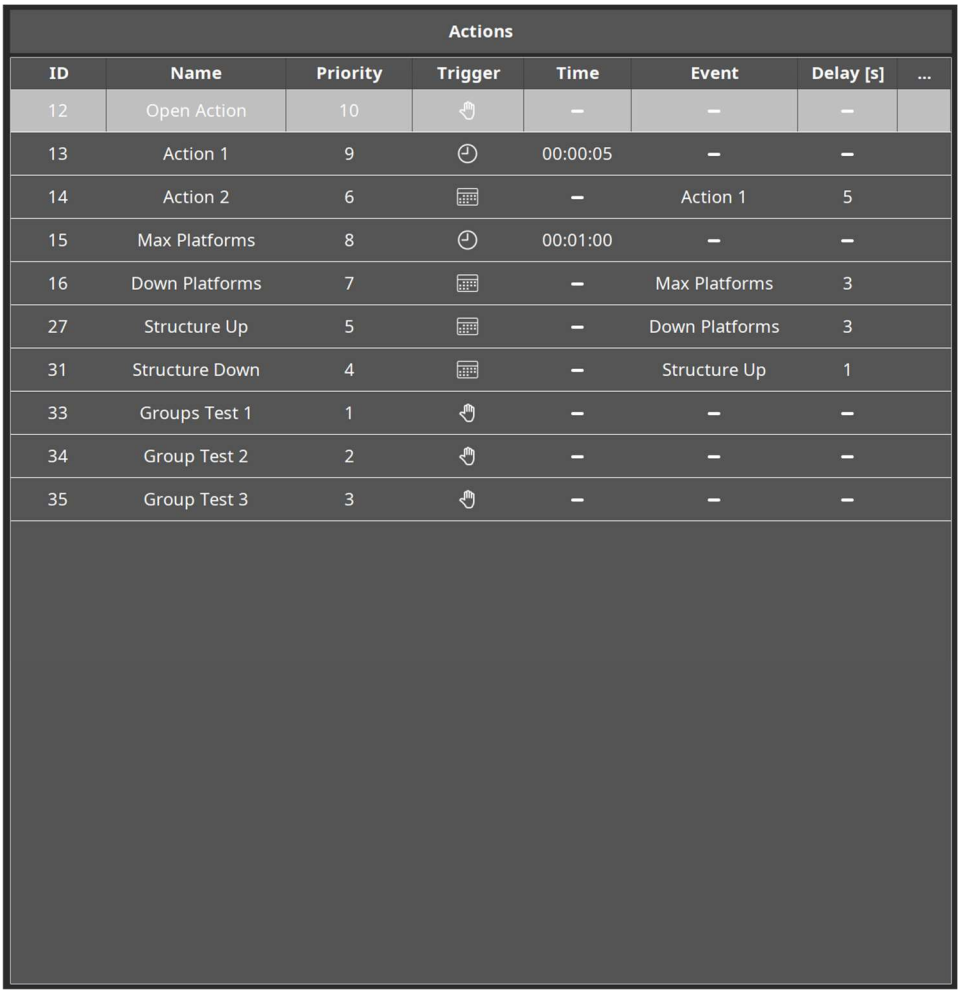


Figura 48 - Vista página ‘Live’

Para uma experiência mais intuitiva para o operador, a página criada encontra-se dividida em três secções: uma tabela contendo informação sobre as várias ações disponíveis, uma tabela contendo informação sobre os eixos utilizados na ação escolhida pelo operador, na tabela anterior, e uma secção designada comandos de operação.

A primeira tabela, detalhada na Figura 49, apresenta todas as ações configuradas para um determinado espetáculo. Esta tabela é obtida através de um *SELECT* ao SQL. O operador pode analisar o grau de prioridade de cada ação, qual o seu *trigger*, e os detalhes do seu *trigger*, caso existam. Nesta tabela, para ser visualmente mais intuitivo para o utilizador, são utilizados ícones indicativos. Para ação com um *trigger* manual, é utilizado o ícone de uma mão, para um *trigger* que é atuado a um determinado tempo é utilizado o ícone de um relógio e para o *trigger* de evento que é atuado após uma ação ter concluído a sua movimentação, é usado o ícone de um calendário. Além disso, existe uma coluna designada ao estado de cada ação, sendo animada com um código de cores.



Actions

ID	Name	Priority	Trigger	Time	Event	Delay [s]	...
12	Open Action	10		–	–	–	
13	Action 1	9		00:00:05	–	–	
14	Action 2	6		–	Action 1	5	
15	Max Platforms	8		00:01:00	–	–	
16	Down Platforms	7		–	Max Platforms	3	
27	Structure Up	5		–	Down Platforms	3	
31	Structure Down	4		–	Structure Up	1	
33	Groups Test 1	1		–	–	–	
34	Group Test 2	2		–	–	–	
35	Group Test 3	3		–	–	–	

Figura 49 – Tabela de ações modo Live

A Figura 50, mostra a tabela que constitui a segunda secção desta página. Nesta tabela é possível ver com maior detalhe todos os eixos que estão configurados na ação selecionada na tabela das ações.

Para cada eixo, é possível observar se pertence a um grupo, qual a posição em que se encontra, a posição inicial do seu movimento, a sua posição de destino, a velocidade do seu movimento, o tempo de execução do movimento, o tempo de aceleração, o tempo de desaceleração e se o movimento é obrigatório. Além disso, e mantendo a uniformidade com a tabela anterior, existe uma coluna designada, identificar o estado de cada motor.

Axis

...	AxisID	Nome	List	Group	Actual [m]	Start [m]	Target [m]	Velocity [m/s]	Time [s]	Acc [ms]	Dec [ms]	Mandatory
	0	FB1	AFB	–	10.0	15	17	0,2	11,25	1 000	1 500	✓
	1	FB2	AFB	–	10.0	15	9	0,2	31,25	1 000	1 500	✓
	2	FB3	AFB	–	10.0	15	9	0,2	31,25	1 000	1 500	✓
	3	FB4	AFB	–	11.42	15	17	0,15	15	1 000	1 500	✓
	4	FB5	AFB	–	14.0	15	17	0,15	15	1 000	1 500	✓
	5	FB6	AFB	–	13.12	15	17	0,15	15	1 000	1 500	✓
	6	FB7	AFB	–	13.12	15	17	0,15	15	1 000	1 500	✓
	7	FB8	AFB	–	11.12	15	17	0,15	15	1 000	1 500	✓
	8	SBL1	–	–	16.0	13	15	0,2	11,25	1 000	1 500	✓
	9	SBL2	–	–	18.0	15	17	0,2	11,25	1 000	1 500	✓
	10	SBR1	–	–	15.0	12	14	0,2	11,25	1 000	1 500	✓
	11	SBR2	–	–	16.0	13	15	0,2	11,25	1 000	1 500	✓
	12	CIC	CLR	–	10.0	15	17	0,2	11,25	1 000	1 500	✓
	13	PL1	APL	–	0.0	-0	-2	0,2	11,25	1 000	1 500	✓
	14	PL2	APL	–	0.0	-0	-2	0,2	11,25	1 000	1 500	✓
	15	PL3	APL	–	0.0	0	-2	0,2	11,25	1 000	1 500	✓
	16	PL4	APL	–	0.0	-0	-2	0,2	11,25	1 000	1 500	✓
	17	PL5	APL	–	0.0	-0	-2	0,2	11,25	1 000	1 500	✓
	18	PL6	APL	–	0.0	-0	-2	0,2	11,25	1 000	1 500	✓
	19	PL7	APL	–	0.0	-0	-2	0,2	11,25	1 000	1 500	✓
	20	PL8	APL	–	0.0	-0	-2	0,2	11,25	1 000	1 500	✓

Figura 50 – Tabela eixos modo Live

Para a coluna que retrata o estado das ações e dos motores, referida nas duas tabelas anteriores, optou-se pelo seguinte funcionamento: Quando uma ação não tem nenhum comando de movimento, o estado terá uma cor cinzenta, se a ação estiver a ser executada, terá uma cor amarela, se a ação for colocada em pausa pelo operador, terá uma cor laranja, se a ação estiver em espera para iniciar o movimento, terá uma cor amarela e laranja intermitente, se houver um erro terá a cor vermelha e quando a ação é concluída tem a cor verde.

Como última seção da página, existe do lado direito da página, uma zona designada a comandar estas ações, ilustrada na Figura 51. Foram criados dois *Tab Strips* e um botão de 'Enable', que apenas é visível no modo de simulação. Em modo *On-line*, o botão físico do *joystick* substituirá o botão virtual.

Figura 51 - Comandos modo Live

A primeira *Tab Strip*, tem a função de comandar as ações, e possui cinco opções.

- A opção de 'Run' na qual o operador pode selecionar uma ação, independentemente do seu *trigger* (não necessita de ser uma ação com *trigger* Manual), e todos os eixos pertencentes a esta ação irão realizar os seus movimentos configurados. Após todos os movimentos serem concluídos a ação é dada por concluída;
- A opção 'Pause', onde o operador pode colocar em pausa uma ação selecionada. Todos os eixos pertencentes a esta ação irão ficar em pausa;
- A opção 'StopAll', coloca as ações, que estão em movimento, em pausa;
- A opção 'Resume', retoma todas as ações que estiverem em pausa;
- A opção 'Clean', limpa o estado de todas as ações, voltando todas ao seu estado inicial.

O botão de 'Enable' como nos casos anteriores, enquanto ativo, ativa os *bits* necessários para iniciar a movimentação, faltando apenas o valor de velocidade que é dado pelo *joystick*.

A segunda *Tab Strip*, contem apenas duas opções, nas quais é possível realizar o *reset* a um eixo e corrigir o *offset* de um grupo. Estas opções são de correção e apenas podem ser realizadas caso ocorra um erro num eixo, ou no *offset* de um grupo, respetivamente.

Por motivos de confidencialidade a nível contractual com a entidade onde se realizou o estágio, não possível mostrar o código desenvolvido.

Por último, temos representado na Figura 52 o *popup* onde é possível sair da aplicação. Esta operação necessita da confirmação do seu operador antes de ser executada.

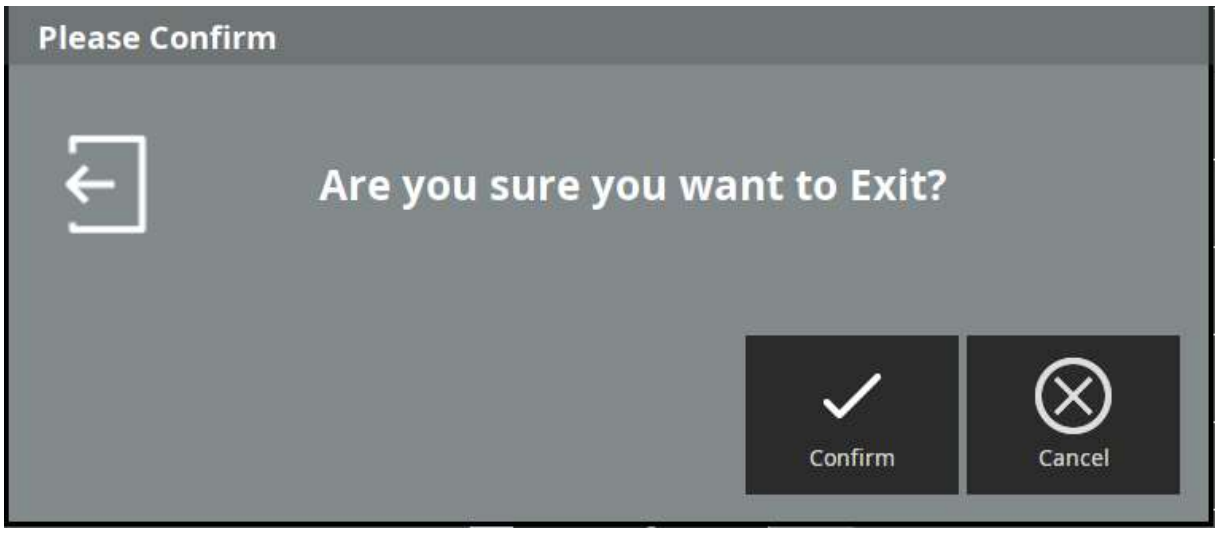


Figura 52 - Pop-up sair da aplicação

5. Conclusões e trabalho futuro

Este capítulo tem como finalidade realizar um balanço do trabalho realizado, retirando as suas conclusões, e analisar que perspetivas de futuro existem para a aplicação desenvolvida.

5.1 – Conclusões

Os softwares de SCADA têm uma forte presença no mercado mundial e podem ajudar a desenvolver as mais diversas aplicações. Foi, através deste tipo de software, mais concretamente o *Ignition*, que foi desenvolvida a aplicação, para gerir e controlar equipamentos de palco, que fundamenta este relatório.

Como referido anteriormente, o objetivo desta aplicação seria permitir ao seu operador configurar os vários equipamentos necessários para um espetáculo, possibilitar a operação destes mesmos equipamentos em vários modos (manual e automático) e ainda permitir também de forma intuitiva, informativa e funcional de analisar estes mesmos dados. Além disso, foi também proposta a opção de funcionar em modo de simulação, na qual o operador pode testar todos os movimentos sem ter de operar os motores físicos, e de ter o modo de funcionamento normal (*On-line*), na qual a aplicação comunica com o PLC e efetua a movimentação dos motores físicos.

Após a conclusão do presente relatório, foi possível analisar que todos os objetivos propostos inicialmente para o estágio foram cumpridos, sendo que ainda se iniciou o processo de planeamento da comunicação, como o PLC e com os equipamentos de campo.

Deste modo, considero que o estágio curricular foi realizado com sucesso. Não só pelas componentes profissionais adquiridas, mas também a nível pessoal, pela oportunidade que tive de trabalhar diariamente com um grupo tão dedicado e profissional que me ajudaram em todo o processo de adaptação.

As valências académicas adquiridas no período de estágio serão bastante úteis para o futuro, uma vez que foi possível ter contacto com diversas partes do processo de desenvolvimento de uma solução para a indústria, conferindo assim uma vista geral mais esclarecedora de todo o procedimento.

5.2 – Trabalho futuro

Durante o período do estágio curricular foi feito o planeamento da aplicação e foi iniciado o seu processo de desenvolvimento. Além disso, a aplicação desenvolvida tem como finalidade corresponder às necessidades de futuros cliente. Essas necessidades podem levar a futuras alterações na aplicação.

Estas alterações podem estar fundamentadas na secção de *Setup*, nomeadamente na configuração de grupos, como também podem ser no modo de operação. Enquanto a configuração e movimentação de motores isolados apresenta um funcionamento fiável e robusto, a movimentação de vários motores em grupo pode criar algum conflito entre os mesmos, no caso de falha de comunicação ou avaria mecânica. Estas questões terão de ser abordadas futuramente de modo a corrigir da melhor forma estas avarias sempre que necessário. As alterações podem ser significativas, não só a nível estético, como também do ponto de vista de código. Poderá ser necessário corrigir a estrutura de tabelas no SQL, criando colunas e adaptando o código de modo a contornar estes problemas.

Além disso, como abordado no presente relatório, a aplicação tem como objetivo a dois modos, o de 'Simulação' e o '*On-line*'. Enquanto a componente de simulação ficou otimizada, não foi possível a programação do PLC em tempo útil de estágio. Deste modo, uma das etapas seguintes a cumprir é a programação do autómato. A comunicação com o PLC será feita por OPC-UA. Uma das valências do *Ignition* é a facilidade de configuração o tipo de comunicação com PLC. Esta programação será feita com o software TIA *Portal* V17 da *Siemens*.

Após estas alterações, estarem concluídas e a aplicação seja testada de modo a garantir a sua fiabilidade, terá como finalidade a venda a clientes. Neste tipo de aplicações, a SA passa por um processo de acompanhamento e suporte às aplicações desenvolvidas. Durante esse período de testes ou mesmo de suporte, pode ser necessário efetuar alterações de modo a corrigir eventuais bugs, ou até adicionar novas funcionalidades que sejam proveitosas.

REFERÊNCIAS

[1] Automação Industrial (2022). *http://www.automacaoindustrial.info/o-que-sao-sistemas-supervisorios,* acedido em 12 de novembro de 2022

[2] Ujvarosi, Alexandru. 2016. "EVOLUTION OF SCADA SYSTEMS"

[3] Daneels, A, e W Salter. 1999. "What is Scada?". in International Conference on Accelerator and Large Experimental Physics Control Systems, 1999, Trieste, Italy*, https://cds.cern.ch/record/532624/files/mc1i01.pdf,* acedido em 12 de novembro de 2022

[4] SA Automação, Aveva Edge (2022). *http://www.sa.online.pt/pt/artigos/info/aveva-edge_47/,* acedido em 19 de novembro de 2022

[5] Siemens (2022). https://new.siemens.com/global/en/products/automation/industry-software/automation-software/scada/simatic-wincc-v7.html*,* acedido em 19 de novembro de 2022

[6] SA Automação, Wonderware Intouch (2022). *http://www.sa.online.pt/pt/artigos/info/wonderware-intouch_35/,* acedido em 19 de novembro de 2022

[7] SA Automação, Wonderware System Platform (2022). *http://www.sa.online.pt/pt/artigos/info/wonderware-system-platform_46/,* acedido em 19 de novembro de 2022

[8] Inductive Automation (2022). *https://inductiveautomation.com/scada-software/,* acedido em 19 de novembro de 2022

[9] Inductive Automation (2022). *https://inductiveautomation.com ,* acedido em 26 de novembro de 2022

[10] Inductive Automation, o que é o Ignition (2022). *https://inductiveuniversity.com/videos/what-is-ignition,* acedido em 26 de novembro de 2022

[11] Inductive Automation, arquiteturas (2022). *https://inductiveautomation.com/ignition/architectures,* acedido em 27 de novembro de 2022

[12] Inductive Automation (2022). *https://docs.inductiveautomation.com/display/DOC80/Introducing+Ignition,* acedido em 26 de novembro de 2022

[13] Microsoft (2022). *https://support.microsoft.com/pt-pt/office/no%C3%A7%C3%B5es-b%C3%A1sicas-da-base-de-dados-a849ac16-07c7-4a31-9948-3c8c94a7c204#__toc257378454,* acedido em 3 de dezembro de 2022

[14] SQLite (2022). *https://www.sqlite.org/index.html,* acedido em 3 de dezembro de 2022

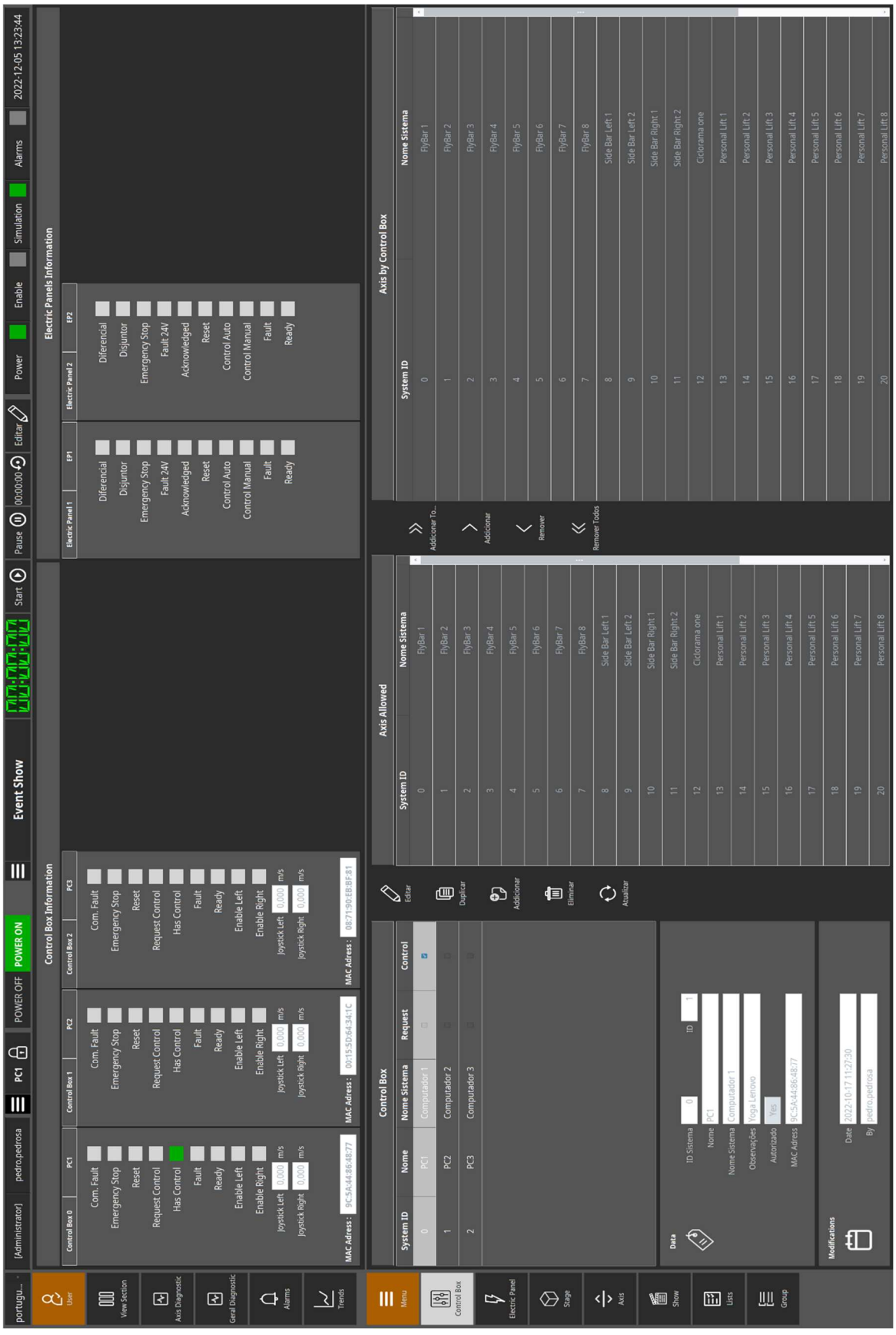
[15] TechTarget (2022). *https://www.techtarget.com/searchdatamanagement/definition/SQL-Server,* acedido em 3 de dezembro de 2022

[16] Microsoft (2022). *https://learn.microsoft.com/en-us/sql/relational-databases/views/views?view=sql-server-ver16,* acedido em 4 de dezembro de 2022

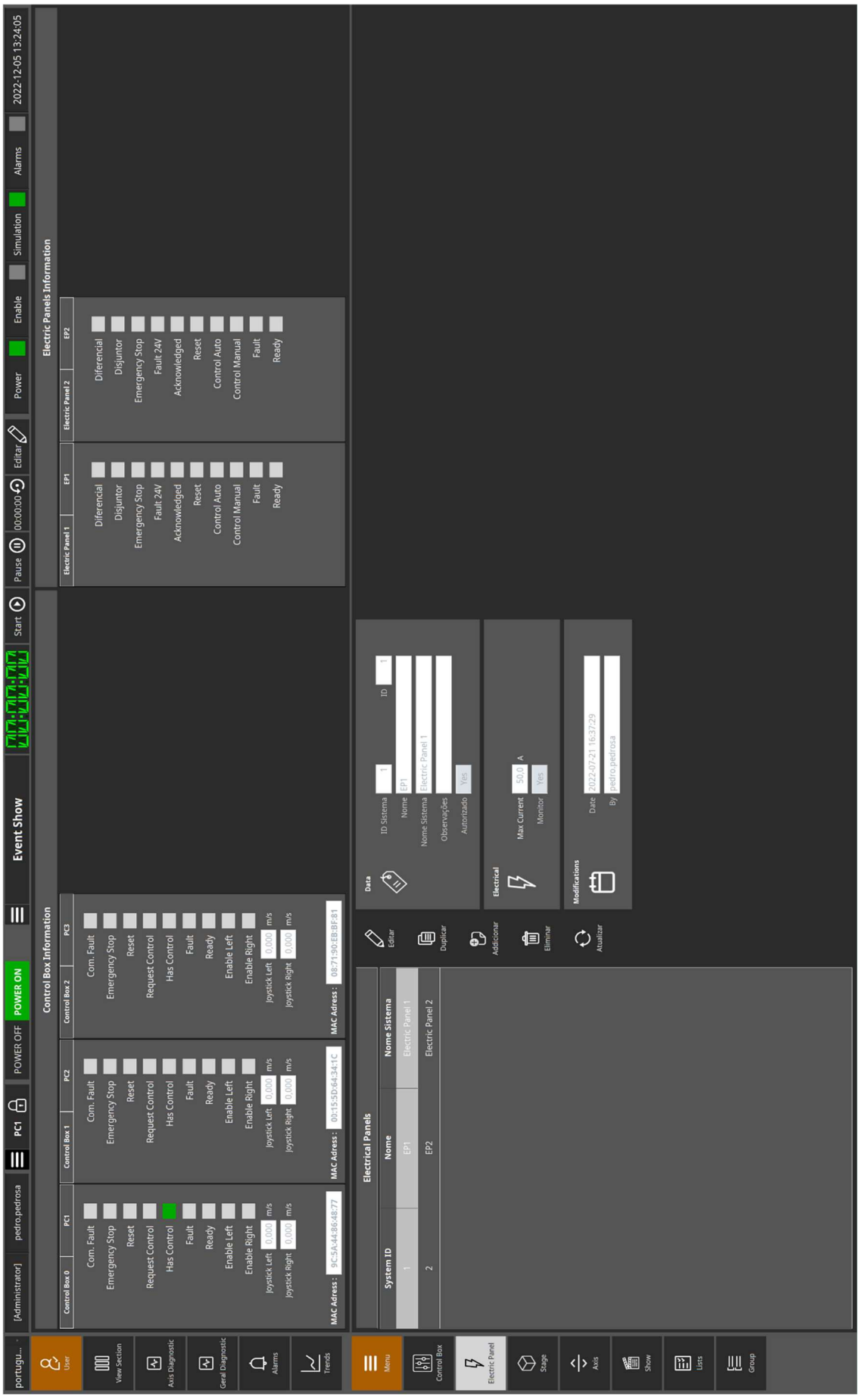
[17] John K. Haas (2014). A History of the Unity Game Engine. https://core.ac.uk/download/pdf/212986458.pdf, acedido em 4 de dezembro de 2022

ANEXO

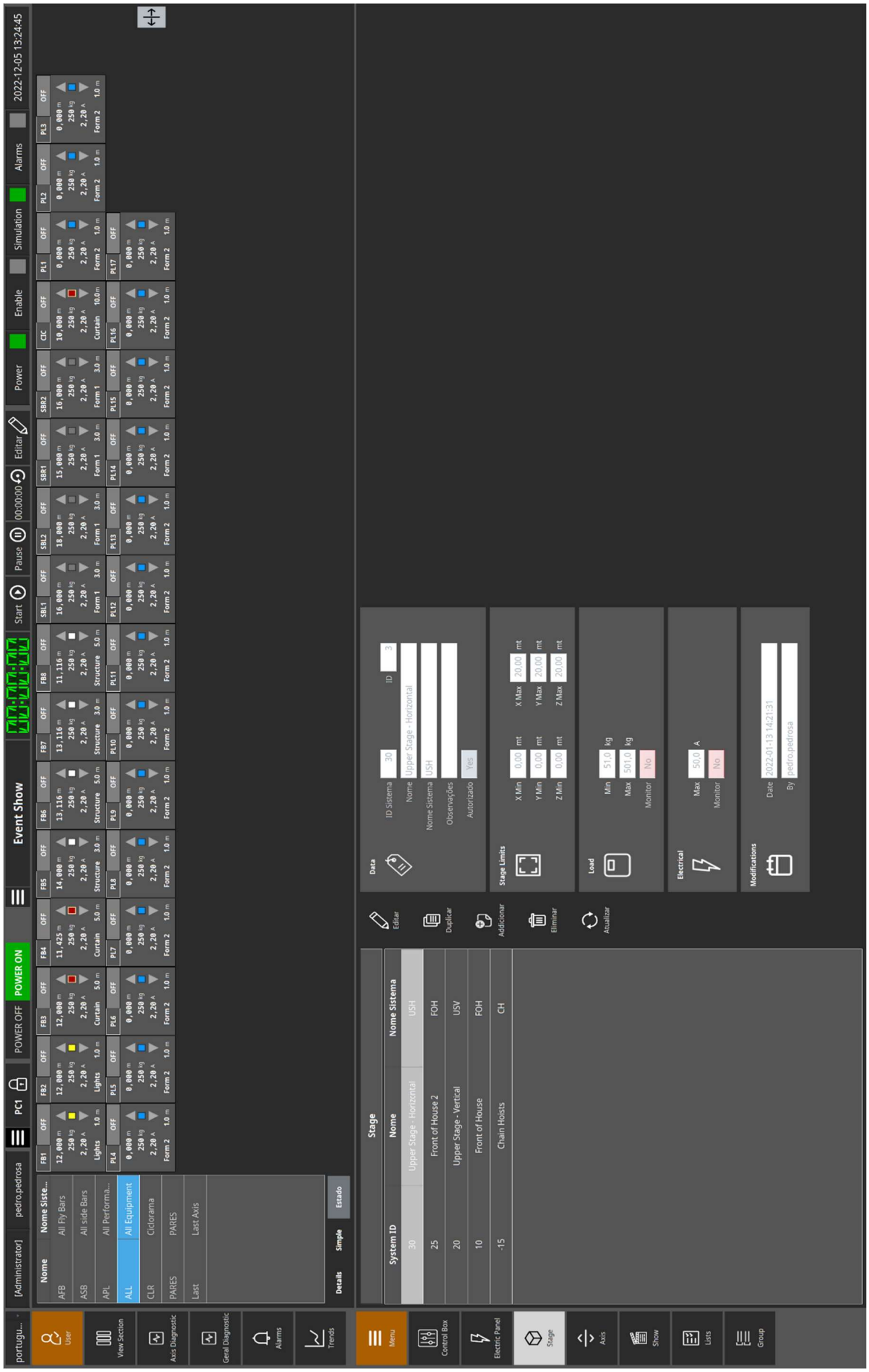
Anexo 1 – Ampliação da Figura 22 e Figura 24



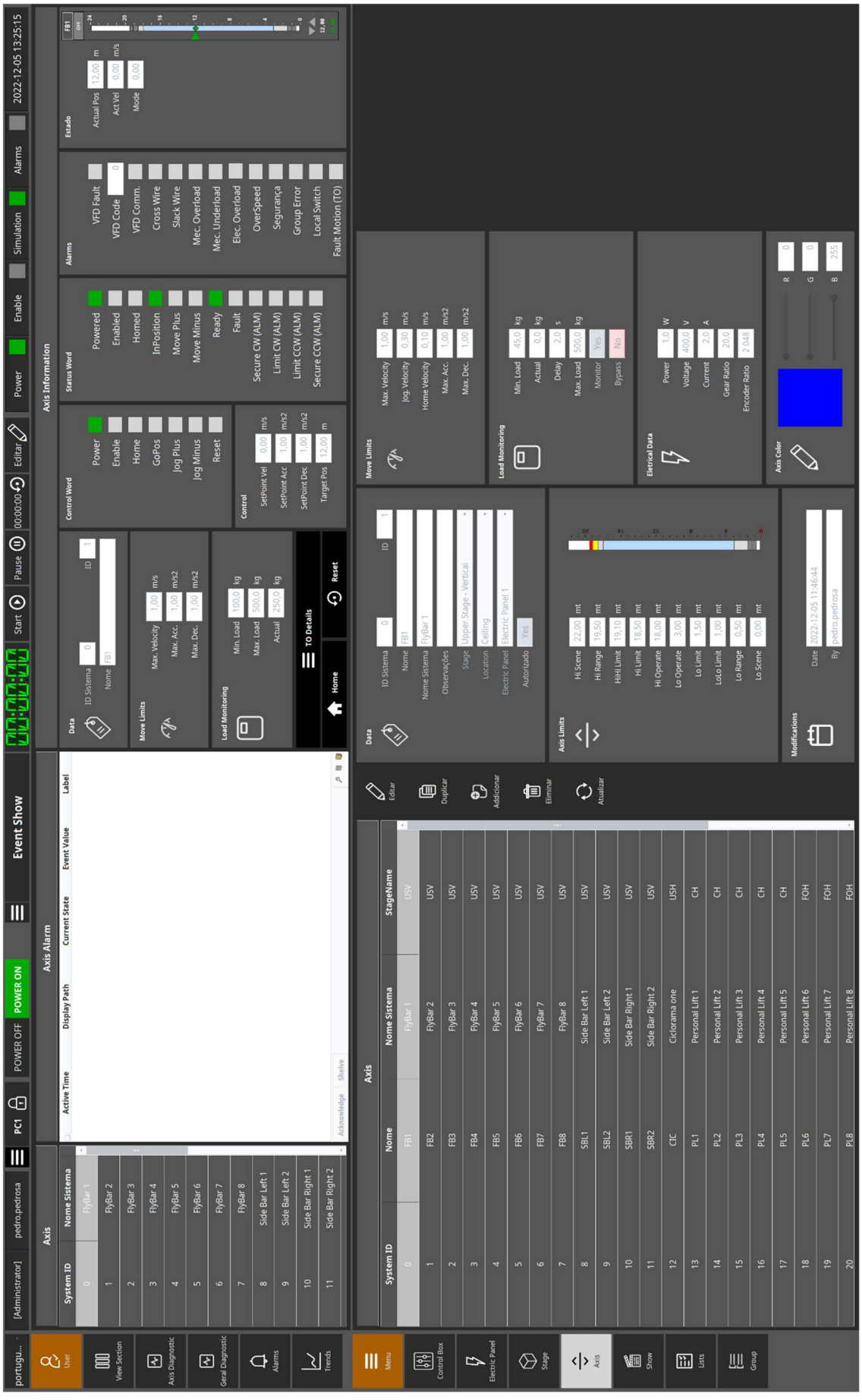
Anexo 2 – Ampliação da Figura 22 e Figura 25



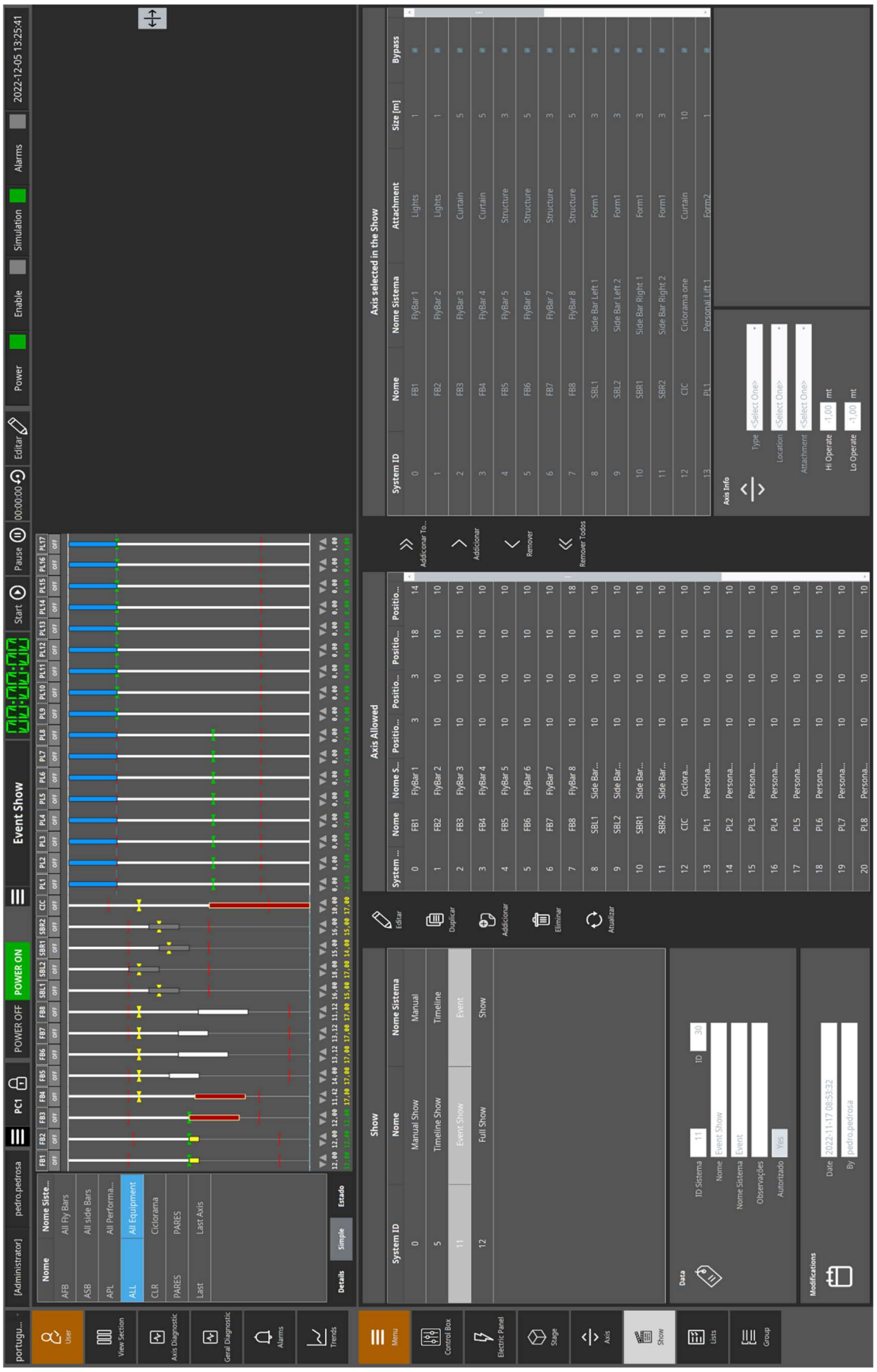
Anexo 3 – Ampliação da Figura 19 e Figura26



Anexo 4 – Ampliação da Figura 20 e Figura27



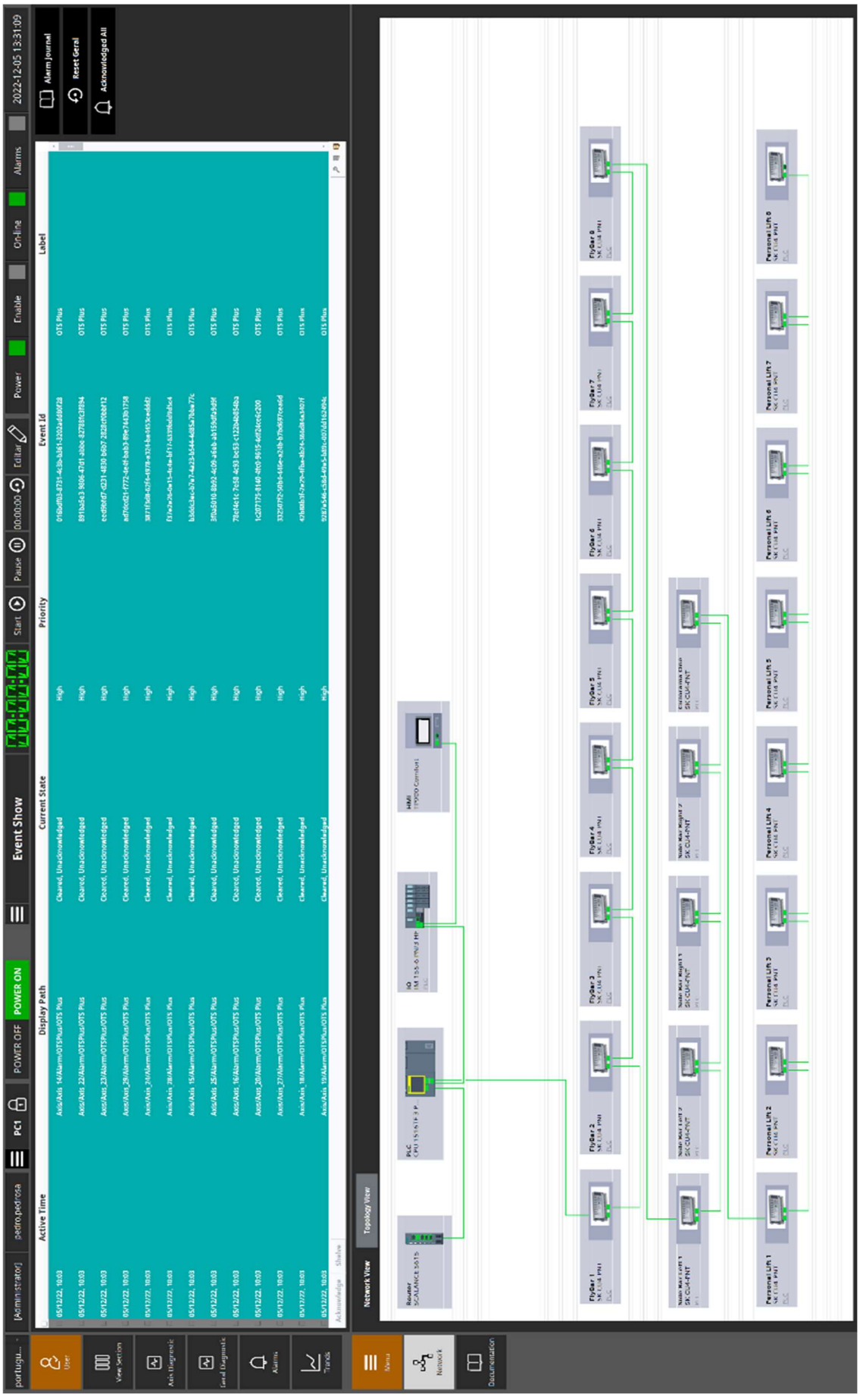
Anexo 5 – Ampliação da Figura 18 e Figura 28



Anexo 6 – Ampliação da Figura 17 e Figura 30

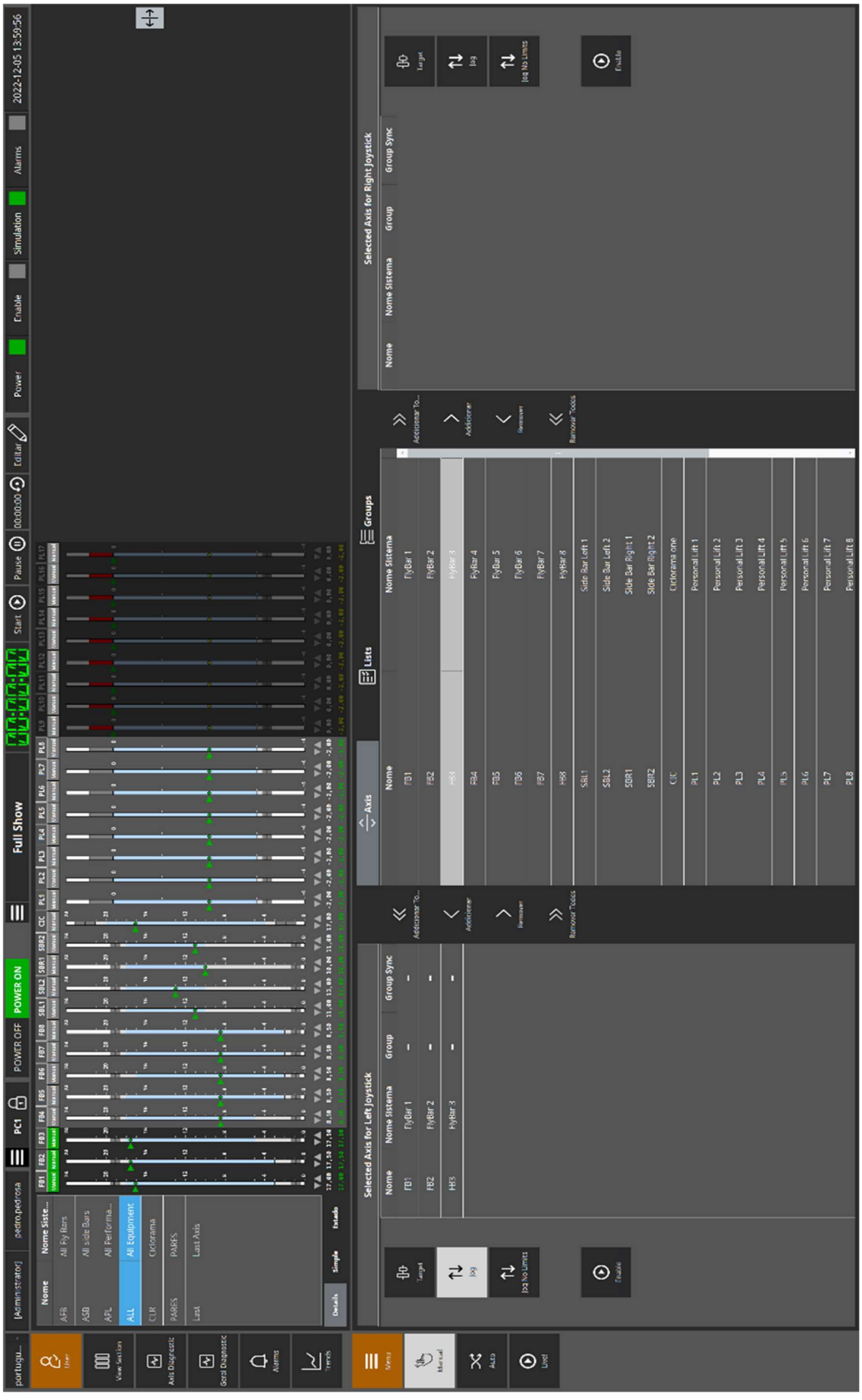
Anexo 7 – Ampliação da Figura 17, Figura 18 e Figura 31

Anexo 8 – Ampliação da Figura 21 e Figura 33

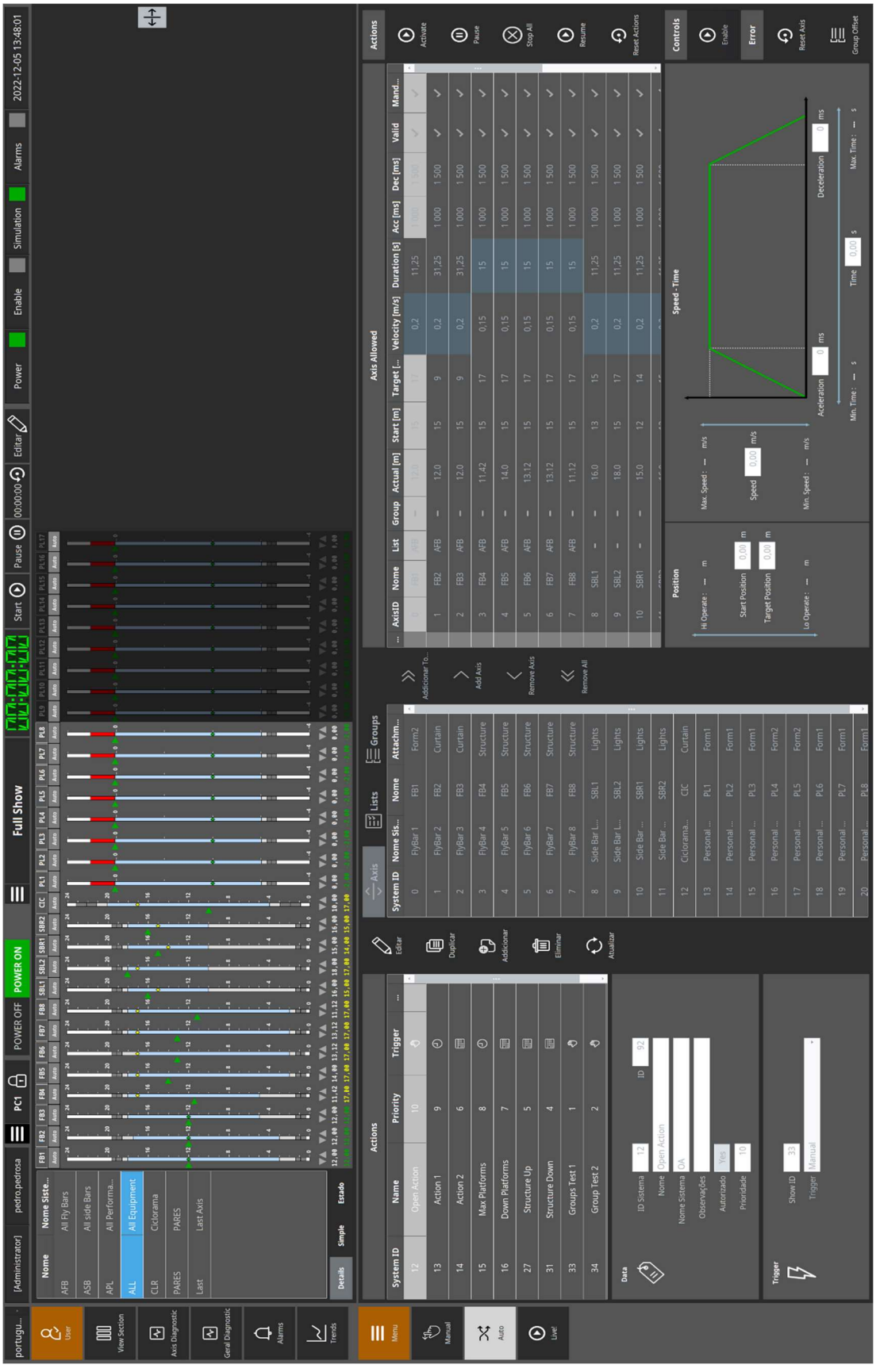


Anexo 9 – Ampliação da Figura 21 e Figura 34

Anexo 10 – Ampliação da Figura 17 e Figura 35



Anexo 11 – Ampliação da Figura 17 e Figura 39



Anexo 12 – Ampliação da Figura 23 e Figura 48