Eric Maciel Cardoso

Análise de Sensibilidade na Plataforma +Precoce

Campo Grande - MS

Março de 2021

Eric Maciel Cardoso

Análise de Sensibilidade na Plataforma +Precoce

Dissertação apresentada ao Programa de Mestrado Stricto Sensu em Computação Aplicada, mantido pela Universidade Federal do Mato Grosso do Sul, como parte dos requisitos para a obtenção do título de Mestre em Computação Aplicada.

Universidade Federal de Mato Grosso do Sul – UFMS

Faculdade de Computação

Mestrado em Computação Aplicada

Orientador: Prof. Dr. Milton Ernesto Romero Romero

Coorientador: MSc. Fernando Rodrigues Teixeira Dias

Campo Grande - MS

Março de 2021

Agradecimentos

Em primeiro lugar agradeço a Deus pelo dom da vida e por me permitir chegar até aqui, me sustentar, me direcionar para tomar as melhores decisões e me ajudar em todos os momentos.

A minha esposa Aymee que está comigo em todos os momentos me motivando, me encorajando e acreditando em mim em todo tempo e ao meu filho Noah que está a caminho e já é um dos principais motivos de alegria da minha vida.

Aos meus pais Celso e Rute e meus sogros Orisvaldo e Clair por serem incentivadores de sonhos. Sem vocês eu não estaria aqui!

Aos meus irmãos de sangue (Henrique e Andrey) e de coração (Tiz e Cézar) por serem suporte e alegria em todo tempo. Aos sobrinhos que nos alegram com seu jeitinho todo especial.

À Universidade Federal de Mato Grosso do Sul (UFMS), em especial a Faculdade de Computação (FACOM) e o programa de Mestrado em Computação Aplicada, pela oportunidade.

Ao meu orientador professor Dr. Milton Ernesto Romero Romero, por tudo que me ensinou e por cada conselho.

A Empresa Brasileira de Pesquisa Agropecuária (Embrapa), na pessoa do meu coorientador MSc. Fernando Rodrigues Teixeira Dias. Obrigado por disponibilizar do seu tempo para me ensinar e direcionar para que este projeto fosse desenvolvido.

A todos os familiares, amigos e professores que direta ou indiretamente fizeram parte desta jornada e contribuíram para que eu chegasse até aqui.

Muito obrigado!

"Dificuldades preparam pessoas comuns para destinos extraordinários"
(C.S Lewis)

Resumo

A Plataforma +Precoce (P+P) é uma ferramenta que permite modelar e simular sistemas melhorados de produção de gado de corte auxiliando o produtor na tomada de decisões e disponibilizando informações a respeito de sistemas de cria, recria e engorda para a produção de bovinos. Atualmente, a P+P permite a um usuário simular resultados de sistemas de produção de gado de corte apenas “ponto a ponto”, isto é, só é possível realizar a simulação de um determinado sistema a partir da seleção de valores específicos para cada parâmetro, mas sem entender diretamente o quanto a variação destes parâmetros pode influenciar o resultado final dos indicadores do sistema. Este trabalho teve como objetivo desenvolver uma ferramenta web para análise de sensibilidade por meio de gráficos em sistemas de produção modelados na P+P, de forma que o usuário seja capaz de identificar parâmetros relevantes para o resultado de um indicador em um determinado sistema.

Um dos principais pontos no desenvolvimento da ferramenta foi utilizar métodos que pudessem ser executados no *browser* do cliente, com o mínimo de solicitações possíveis ao servidor. Pensando nisso, optou-se pela utilização de métodos de análise por triagem e análise local devido, principalmente à sua baixa complexidade na execução do método. Os métodos de análise por triagem fazem uso de um "cenário de controle"onde cada parâmetro possui um valor mínimo e um valor máximo e os parâmetros são variados individualmente e analisados em todo este intervalo. Isto é bem interessante pelo fato dos parâmetros da P+P já serem definidos previamente com um valor *default*, mínimo e máximo, o que facilita a implementação do método. Já os métodos de análise local, trabalham sob a variação dos parâmetros individualmente somente nas proximidades de um valor padrão.

Desta forma, foi verificado que os valores relativos aos parâmetros de um sistema raramente mudam, de forma que a análise de sensibilidade para estes parâmetros podem futuramente serem persistidas no servidor, sem a necessidade de ser calculada em tempo real pelo cliente. Já uma simulação possui os valores dos parâmetros calibrados em tempo real pelo usuário, o que demanda a análise de sensibilidade também em tempo real. Por este motivo, optou-se pela análise de uma faixa maior de valores dos parâmetros (análise por triagem) para análise de sistemas e a utilização de análise local para análise de simulações.

Para analisar todas as variações de um parâmetro nos intervalos estabelecidos, todos os demais parâmetros são mantidos em seus valores padrões, enquanto que para cada ponto variado de um parâmetro, o Simulador da P+P é invocado. A apresentação dos resultados é feita por meio de gráficos para facilitar a leitura e entendimento pelos usuários da P+P. Na ferramenta desenvolvida é possível visualizar quatro tipos de gráficos, sendo três relacionados ao sistema e um relacionado à uma simulação. Os gráficos relacionados ao sistema são eles, respectivamente, gráficos de tornado, gráfico indicadorXparâmetro e histograma de probabilidade. Já o gráfico relacionado à uma simulação é representado por meio de um gráfico de tornado da simulação.

Para evitar recálculo e chamadas ao Simulador com os mesmos valores de parâmetros, o que produziria o mesmo resultado para os indicadores, foi utilizado uma técnica de "*memoize*", que consiste em armazenar os valores já calculados em uma "tabela de pesquisa", de forma a sempre consultá-la quando um novo cálculo precisar ser executado. Desta forma, se um cálculo já estiver sido realizado para um

conjunto de valores de parâmetros e indicador, este é apenas resgatado da tabela, com a chamada do Simulador realizada apenas nos casos em que a combinação não for encontrada na tabela de pesquisa.

Uma limitação encontrada no decorrer do desenvolvimento é que, neste momento, o Simulador da P+P passa por uma atualização e mudança de tecnologia que ainda está em andamento, o que impossibilitou a integração da ferramenta desenvolvida à P+P. Para solucionar este problema e realizar os devidos testes de chamada ao Simulador, foi desenvolvido um "pseudo-simulador", que será substituído assim que o Simulador real da P+P estiver atualizado. Espera-se que a construção desta ferramenta auxilie produtores rurais e/ou técnicos na tomada de decisões ainda mais eficientes, elevando assim a produtividade e qualidade da bovinocultura de corte.

Palavras-chaves: análise de sensibilidade; bovinocultura de corte; indicadores; sistemas de produção, pecuária digital.

Abstract

The Plataforma +Precoce (P+P) is an application that allows to model and simulate improved beef cattle production systems, assisting the producer in decision making and providing information about cow-calf, rearing and finishing systems for beef cattle production. Currently, P+P alows a user to simulate results from beef cattle production systems only “point to point”, i.e., it’s only possible to perform the simulation of a givem system by selecting specific values for each parameter, but without directly understanding how much the variation of these parameters can influence the final result of system’s indicators. This work aimed to develop a web application for sensitivity analysis through graphics in production systems used by P+P, so the user is able to identify relevant parameters for the result of an indicator in a given system.

One of the main aspects in the tool development was to use methods that could be executed in the client’s browser, with the minimum possible requests to the server. With that in mind, we opted for the use of analysis methods by screening and local analysis, mainly due to their low complexity in the method execution. The screening analysis methods make use of a "control scenario"where each parameter has a minimum and a maximum value and the parameters are varied individually and analyzed throughout this interval. This is very interesting because the parameters of the P+P have already been defined in advance with a default value, minimum and maximum, which facilitates the method implementation. On the other hand, the methods of local analysis work under the variation of the parameters individually only in the proximity of a standard value.

In this way, it was verified that the values related to the parameters of a system rarely change, so that the sensitivity analysis for these parameters can be persisted in the server in the future, without the need to be calculated in real time by the client. A simulation, on the other hand, has the parameter values calibrated in real time by the user, which requires sensitivity analysis also in real time. For this reason, we opted for the analysis of a larger range of parameter values (analysis by screening) for systems analysis and the use of local analysis for analysis of simulations.

To analyze all variations of a parameter in the established intervals, all other parameters are kept at their default values, while for each variable point of a parameter, the P + P Simulator is activated. The results are presented through graphics to facilitate the reading and understanding by P + P users. In the developed tool it’s possible to have four types of graphs, three related to the system and one related to a simulation. The graphs related to the system are tornado graphs, graph indicator/parameter and probability histogram. The graph related to a simulation is represented by a simulation tornado graph.

To avoid recalculation and calls to the Simulator with the same parameter values, which would produce the same result for the indicators, a "memoize"technique was used, which consists of storing the values already calculated in a "research table", in order to always refer to it when a new calculation needs to be performed. In this way, if a calculation has already been carried out for a set of parameter and indicator values, it is only retrieved from the table, with the Simulator call made only in cases where the combination is not found in the research table.

A limitation found during the development is that, at this moment, the P + P Simulator is undergoing an update and technology change that is still in progress, which made it impossible to integrate the developed tool with P + P. In order to solve this problem and carry out the necessary call tests to the Simulator, a "pseudo-simulator"was developed, which will be replaced as soon as the real P + P Simulator is updated. It's expected that the development of this tool will assist beef cattle producers and technicians in making even more efficient decisions, thus increasing beef cattle quality and productivity.

Key-words: sensibility analysis; beef cattle; indicator; production systems; digital livestock.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface* - Interface de Programação de Aplicações
Facom	Faculdade de Computação
JSON	*JavaScript Object Notation* - formato compacto, de padrão aberto independente, de troca de dados simples e rápida entre sistemas,
MAPA	Ministério da Agricultura, Pecuária e Abastecimento
PIB	Produto Interno Bruto
PWA	*Progressive Web Apps* - Aplicações Web Progressivas
P+P	Plataforma + Precoce
UFMS	Universidade Federal de Mato Grosso do Sul
W3C	*World Wide Web Consortium* - principal organização de padronização da World Wide Web

Sumário

1 Introdução

1.1 Bovinocultura de Corte

A bovinocultura de corte tem crescido consideravelmente no Brasil de forma a se consolidar como uma importante fonte econômica, principalmente no centro-oeste, onde o clima e o relevo são altamente favoráveis à atividade (IBGE, 2019). Atualmente, o país possui o maior rebanho bovino comercial do mundo e se configura como um dos líderes mundiais na produção, consumo e exportação de carne bovina (BARCELLOS *et al.*, 2013), com o PIB da pecuária representando 8,5% do PIB brasileiro, ou R$ 618,50 bilhões (ABIEC, 2020). Esta configuração se deve principalmente à fatores como a melhora na genética de rebanhos bovinos, a melhoria dos sistemas de manejo e pastagem e gestão de propriedades, alcançados principalmente nas últimas três décadas (BOLFE, 2017).

O centro-oeste, atualmente, se configura como uma das principais regiões para a bovinocultura de corte. De acordo com dados da ABIEC (2020), a região possui uma parte considerável do o rebanho do país, com os três estados (Mato Grosso do Sul, Mato Grosso e Goiás) somando cerca de 34,3% de do rebanho nacional (tabela 1).

Estados	**Rebanho Estimado em 2019 (cabeças)**	**Participação no rebanho do Estado no total do Brasil (%)**	**Crescimento do rebanho nos últimos 10 anos (%)**
Mato Grosso	29.873.068	13,98	9,20
Goiás	22.430.742	10,50	7,45
Minas Gerais	22.321.084	10,45	-0,66
Mato Grosso do Sul	20.985.665	9,82	-6,00
Pará	20.510.169	9,60	21,67
Rondônia	13.973.714	6,54	21,16

Tabela 1: Maiores rebanhos bovinos por estado. Adaptado de ABIEC (2020)

A bovinocultura de corte pode ser dividida em 3 fases distintas: cria, recria e engorda (terminação). Tais fases podem ser desenvolvidas de forma isolada ou combinada (CEZAR *et al.*, 2005). Embora grande parte dos bovinos abatidos no Brasil sejam criados, recriados e terminados em pastagens, tem-se notado ótimos resultados da adoção da engorda (terminação) em confinamento, o que tem despertado interesse por parte de produtores (LANDSKRON; KESSLER, 2018).

Na busca de garantir eficiência em um sistema de produção, o planejamento, controle e gestão produtiva e empresarial têm se tornado prioridade para pecuaristas (ARAÚJO *et al.*, 2012). Como nem sempre é possível controlar o preço do produto vendido e o lucro obtido, é necessário que o produtor faça uma boa administração das variáveis que estão sob seu controle, como a redução de custos, aumento da escala, trabalho com animais de maior produção, etc (ARAÚJO *et al.*, 2012). Quando se fala em busca pela efi-

ciência nos sistemas de produção, o Brasil tem se mostrado promissor no desenvolvimento de tecnologias inovadoras, que o permite "estabelecer sistemas cada vez mais sustentáveis e de qualidade" (BOLFE, 2017).

Desta forma, têm surgido ferramentas que auxiliam produtores na tomada de decisões sobre seu sistema de produção, de modo a mantê-los cada vez mais seguros e produtivos. Podemos citar como exemplo, a Plataforma+Precoce (P+P), desenvolvida por parceria entre a FACOM/UFMS e a Embrapa.

1.2 Plataforma +Precoce

A P+P é um software web capaz de modelar e simular o desempenho econômico e ambiental de sistemas de produção de bovinos de corte. O objetivo da P+P é auxiliar o produtor na tomada de decisões, "disponibilizando informações a respeito de sistemas de cria, recria e engorda para a produção de bovinos, destinados a programas e parcerias que bonificam carcaça por qualidade" (BASSO, 2018).

Basso (2018) define alguns termos que são utilizados pela P+P que, no decorrer deste documento serão bastante citados e, desta forma, se faz importante listá-los neste momento:

- **Modelo**: modelo matemático que calcula recursos necessários para uma produção máxima de um sistema de uma determinada área de produção escolhida pelo usuário, usando um valor atribuído aos seus parâmetros.
- **Sistema**: sistema de produção de bovinos de corte.
- **Simulação**: valores estimados para os indicadores a partir de valores definidos para os parâmetros do sistema, calculados para o modelo do sistema (Figura 1). Após a escolha de um sistema por parte do usuário para consulta ou simulação, são exibidos os dados padrões para o sistema e os parâmetros com seus valores padrões (default).
- **Parâmetro**: entrada utilizada para o cálculo de simulação de um modelo associado ao sistema. Parâmetros possuem nome e descrições únicas, e um valor padrão (default – valor mais provável ou recomendado), mínimo e máximo definidos pelo administrador do sistema (Figura 2), podendo ocorrer casos em que um parâmetro é definido como fixo pelo administrador do sistema. Em geral, existem muitas dezenas ou algumas centenas de parâmetros em um sistema, e cada um deles possui características próprias. Dentre os vários tipos de parâmetros para o sistema, podemos destacar: área de produção, parâmetros de protocolo (Ex.: idade à desmama, quantidade de suplemento, etc), parâmetros de desempenho (Ex.: taxa de natalidade, taxa de mortalidade, ganho de peso médio diário, etc), parâmetros de mercado

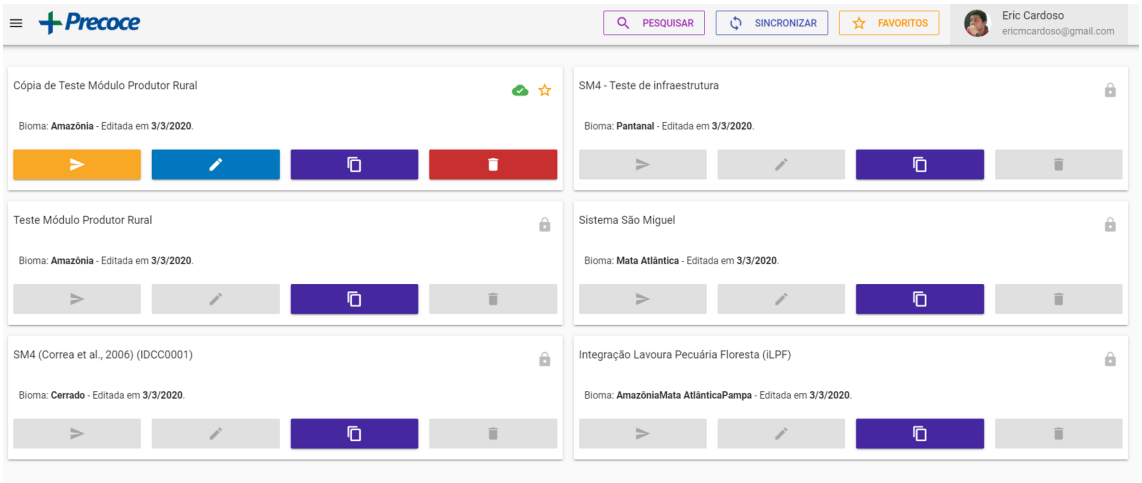

Figura 1: Exemplo de simulações na P+P.

(Ex.: preço do bezerro, preço do boi gordo, preço da arroba, etc), parâmetros de qualidade (relativos à qualidade da carcaça produzida), dentre outros.

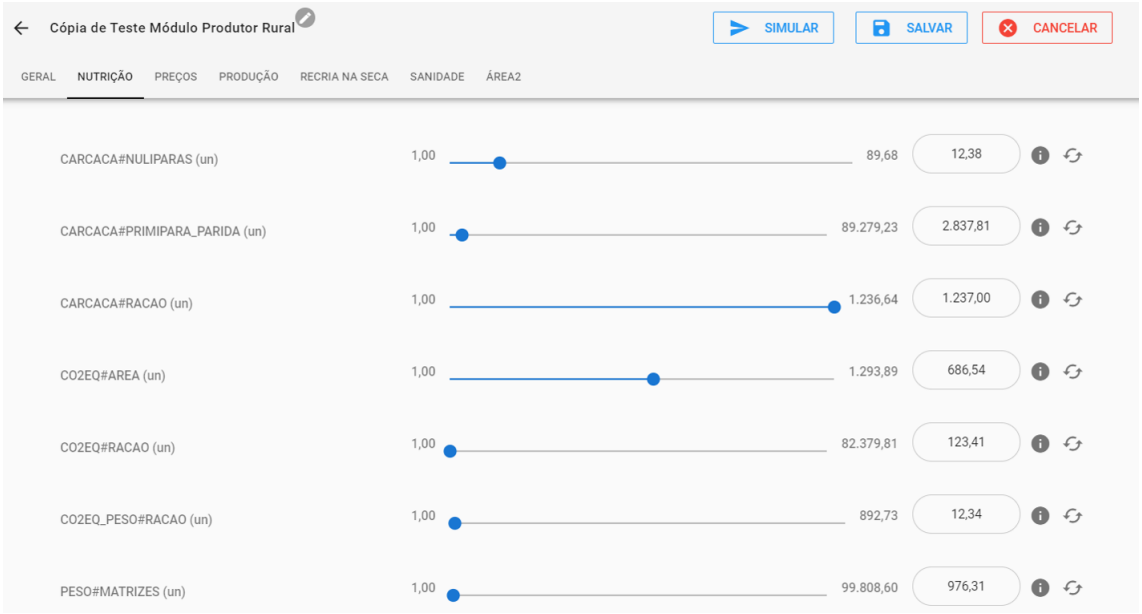

Figura 2: Exemplo de parâmetros para calibragem na P+P.

- **Indicador**: variável calculada pela simulação a partir dos valores atribuídos aos parâmetros quando uma simulação é solicitada. Um indicador possui valor numérico, tendo poucos indicadores com valores alfanuméricos. Os indicadores podem ser divididos em categorias, por exemplo: físicos (estimam o desempenho físico do sistema simulado, como por exemplo, toneladas de carne produzidas por ano), econômico-financeiros (receita líquida anual, lucratividade, etc), ambientais (emissão de GEE de CO2eq em ton, etc). Ao todo são 122 indicadores definidos para a P+P, nem todos aplicáveis a todos os sistemas.

A P+P passa atualmente por atualizações e mudança de tecnologia em alguns de seus módulos e sua arquitetura básica pode ser observada na figura 3.

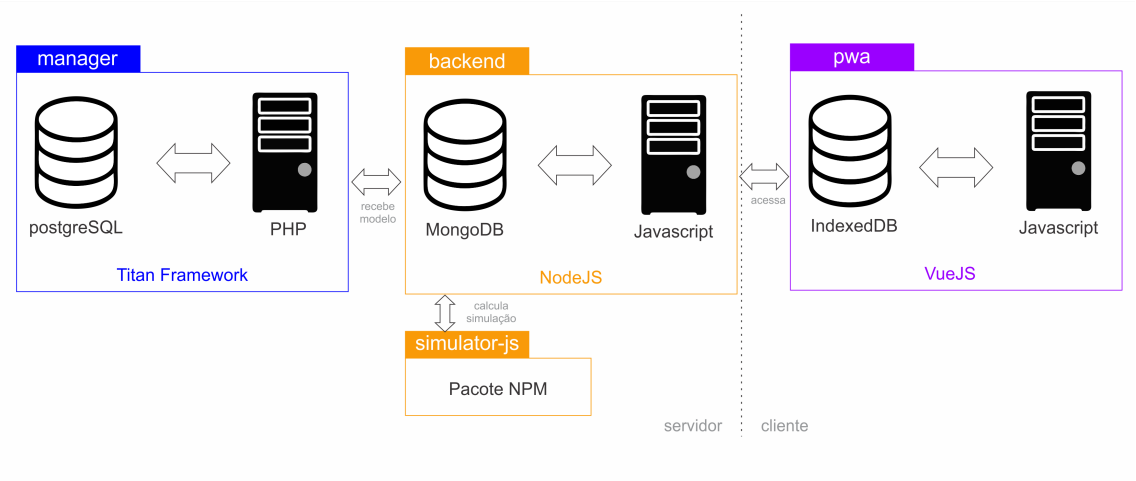

Figura 3: Arquitetura da plataforma + Precoce.

1.3 Motivação e Justificativa

Atualmente, a P+P permite a um usuário simular os resultados de sistemas de produção de gado de corte apenas "ponto a ponto", ou seja, o usuário seleciona o valor de cada parâmetro, entre um valor mínimo e um valor máximo e, a partir daí, gera a simulação, não tendo como saber como a calibração de um determinado parâmetro influencia sobre o resultado final de um indicador. A única forma hoje de avaliar as influências de cada Parâmetro em cada Indicador é através da "tentativa e erro" na alteração dos valores dos parâmetros, o que se torna dispendioso e pode influenciá-lo na tomada de decisões equivocadas.

Uma das técnicas mais utilizadas para solucionar tal dificuldade é a análise de sensibilidade. A análise de sensibilidade busca entender e estimar a relação que possui o resultado final em decorrência da variação de parâmetros de entrada (LESSA, 2013; FREY; MOKHTARI; DANISH, 2003).

1.4 Objetivos

Esta seção descreve os objetivos deste trabalho, geral e específico.

1.4.1 Objetivo Geral

Projetar e desenvolver uma aplicação web para análise de sensibilidade em sistemas de produção usados pela P+P, de forma que o usuário seja capaz de identificar o quanto

a variação de um ou mais parâmetros influenciam no resultado de um indicador em um determinado sistema e/ou simulação.

1.4.2 Objetivos Específicos

- Contribuir, no contexto agropecuário, para a representação de análise de sensibilidade em sistemas de produção de bovinos de corte;
- Especificar gráficos capazes de demonstrar visualmente a análise de sensibilidade nos sistemas utilizados pela P+P;
- Desenvolver módulo adicional de análise de sensibilidade na P+P;
- Aplicar o módulo desenvolvido para fins de validação;

1.5 Organização do texto

O capítulo 2 apresenta os principais conceitos que serviram de base para o desenvolvimento deste trabalho, ou seja, a fundamentação teórica, abordando conceitos como bovinocultura de corte, análise de sensibilidade e as tecnologias utilizadas para desenvolvimento. O capítulo 3 apresenta detalhes do projeto e implementação da aplicação. O capítulo 4 apresenta o passo a passo da utilização da ferramenta de análise de sensibilidade. O capítulo 5 apresenta a conclusão e propostas de trabalhos futuros.

2 Fundamentação Teórica

O capítulo apresenta o referencial teórico necessário para compreensão deste trabalho, isto é, conceitos de bovinocultura de corte, análise de sensibilidade e as tecnologias utilizadas para desenvolvimento do projeto.

2.1 Bovinocultura de Corte

Bovinocultura de corte pode ser definido como "os sistemas comerciais de criação de bovinos destinados à produção de carne" (RODRIGUES; LAGES; SILVA, 2014). Estes sistemas podem ser vistos como um conjunto de tecnologia e práticas de manejo, que estão intimamente relacionados à condições socioeconômicas da região, a capacidade de esta promover investimentos, o tipo do animal e o propósito da criação (FILHO, 2007). Todas estas características são decisivas para uma que um sistema seja considerado eficaz e se alcance o objetivo esperado. Em suma, a bovinocultura de corte dedica-se à criação do bovino "a partir do desenvolvimento de um ciclo produtivo que envolve desde o manejo reprodutivo até o beneficiamento da carne" (SOARES *et al.*, 2016).

Rodrigues, Lages e Silva (2014) classifica os sistemas de produção em 3 categorias principais: intensiva, onde o rebanho depende da ação humana para suas necessidades básicas diárias e está em confinamento; extensiva, onde o rebanho possui certa autonomia sobre as suas necessidades básicas possuindo uma certa liberdade para movimentar-se ao ar livre; e semi-intensiva, onde o rebanho "está exposto a qualquer combinação de métodos de pecuária intensiva e extensiva, simultânea ou alternadamente" (RODRIGUES; LAGES; SILVA, 2014). "A característica principal da pecuária de corte brasileira é a diversidade de sistemas produtivos ajustados aos diferentes biomas e padrões socioculturais de cada região" (BARCELLOS; OLIVEIRA; MARQUES, 2016). Com toda esta diversidade apresentada em cada região do Brasil, dificilmente um sistema de produção de gado de corte possuirá características padrões (únicas) por todo o território, o que obriga ao produtor planejar e analisar cada contexto em particular, de forma a empregar as melhores técnicas de manejo e aplicar da melhor forma a tecnologia para garantir a eficiência do seu sistema de produção (FILHO, 2007).

Neste contexto, o uso de ferramentas tecnológicas tem se apresentado como algo bastante promissor no planejamento de sistemas de produção. As inovações aplicadas a estes sistemas têm surpreendido e muito nas últimas décadas. O que antes era visto como resistente à inovações tecnológicas, que possuía uma gestão ultrapassada (BARCELLOS *et al.*, 2013), um baixo comprometimento com os recursos naturais (o que contribuía para o desequilíbrio planta-solo-animal) resultando em consequências graves para a pró-

pria atividade e também para o meio ambiente (FILHO, 2007), os sistemas de produção de bovinos de corte têm apresentando nas últimas 3 décadas uma revolucionária modernização "sustentada por avanços no nível tecnológico dos sistemas de produção e na organização da cadeia, com claro reflexo na qualidade da carne bovina" (GOMES; FEIJÓ; CHIARI, 2017).

Esta modernização tem fomentado um aumento significativo na produtividade e qualidade do rebanho brasileiro, permitindo que o país se configure, nos últimos anos, como um dos principais atores na produção e comércio de carne bovina no mundo (GOMES; FEIJÓ; CHIARI, 2017). Com isso, as boas práticas têm se tornado cada vez mais frequentes na busca de sistemas de produção mais eficientes, mais competitivos, de forma a oferecer produtos com qualidade cada vez maiores, respeitando o meio ambiente e garantindo eficiência no objetivo final.

2.2 Análise de Sensibilidade

Análise de sensibilidade pode ser definida como uma investigação sistemática que possibilita a análise de variáveis de saída de um processo, com o objetivo de identificar como as variáveis de entrada influenciam no resultado final, isto é, como um modelo responde, quantitativa e qualitativamente, a diferentes variações nas informações que o alimenta (JÚNIOR, 2012; SILVA, 2010; SILVA; GHISI, 2013).

O objetivo da análise de sensibilidade é buscar entender e estimar a relação que possui o resultado final em decorrência da variação das variáveis de entrada, ou seja, seu foco de estudo está na relação entrada e saída de um modelo e, como este modelo responde à variação de parâmetros (LESSA, 2013; SILVA, 2010). Para Lessa (2013), a análise de sensibilidade pode ser útil para vários fins em um modelo ou sistema, como por exemplo:

- testar a robustez dos resultados diante de incertezas;
- entender e estimar a relação existente entre as variáveis de entrada e de saída;
- identificar variáveis relevantes e influentes para as variáveis de saída, de modo a reduzir riscos e encontrar erros;
- simplificar e otimizar o modelo, fixando valores em variáveis de entrada que não possui uma relevante influência sobre as variáveis de saída.

Silva (2010) e Nunes (2012), entendem que a análise de sensibilidade é conduzida pelos modeladores de sistemas computacionais para os seguintes pontos principais:

1. determinar se o modelo representa o sistema estudado;

2. verificar a relevância dos parâmetros, de modo a entender, quais mais contribuem para o resultado final (saída) do modelo;

3. identificar quais parâmetros no modelo não exercem influência no resultado final;

4. verificar a interação entre grupos de dados;

5. se existe alguma região no espaço dos dados de entrada em que a variação do modelo é máxima.

De modo geral, a análise de sensibilidade auxilia na tomada de decisões, porque permite analisar quais dados devem ser refinados dentre os que possuem mais influência sobre o resultado final, a partir principalmente dos fatores críticos do modelo ou sistema (JÚNIOR, 2012). Para Silva (2010), entender e priorizar parâmetros é essencial para tirar o foco de parâmetros que não são sensíveis (não exercem influência) sobre o resultado final. A literatura apresenta, pelo menos 3 tipos de análise de sensibilidade (SILVA; GHISI, 2013): Análise por triagem, Análise local e Análise global.

2.2.1 Análise por triagem

Trata-se de métodos que fazem uso da variação individual dos parâmetros de entrada em dois extremos (um valor máximo e um valor mínimo), enquanto os demais parâmetros se mantêm no valor padrão (médio) (SILVA; GHISI, 2013). A análise por triagem é útil para o tratamento de modelos que "exigem muito esforço computacional e têm um grande número de parâmetros de entrada", identificando um subconjunto de fatores que controlam a variação do resultado (NUNES, 2012). Métodos de triagem podem ser considerados como uma "atividade preliminar a uma análise de sensibilidade maior, que independe do número de rodagens do programa utilizado" (NUNES, 2012). O objetivo principal destes métodos é identificar quais fatores (qualquer entrada de um modelo, como um parâmetro por exemplo) envolvidos no modelo são mais relevantes para o resultado final (saída). Esta característica pode ser vista como uma desvantagem dos métodos de análise por triagem, visto que estes métodos classificam os parâmetros em ordem de importância (medidas qualitativas), mas não possibilitam quantificar o quanto (medida quantitativa) um parâmetro é mais relevante que outro (NUNES, 2012). Dentre os métodos de triagem encontrados na literatura, podemos citar o método *Ceteris Paribus* (um fator por vez) e o método de Efeitos Elementares, proposto por Morris em 1991.

A abordagem *Ceteris paribus* trabalha em um cenário denominado como "cenário de controle" (NUNES, 2012), onde cada parâmetro possui um valor padrão (*default*). O objetivo deste método é calcular a saída do sistema ajustando um fator por vez, em seus valores extremos (mínimo e máximo) de forma que todos os outros se mantém constantes (SILVA, 2010; NUNES, 2012). "Analisando a magnitude da diferença entre a resposta

do modelo alterado e a resposta obtida no cenário de controle" (NUNES, 2012) a partir da variação para os valores extremos do fator é possível medir a influência de cada fator sobre o modelo. Este tipo de abordagem tem um custo computacional baixo, normalmente 2k+1, onde k é o número de fatores examinados (NUNES, 2012).

O método de Efeitos Elementares, conhecido também como método de Morris, tem por objetivo "o estudo das interações lineares entre os parâmetros, determinando a ordem de importância dos valores dos parâmetros em termo de seus efeitos médios na variância dos resultados simulados" (SILVA, 2010). Diferentemente do método *Ceteris Paribus*, que todos os parâmetros permanecem inalterados enquanto apenas um parâmetro é variado, o método de Efeitos Elementares é aplicado para analisar a sensibilidade global do modelo (SILVA, 2010), daí o fato deste método ser visto também como um método de análise global. Para Nunes (2012), o método proposto por Morris é simples, fácil de implementar, seus resultados possuem uma interpretação direta e seu custo computacional é visto como econômico.

2.2.2 Análise local

Para este tipo de análise, a variação dos parâmetros fornece "a tendência do modelo analisado em um determinado ponto do espaço amostral" (NUNES, 2012), ou seja, esta variação normalmente se dá na proximidade dos valores *default* e o resultado depende da escolha destes valores (SILVA, 2010). Para Silva e Ghisi (2013), neste tipo de análise, todos os parâmetros são mantidos constantes, enquanto um parâmetro específico sofre variações em diversos níveis, de forma que se obtenha a tendência deste em cada nível. Neste tipo de análise, pode haver a variação de outros parâmetros, mas de forma separada (SILVA; GHISI, 2013).

2.2.3 Análise global

Tipo de análise mais complexa, que permite a variação de todos os parâmetros de forma simultânea, na ideia de obter índices globais de sensibilidade (SILVA; GHISI, 2013). Os métodos de análise global levam em conta todo o intervalo de valores possíveis para todos os parâmetros, buscando "alocar a incerteza na saída à incerteza dos fatores de entrada" (SILVA, 2010). Como exemplos mais conhecidos de análise global estão os métodos de Monte Carlo e os métodos baseados na variância. Em geral estes tipos de métodos dificultam a execução de análises rotineiras, por serem bem mais complexos que os demais analisados (NUNES, 2012).

2.3 Tecnologia e técnicas de desenvolvimento

2.3.1 *Memoize*

No decorrer da execução de um software uma determinada função pode ser invocada várias vezes pelo mesmo usuário ou por usuários diferentes, com os mesmos valores de entrada, argumentos ou entradas semelhantes produzindo cálculos redundantes, resultado normalmente nos mesmos tipos de saídas. Nestes casos, pode ser interessante, na maioria das vezes, algum mecanismo que identifique estas entradas semelhantes e armazene resultados de saídas gerados a partir delas, de forma a reutilizar resultados, reduzir o tempo de execução da função e não necessitar recalcular valores para mesmos conjuntos de dados (TSUMURA *et al.*, 2007; HALL; MCNAMEE, 1997).

Neste sentido, a *memoização (memoize)* "é uma técnica de programação amplamente usada para acelerar" (TSUMURA *et al.*, 2007). Seu objetivo principal é armazenar resultado de funções para futuras necessidades de reutilização. "A versão manual desta técnica geralmente envolve a construção de tabelas de pesquisa e é uma técnica padrão em programação dinâmica" (HALL; MCNAMEE, 1997). Para esta técnica, todas as vezes que a função é invocada, as variáveis de entrada são comparadas com variáveis já calculadas e que estão armazenadas na tabela de pesquisa. Se o conjunto de entrada analisado corresponder à algum conjunto armazenado na tabela de pesquisa, "o mecanismo de *memoização* grava as saídas armazenadas no cache e/ou nos registros", reduzindo assim a quantidade de cálculos finais e possivelmente, diminuindo o tempo de execução da função por completo (TSUMURA *et al.*, 2007).

2.3.2 Node.js

O *Node.js* surgiu em 2009,desenvolvido por Ryan Dahl e 14 colaboradores, com o intuito de suprir um problema presente nas plataformas de desenvolvimento para web como .NET, Java, PHP, Python, etc. O Node.js é uma "arquitetura *non-blocking thread* (não-bloqueante), apresentando uma boa performance com consumo de memória e utilizando ao máximo e de forma eficiente o poder de processamento dos servidores" (PEREIRA, 2013). Usuários de sistemas Node estão livres de aguardarem por muito tempo o resultado de seus processos, e principalmente não sofrerão de *dead-locks* no sistema, porque nada bloqueia em sua plataforma e desenvolver sistemas nesse paradigma é simples e prático. *Node.js* têm como linguagem de programação o javascript, é orientado a eventos e segue a mesma filosofia de orientação de eventos do javascript *client-side* (PEREIRA, 2013).

2.3.3 PWA (*Progressive Web Apps*)

PWA (acrônimo para *Progressive Web Apps*, ou Aplicações Web Progressivas) é um paradigma de desenvolvimento que visa disponibilizar uma aplicação para o usuário sem nenhum tipo de restrição tecnológica, ou seja, a aplicação deve estar disponível independentemente do dispositivo, navegador ou conexão com a internet. A palavra progressiva vem da ideia de "condicionar uma aplicação para atender o maior número de pessoas possíveis, sob todos os aspectos" (PONTES, 2018).

PWA é uma alternativa rápida e confiável que, quando instaladas em smartphones têm seu funcionamento semelhante à de um aplicativo, permitindo até mesmo o uso sem conexão com a internet. Dentre os principais requisitos de uma PWA citadas por Pontes (2018), podemos destacar:

- registro um *Service Worker*, que pode ser definido como uma especificação W3C que permite implementação de recursos off-line, organização de cache de arquivos estáticos, etc. Isto se dá através da execução de um trecho de código javascript contínuo no nível do navegador;
- resposta com status 200 (sucesso na solicitação), mesmo estando off-line;
- prover uma resposta programada, mesmo que o javascript esteja desabilitado ou não seja suportado pelo navegador;
- fazer uso de HTTPS, de forma que todo conteúdo HTTP seja redirecionado para HTTPS;
- permitir que o usuário instale a aplicação, etc.

2.3.4 Vue.js

Lançado em 2014, o Vue utiliza javascript e é uma estrutura progressiva (PWA) das mais populares para construção de interfaces com o usuário (VUEJS.ORG, 2020). Criado por Evan You, é um *framework* de código aberto, fácil de ser integrado a outras bibliotecas ou projetos existentes (ENZ, 2019). Uma das razões que fazem do Vue um dos *frameworks* mais populares "é o amplo uso de componentes que permitem que os desenvolvedores criem módulos concisos para serem usados e reutilizados em todo o aplicativo" (VUEJS.ORG, 2020). Em sua grande maioria, o *framework* é utilizado para implementação de páginas únicas (*Single Page Application*) (ENZ, 2019).

2.3.5 Vuetify

Uma das principais bibliotecas de componentes do Vue.js têm como principal objetivo, fornecer componentes que permitam ao usuário a criação de aplicações web atraentes

e ricas (VUEJS.COM, 2020). Desenvolvida em 2016, a biblioteca faz uso da especificação Material Design e, cada componente "é criado manualmente para oferecer melhores ferramentas de interface de usuário possíveis" (VUEJS.COM, 2020) para o desenvolvimento de aplicativos. De acordo com Enz (2019), o Vuetify têm como principais características a disponibilização de componentes limpos e reutilizáveis, facilidade na instalação e uso, bem como uma documentação vasta e sólida. "Ao desenvolver com Vuetify, você pode esperar um rico ecossistema de ferramentas e plug-ins, uma comunidade massiva e suporte a longo prazo" (VUEJS.COM, 2020).

2.3.6 Data-Driven Documents (D3.js)

Uma ferramenta de código aberto, D3 é uma biblioteca para visualização de dados implementada em javascript (ENZ, 2019). Disponível desde 2011, a biblioteca é desenvolvida por Mike Bostock e conta com uma ampla documentação, como tutoriais, vídeo aulas, livros, tudo produzido pela própria equipe D3 e por uma comunidade empenhada em auxiliar no aumento do aprendizado e produtividade (D3, 2020). O D3 "combina poderosas técnicas de visualização e interação com uma abordagem orientada a dados para manipulação de DOM" (ENZ, 2019), de forma a permitir liberdade de projetar uma interface visual adequada para visualização dos dados. Esta liberdade de projeto se dá principalmente pelo fato do D3 "se concentrar em primitivas compossíveis, como formas e escalas, ao invés de gráficos configuráveis" (ENZ, 2019). A biblioteca faz uso de SVG (*Scalable Vector Graphics*, traduzido como gráficos vetoriais escalonáveis), Canvas e HTML para dar vida aos dados (D3, 2020).

2.3.7 *IndexedDB*

IndexedDB é uma API de armazenamento *client-side* para um grande volume de dados e busca com alta performance por índices (DOCS, 2020). É uma forma de, no navegador do usuário, armazenar e recuperar dados indexados por meio de uma "chave", de forma consistente. "*IndexedDB* é construído em um modelo de banco de dados transacional"(DOCS, 2020). Isso significa que tudo no *IndexedDB* acontece em uma transação, não sendo possível por exemplo executar comandos fora de uma transação, pois, isso geraria exceções.

2.3.8 Dexie.js

Escrito para ser facilmente utilizado e entendido, Dexie.js é um *wrapper* minimalista para *IndexedDB* (DEXIE.ORG, 2020). Possui uma documentação vasta com exemplos bem documentados, fácil configuração e por seu desempenho e performance é considerado quase nativo por utilizar recursos muitas vezes esquecidos do *IndexedDB*

(DEXIE.ORG, 2020). A vantagem principal em utilizar Dexie é a resolução de três problemas principais encontrados na API *IndexedDB* nativa: tratamento de erros ambíguos, consultas ruins e complexidade do código (FAHLANDER, 2020).

3 Projeto do Módulo de Análise de Sensibilidade na P+P

O capítulo em questão descreve o desenvolvimento do módulo de Análise de Sensibilidade da Plataforma + Precoce. Para facilitar o desenvolvimento e testes deste módulo seu desenvolvimento foi realizado de forma independente da P+P, embora faça uso das mesmas tecnologias, *frameworks* e *wrappers* utilizados na implementação dos principais módulos da P+P.

3.1 Comunicação

Toda comunicação para desenvolvimento do módulo foi realizada em reuniões semanais ou quinzenais por meio de vídeo conferência com o pesquisador da Embrapa Gado de Corte Fernando Rodrigues Teixeira Dias. O orientador Dr. Milton Ernesto Romero Romero também esteve envolvido em várias discussões relativas ao projeto em questão. Vale destacar que durante praticamente todo o processo de desenvolvimento enfrentamos uma pandemia global que fez com que ferramentas de comunicação (*WhatsApp*, *Google Meet*, E-mail, *Github*) fossem de extrema importância para o andamento do projeto.

Embora o projeto tenha sido realizado de forma individual e nenhum processo de desenvolvimento foi adotado na gestão do projeto, os passos de desenvolvimento se assemelham a um modelo ágil por se tratar de passos incrementais de entrega, possibilitando melhor visualização do andamento do projeto. A cada reunião, o projeto foi sendo alinhado para que pudesse chegar ao seu resultado final de forma satisfatória.

3.2 Escolha do método

Para Nunes (2012), escolher um método de análise de sensibilidade não é tão simples, visto que cada método tem seus pontos fortes e fracos, sendo necessário levar em conta o problema a ser investigado, ferramentas computacionais disponíveis, etc. Para Lessa (2013), definir um método a utilizar em uma análise de sensibilidade, alguns fatores precisam ser analisados, como: custo computacional (tempo necessário para que análise do modelo e quantidade de parâmetros que este modelo possui); correlação das entradas (vários dos métodos trabalham com as variáveis como se fossem independentes, e isso nem sempre é verdade); não-linearidade (se a resposta do modelo é não-linear, normalmente os métodos não produzem resultados precisos); interações cruzadas (quando duas ou mais variáveis juntas causam uma maior variação no resultado do que se comparado uma

variável de cada vez); múltiplas saídas (se as variáveis de saída estão correlacionadas, a interpretação dos resultados pode ser mais complexa); e limitação dos dados (em alguns casos, a análise de sensibilidade deve ser realizada com dados disponibilizados por outros estudos).

Neste projeto, a partir de discussões com técnicos da Embrapa Gado de Corte optou-se por uma abordagem que se enquadrasse na categoria de "análise por triagem" para todos os gráficos voltados para análise do sistema devido ao fato desta análise, embora esteja envolvida em analisar uma faixa de valores importantes (variando do mínimo ao máximo de cada parâmetro), o fato destas faixas de valores serem fixadas quando o sistema é desenhado, permite que esta análise seja realizada antecipadamente e os dados persistidos no servidor. Já para os gráficos voltados para a análise da simulação optou-se por trabalhar com um método de "análise local", tendo em vista um menor custo computacional e um menor número de pontos a serem analisados. De acordo com Nunes (2012), "métodos de triagem têm sido aplicados em vários estudos de modelos computacionais gerando bons resultados". Como cada parâmetro já possui um valor *default* pré-definido na plataforma, o objetivo é variar sob um limite mínimo e máximo em torno destes valores de forma a identificar o quanto esta variação pode influenciar no resultado final do modelo. Optou-se por não implementar um método de análise do tipo "global", tendo em vista a complexidade exponencial na quantidade de parâmetros e a necessidade da análise na P+P poder ser executada em tempo real, a qualquer momento no *browser* do próprio usuário.

3.3 Visão Geral do Sistema

O módulo de Análise de Sensibilidade é uma aplicação web que permite ao usuário analisar em sistemas de produção usados pela P+P o quanto a variação de um ou mais parâmetros influenciam no resultado de um determinado indicador. A figura 4 demonstra o esquema do módulo em questão.

O acesso a informações oriundas de módulos já existentes na P+P é dado apenas no início da aplicação, no momento em que o usuário seleciona um determinado sistema para realizar uma simulação, calibra os parâmetros desta simulação de acordo com sua necessidade e confirma a simulação a partir do clique no botão "simular". A partir desta ação, é gerado um *JSON* com todas as informações relativas à simulação escolhida e necessárias para a chamada ao Simulador da P+P, o que nesta seção chamaremos de *JSON* padrão. Para o módulo de análise de sensibilidade, os dados relevantes encontram-se no *JSON* padrão em *simulationData*, sendo eles:

- *parameters*: os parâmetros que fazem parte do sistema escolhido e seus valores definidos para a simulação;

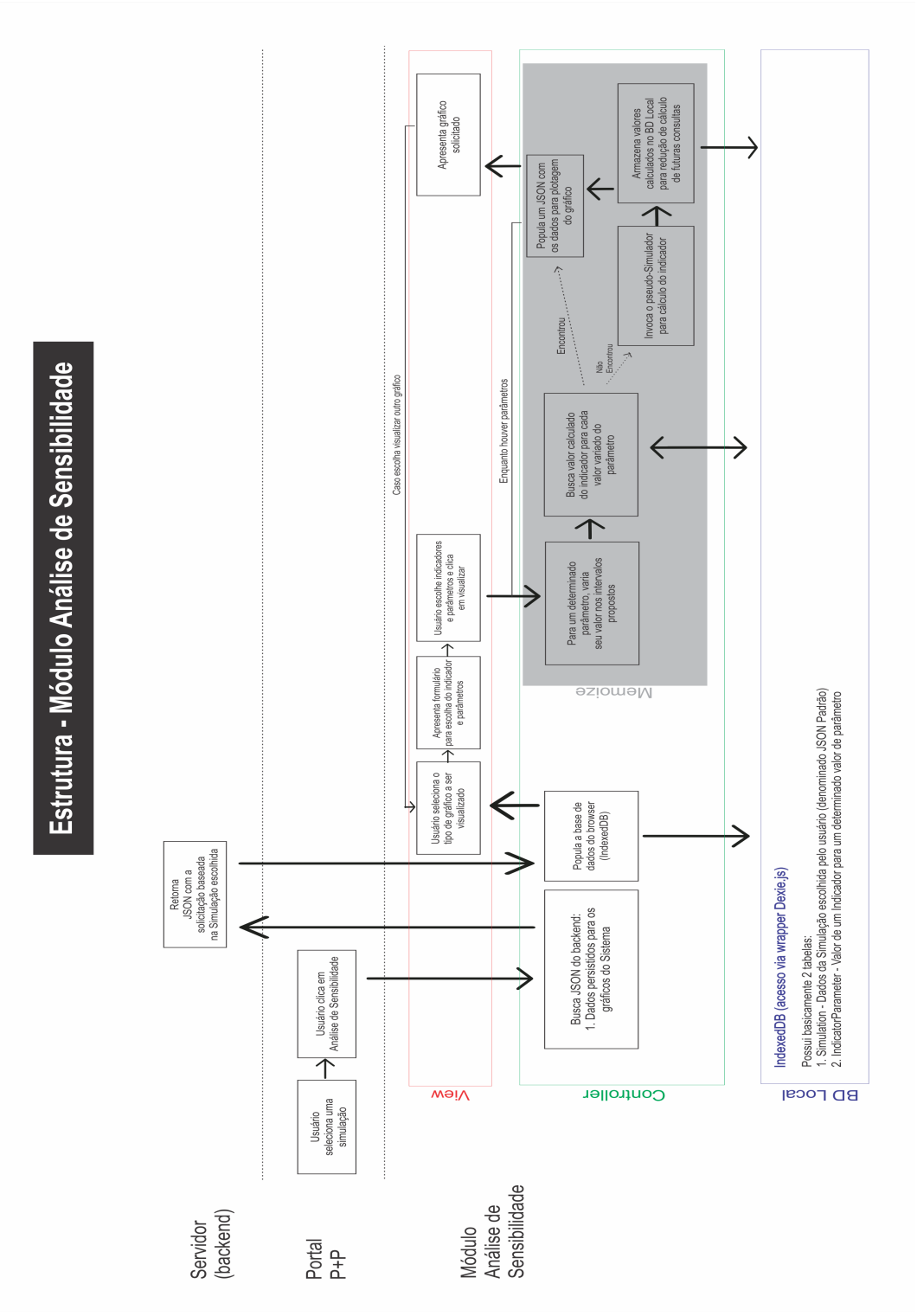

Figura 4: Estrutura do módulo de Análise de Sensibilidade.

- *systemParameters*: valor mínimo (min), máximo (max) e default (std) para cada parâmetro do sistema;

- *indicatorsParameters*: relação de indicadores com seus parâmetros considerados como mais relevantes (*default*) ou comuns (*other*).

Ao selecionar um determinado sistema, é disponibilizado para o usuário a opção de escolher visualizar a análise de sensibilidade para o sistema (parâmetros setados em seus valores *default*) ou simulação (parâmetros calibrados de acordo com a escolha do usuário) escolhido. Ao selecionar uma das opções de gráfico no módulo de análise de sensibilidade, o sistema solicita ao *backend* da P+P a verificação da existência de dados persistidos de cálculo da análise de sensibilidade para os gráficos relativos ao "sistema", com o objetivo de reduzir o tempo de montagem e apresentação destes gráficos. Caso estes dados persistidos existam, são retornados e repassados para o banco de dados local via *wrapper* Dexie.js. Este é o único momento na aplicação em que o servidor é acionado e acessado, de forma que os demais passos são executados localmente no *browser* do usuário (exceto se o usuário retornar para o portal da P+P e escolher um outro sistema para visualização, onde todo o processo é reiniciado).

Na opção da escolha da análise a ser realizada pelo usuário, são apresentados a possibilidade de escolha 4 (quatro) tipos de gráficos (descritos nas próximas subseções) a serem visualizados, sendo 3 (três) relativos ao sistema e 1 (um) relativo à simulação escolhida, sendo eles: Gráfico de tornado do sistema, Gráfico XY (indicador x parâmetro), Distribuição de probabilidade (histograma) e Gráfico de tornado da simulação, respectivamente.

Ao escolher o tipo do gráfico de sua preferência, o usuário é direcionado a um formulário para que seja definido o indicador a ser analisado e o(s) parâmetro(s) relacionados ao indicador escolhido. Cada indicador possui uma lista de parâmetros sugeridos (definidos como "*default*") e outros parâmetros. Se o indicador escolhido possuir sugestões de parâmetros (*default*), estes são apresentados no formulário por padrão, de forma que o usuário escolha utilizá-los na análise de sensibilidade ou selecione outros de sua preferência.

A partir da escolha do indicador e parâmetros e ao clicar no botão visualizar, inicia-se a etapa de cálculos para plotagem dos gráficos. Para cada parâmetro selecionado, estes sofrem uma variação na quantidade de vezes correspondente à cada tipo de gráfico, sendo 100 pontos (que variam em 1% do valor mínimo ao valor máximo do parâmetro) para gráficos do sistema ou 3 pontos (que variam em 1% para cima e para baixo em torno do valor escolhido pelo usuário do parâmetro, onde essa porcentagem é calculada com base na diferença entre o valor mínimo e máximo do parâmetro analisado). Vale ressalta que gráficos de tornado permitem a escolha de até 6 parâmetros, sendo que os demais gráficos permitem a escolha de apenas um parâmetro. As informações descritas acima podem ser visualizadas na tabela 2

Para otimizar o tempo de cálculo para visualização dos gráficos, foi a técnica "*memoize*", que consiste em armazenar cálculos já realizados anteriormente de forma a evitar recálculo. Estes dados já calculados são armazenados no *IndexedDB* (banco de dados

Gráfico	Máximo de indicadores	Máximo de parâmetros	Qtde. de pontos analisados	Limite dos pontos analisados
Tornado da Simulação	1	6	3	+/- 1% a partir do valor escolhido do parâmetro
Tornado do Sistema	1	6	100	valor mínimo e máximo do parâmetro
Indicador x Parâmetro (xy)	1	1	100	valor mínimo e máximo do parâmetro
Histograma	1	1	100 (divididos em 20 faixas de valores)	valor mínimo e máximo do parâmetro

Tabela 2: Caraterísticas dos gráficos disponíveis no módulo

local) e acessados por meio do *wrapper Dexie.js*. Desta forma, sempre que um parâmetro é variado em um determinado ponto, é realizado uma verificação se este valor já foi usado em cálculo anterior que está persistido, recuperando os resultados deste cálculo, se sim. Caso contrário, a função *_calculate*, responsável pela chamada ao Simulador é invocada.

Com os pontos devidamente calculados, é realizada a montagem do *JSON* final para apresentação do gráfico escolhido no moemento pelo usuário. Após a montagem do *JSON* final, o gráfico é apresentado para o usuário com a relação do indicador e parâmetros escolhidos. Algo importante a ser mencionado é que, depois da escolha de uma simulação por parte do usuário, este pode variar entre os gráficos e mudança de parâmetros quantas vezes forem necessárias sem que os dados já calculados sejam perdidos, o que reforça a importância da técnica de *memoize* utilizada. Somente se o usuário voltar para a P+P e escolher um outro sistema ou simulação é que os dados calculados anteriormente serão perdidos.

3.3.1 Gráfico de Tornado da Simulação

O gráfico de tornado de simulação(Figura 5) mostra como um Indicador escolhido varia em torno dos valores escolhidos pelo usuário na simulação para até 6 parâmetros escolhidos.

Um gráfico de tornado pode ser definido como um gráfico de barras, consistindo de um eixo vertical central, barras horizontais à esquerda e à direita deste eixo (SWENSON *et al.*, 2010). As barras à esquerda e direita representam uma variação mínima e máxima de um parâmetro específico em relação à um indicador (no caso do projeto em questão). A ordenação das barras em magnitude decrescente de variação é proposital, o que confere ao gráfico a ideia de um "tornado" (SWENSON *et al.*, 2010).

Componentes do gráfico:

- O eixo do tornado representa o valor do Indicador para a Simulação escolhida pelo Usuário, isto é, o cálculo do Indicador usando os valores definidos pelo Usuário para a Simulação escolhida.

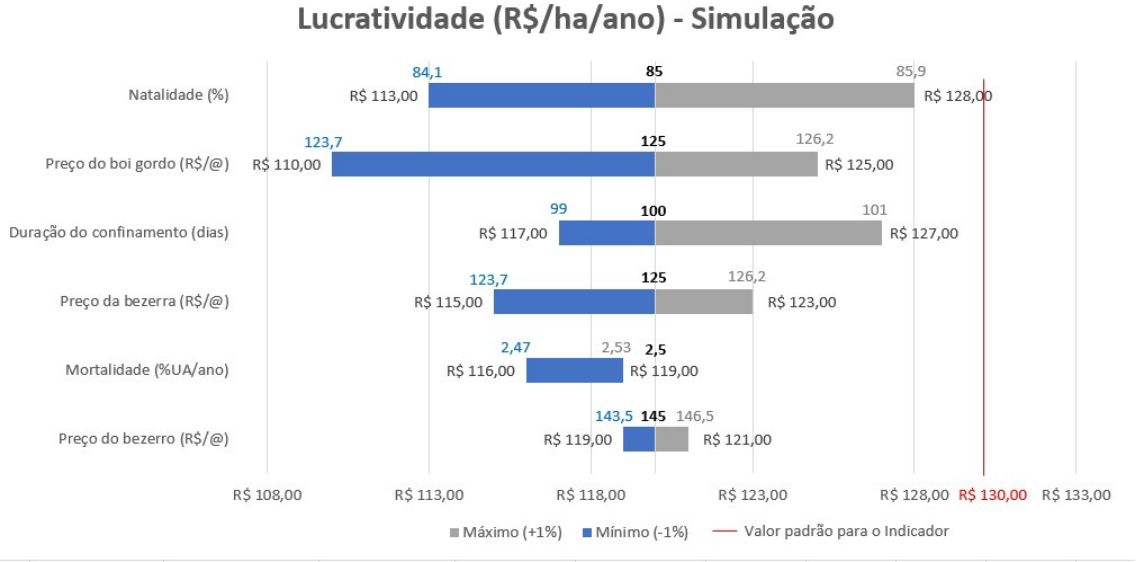

Figura 5: Exemplo de gráfico de tornado da simulação.

- O valor mais à esquerda de cada barra do tornado representa o menor valor encontrado para o Indicador, e o valor mais à direita representa o maior valor encontrado para o Indicador.
- Eixo vertical adicional representa o valor encontrado para o Indicador na Simulação Padrão do Sistema.
- Escala do eixo horizontal na unidade de medida do Indicador.
- Título: nome do indicador seguido de sua unidade de medida, e nome da Simulação.
- Legenda: nome do Parâmetro representado por cada barra seguido de sua unidade de medida.
- Rótulos: valores mínimo e máximo do indicador em cada barra e os valores correspondentes para o Parâmetro representado pela barra. Caso haja mais de um Parâmetro para o mesmo valor do Indicador, cada um dos valores do Parâmetro será exibido.

Forma de cálculo do gráfico:

- Para encontrar o menor e o maior valor de cada indicador na barra de cada parâmetro, varia o Parâmetro com todos os demais Parâmetros fixos no valor do Parâmetro para a Simulação.
- Serão três os pontos para cada parâmetro usados no cálculo: valor do parâmetro para a Simulação, +1% da Variação, -1% da Variação; em que "variação"é a diferença entre o valor máximo e o valor mínimo do parâmetro para o sistema.

3.3.2 Gráfico de Tornado do Sistema

O gráfico de tornado (Figura 6) mostra como um indicador escolhido varia ao longo dos valores permitidos para o Sistema da Simulação escolhida para até 6 parâmetros escolhidos. A principal diferença deste gráfico para o anterior é o fato do gráfico se basear nos valores mínimo, máximo e padrão estabelecidos para o sistema escolhido pelo usuário. Uma outra diferença é a maior quantidade de pontos analisados no gráfico de tornado do sistema, sendo 100 os pontos, variando entre o valor mínimo e máximo dos parâmetros selecionados.

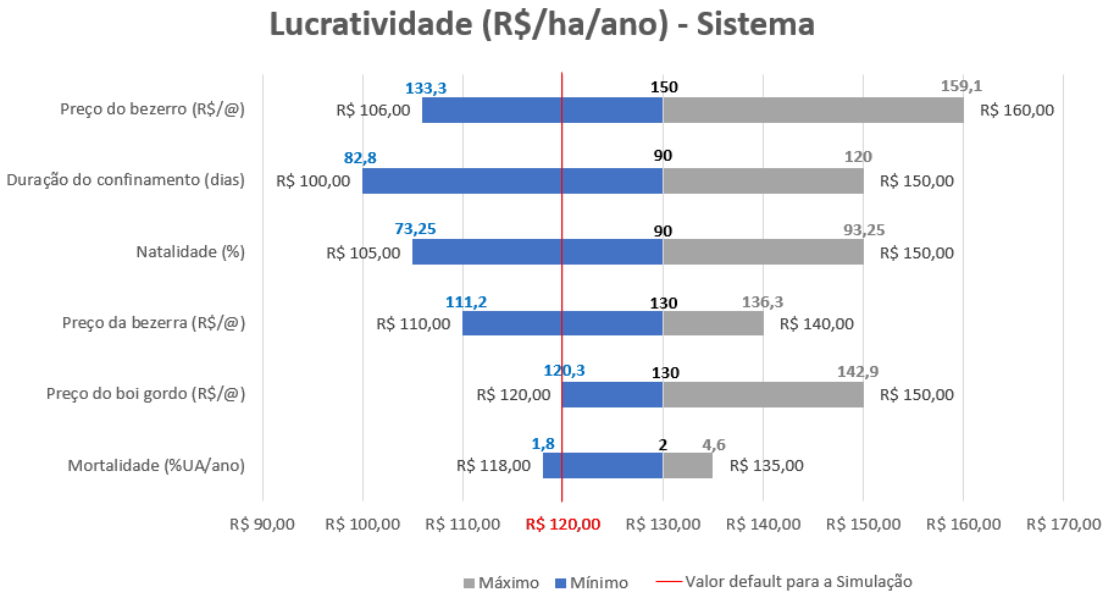

Figura 6: Exemplo de gráfico de tornado do sistema.

Componentes do gráfico:

- O eixo do tornado representa o valor do indicador para a simulação padrão do sistema, isto é, o cálculo do indicador usando os valores padrão para todos os parâmetros do Sistema.
- O valor mais à esquerda de cada barra do tornado representa o menor valor encontrado para o indicador, e o valor mais à direita representa o maior valor encontrado para o indicador.
- Eixo vertical adicional representa o valor encontrado para o indicador na simulação selecionada pelo Usuário.
- Escala do eixo horizontal na unidade de medida do indicador.
- Título: nome do indicador seguido de sua unidade de medida, e nome do sistema.

- Legenda: nome do parâmetro representado por cada barra seguido de sua unidade de medida.
- Rótulos: Valores mínimo e máximo do indicador em cada barra e os valores correspondentes para o Parâmetro representado pela barra. Caso haja mais de um Parâmetro para o mesmo valor do Indicador, cada um dos valores do Parâmetro será exibido.

Cálculo do gráfico:

- Para encontrar o menor e o maior valor de cada Indicador na barra de cada Parâmetro, varia o Parâmetro, com todos os demais Parâmetros fixos no valor Padrão do Parâmetro para o Sistema.
- Serão cem os pontos para cada Parâmetro usados no cálculo: indo do valor Mínimo do Parâmetro para o Sistema ao valor Máximo do Parâmetro para o Sistema, de 1% em 1% desta faixa de valores.

3.3.3 Gráfico Indicador x Parâmetro

Para uma Simulação selecionada pelo Usuário no Portal da P+P, exibe um gráfico demonstrando como um Indicador varia em relação à um Parâmetro ao longo dos valores permitidos para o Sistema da Simulação (Figura 7).

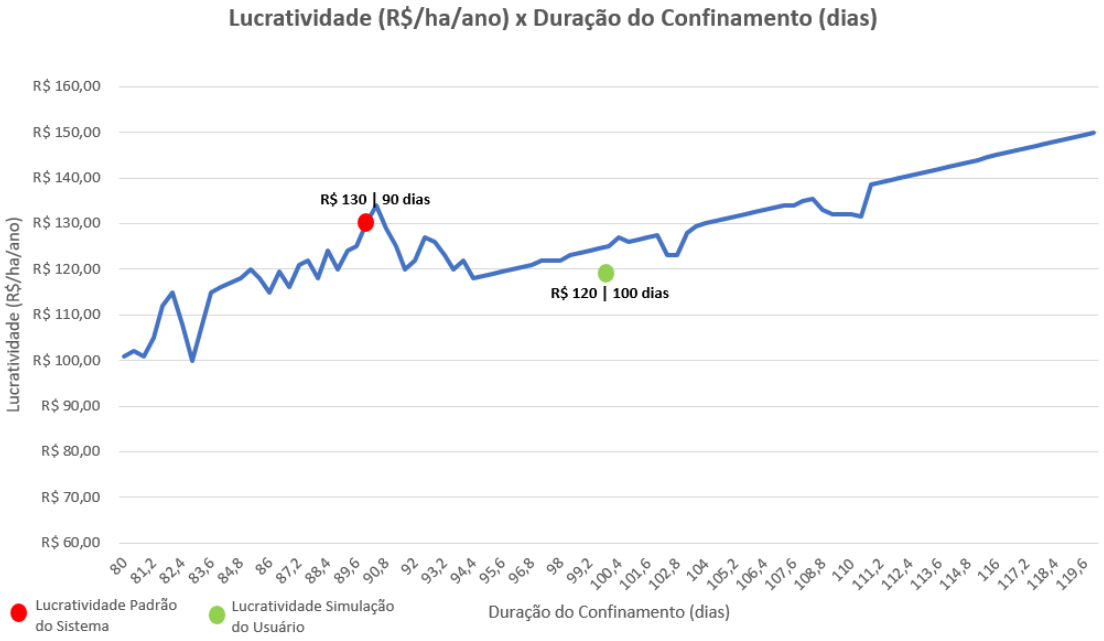

Figura 7: Exemplo de gráfico Indicador x Parâmetro (XY).

Componentes do gráfico:

- Escala do eixo vertical, na unidade de medida do Indicador.
- Escala do eixo horizontal na unidade de medida do Parâmetro, indo do valor Mínimo ao Máximo.
- Título: nome do Indicador seguido de sua unidade de medida e nome do Parâmetro seguido de sua unidade de medida, seguidos de nome do tipo do gráfico.

Cálculo do gráfico:

- São 100 pontos: indo do valor Mínimo do Parâmetro para o Sistema ao valor Máximo do Parâmetro para o Sistema, de 1% em 1% desta faixa de valores.

Dados para o gráfico:

Os dados para este gráfico são um subconjunto dos dados de análise de sensibilidade persistido para o Tornado do Sistema, exceto a marcação que indica o valor padrão do indicador x parâmetro da simulação escolhida pelo usuário.

3.3.4 Histograma

O gráfico de histograma mostra a distribuição de probabilidade de um Indicador supondo distribuição de probabilidade triangular para um parâmetro escolhido (Figura 8) .

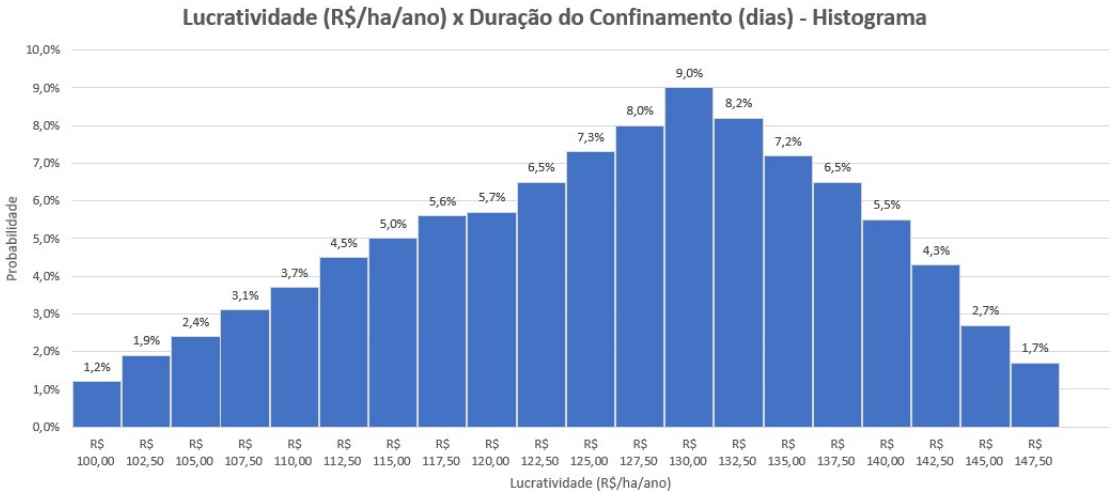

Figura 8: Exemplo de histograma de probabilidade (indicador x parâmetro).

Componentes do gráfico:

- Escala do eixo vertical de 0 a 100%.

- Escala do eixo horizontal na unidade de medida do Indicador.
- Título: nome do Indicador seguido de sua unidade de medida e nome do Parâmetro seguido de sua unidade de medida.

Cálculo do gráfico:

- São 100 pontos usados no cálculo do histograma: indo do valor Mínimo do Parâmetro para o Sistema ao valor Máximo do Parâmetro para o Sistema, de 1% em 1% desta faixa de valores.
- Vinte colunas do histograma representam 20 faixas de valores encontrados para o Indicador.

Dados para o gráfico:

Os dados para este gráfico podem ser calculados a partir dos dados do gráfico XY.

3.4 Estrutura do código

A aplicação web foi desenvolvida utilizando componentes *Vue.js* já descritos nas seções anteriores em conjunto com a biblioteca *D3.js* para plotagem dos gráficos propostos. Para atingir o objetivo final, vários componentes foram desenvolvidos, conforme podem ser observados na figura 9. Os componentes foram divididos em "*forms*" (relativos a implementações e apresentação dos formulários), "*graph*" (referentes a apresentação dos gráficos no *browser*) e "extra" (relativos a funções simuladas de funcionalidades da P+P, visto que sua atualização ainda não foi concluída até o presente momento - vide seção Limitações no capítulo Conclusões).

Além dos componentes, o sistema possui uma "*store*" (que trata-se de um *container* que mantém o estado geral da aplicação) com 3 módulos desenvolvidos: *forms* (dados e funções relacionadas a disponibilidades de dados para o formulário, tratamento destes dados e distribuição de chamadas às funções de cálculo, etc.), *graphs* (dados já calculados onde o *JSON* para plotagem dos gráficos é armazenado). Dentro dos módulos da *store* responsáveis por gerenciar todo o estado da aplicação, vale destacar a função "*searchRelation*" presente no módulo *forms* e responsável por resgatar os dados selecionados no formulário, invocar as funções *memoize* para cálculo da análise de sensibilidade e alimentar o *JSON* para plotagem dos gráficos da aplicação. Parte desta função pode ser observada na figura 10 e, embora a figura esteja comentada em detalhes, é importante detalharmos algumas partes desta função. A função recebe um objeto "*form*" que contém o tipo de gráfico (*system* ou *simulation*) e o modelo de gráfico (tornado, xy, histograma), pois cada um dos gráficos possui particularidades em sua montagem final. A função "*calculation*"

Figura 9: Estrutura de arquivos.

é responsável por retornar os valores do indicador para os 100 pontos analisados de um determinado parâmetro. Ao selecionar 6 parâmetros (no caso dos gráficos de tornado), esta função é invocada por 6 vezes. Após o retorno dos 100 valores do indicador para a variação do parâmetro, a função "*maiorMenor*" é chamada (somente para gráficos de tornado) e possui como retorno o valor mínimo e o máximo encontrados do indicador para as variações do parâmetro. Após isto, as funções "*insertTornado*" (para gráficos de tornado), "*setParametersXY*" e "*setParametersHistogram*", são responsáveis por montar o *JSON* final para a plotagem do gráfico proposto.

3.5 *Memoize* no desenvolvimento

Para otimizar o cálculo dos dados relativos aos gráficos de sistema, foi utilizado princípios de técnicas de *memoize* que, como já explicados em capítulos anteriores, consistem em memorizar cálculos já realizados, a fim de não precisar executá-los novamente. A função "*calculation*" é a responsável por fazer esta análise e pode ser observada na figura 11. As primeiras informações da função dizem respeito a verificação se existe realmente um indicador e um parâmetro passado como parâmetro da função. Após isto, a função "*calculeDistance*" é chamada para calcular a distância entre os pontos do parâmetro (1%) a serem analisados, onde a distância se baseia na diferença do valor mínimo e máximo do parâmetro divididos por cem. Caso o tipo da análise seja "*system*" (gráficos de sistema),

```
//FUNÇÃO RESPONSÁVEL POR ANALISAR OS DADOS SELECIONADOS NO FORMULÁRIO,
//CHAMADA DA FUNÇÃO DE CÁLCULO DA ANÁLISE DE SENSIBILIDADE, E ALIMENTAÇÃO
//DOS DADOS PARA PLOTAGEM DO GRÁFICO
searchRelation: async function ({ state, commit, dispatch }, form) {
  //O INDICADOR SELECIONADO ESTÁ EM STATE.INDICATORSELECTED
  //OS PARÂMETROS EM STATE.PSELECTED

  //resgata o JSON padrão da Simulação
  let Simulation = await db.searchSimulation(state.idSystem);

  //caso o typo do formulário preenchido seja de Sistema
  if (form.type == "system") {
    //verifica o tipo do gráfico de sistema
    if (form.model == "tornado") { //GRÁFICO DE TORNADO DO SISTEMA
      //array temporário para armazenar os dados do gráfico final
      let tornadoGraph = []; //array dos valores para o gráfico

      //loop para todos os parâmetros selecionados (máximo 6)
      for (var i = 0; i < state.pSelected.length; i++) {
        //chamada à função de cálculo da análise de sensibilidade com retorno dos valores para o indicador naquele parâmetro
        let valueIndicators = await memoize.calculation(Simulation, form.type, state.indicatorSelected, state.pSelected[i])
        //calculo dos menores e maiores valores para inserção no gráfico
        let values = helpersimulation.maiorMenor(valueIndicators);
        let min = Simulation.simulationData.systemParameters[state.pSelected[i]].min
        let max = Simulation.simulationData.systemParameters[state.pSelected[i]].max
        let d = memoize.calculeDistance(min, max);
        //função que preenche os menores e maiores valores de cada parâmetro
        helpersimulation.insertTornado(tornadoGraph, state.pSelected[i], values, min, d)
      }
      //setando os dados para o gráfico
      dispatch("setDataGraph", { type: "system", model: "tornado", dataGraph: tornadoGraph })

    } else {
      //chamada à função de cálculo da análise de sensibilidade com retorno dos valores para o indicador naquele parâmetro
      let valueIndicators = await memoize.calculation(Simulation, form.type, state.indicatorSelected, state.pSelected)
      if (form.model == "xy") { // GRÁFICO XY
        let xy = []; //array dos valores para o gráfico
        let min = Simulation.simulationData.systemParameters[state.pSelected].min
        let max = Simulation.simulationData.systemParameters[state.pSelected].max
        let d = memoize.calculeDistance(min, max);
        xy = helpersimulation.setParametersXY(valueIndicators, min, d)
        //setando os dados para o gráfico
        dispatch("setDataGraph", { type: "system", model: "xy", dataGraph: xy })
      } else { //HISTOGRAMA
        let histograma = []
        let max = Math.max(...valueIndicators)
        let min = Math.min(...valueIndicators)
        let d = memoize.calculeDistance(min, max) * 5; //apenas 20 pontos
        histograma = helpersimulation.setParametersHistogram(valueIndicators, min, d)
        //setando os dados para o gráfico
        dispatch("setDataGraph", { type: "system", model: "histograma", dataGraph: histograma })
      }
    }
```

Figura 10: Função *searchRelation*.

há um *loop* de cem pontos a serem analisados, sendo que do contrário (*simulation*) apenas 3 pontos (mais próximos ao valor escolhido para o parâmetro) serão analisados. Para cada ponto a ser verificado do parâmetro, a função "*searchData*" é chamada e é responsável por verificar se o cálculo para a combinação de indicador e valores dos parâmetros já foi realizado e está armazenado, sendo que caso já tenha sido calculado, não é necessário realizar uma nova chamada ao Simulador. Caso não exista o cálculo já armazenado, a função "*_calculate*" é invocada. A função "*_calculate*" é a própria chamada ao Simulador da P+P.

Caso encontre um conjunto exatamente igual, a variável *valueindicator* é devolvida para dispensar a chamada ao Simulador e, consequentemente o cálculo novamente do conjunto. Esta técnica de *memoize* reduz o tempo de resposta para apresentação do gráfico, principalmente gráficos de sistema que necessitam do cálculo do indicador para 100 pontos de cada parâmetro, para até seis parâmetros.

Os dados já calculados são armazenados no *IndexedDB* através do *wrapper Dexie.js*, onde cada linha armazenada possui a seguinte estrutura: *id* (chave gerada auto-

maticamente com auto-incremento a cada inserção), *idSystem* (id do sistema analisado), *valueindicator* (valor calculado para o indicador), *indicator* (nome do indicador calculado), *parameter* (nome do parâmetro calculado) e *valueParameter* (valor analisado do parâmetro).

```
//Simulation: JSON padrão
//type: system ou simulation
//indicator: indicador a ser calculado
//parameter: parâmetro a ser variado
async calculation(Simulation, type, indicator, parameter) {
    this.valueIndicators = [];
    if (indicator != null) {
        if (parameter != null) {

            let min = Simulation.simulationData.systemParameters[parameter].min
            let max = Simulation.simulationData.systemParameters[parameter].max

            let d = this.calculeDistance(min, max)
            if (type == 'system') { //100 PONTOS DO MINIMO AO MÁXIMO
                //controle corresponde ao valor modificado do parâmetro analisado
                //d corresponde à 1% da diferença entre o mínimo e o máximo do parâmetro
                let controle = min;
                //Invoca a função de Simulação 100 vezes para analisar o menor e maior valor
                for (var j = 0; j < 100; j++) {
                    //busca na base de dados local a combinação
                    let vIndicator = await db.searchData({"indicator": indicator, "parameter": parameter, "valueParameter": controle}, type)
                    if (vIndicator != undefined) {
                        //PASSO 2: SE EXISTE O VALOR, RETORNA
                        this.valueIndicators.push(vIndicator.valueindicator)
                        controle += d; //incrementa o valor a ser analisado do parâmetro
                    } else {
                        this.valueIndicators.push(this._calculate(Simulation, type, indicator, parameter, controle))
                        controle += d; //incrementa o valor a ser analisado do parâmetro
                    }

                }
            } else { //3 PONTOS PRÓXIMOS
                //controle corresponde ao valor modificado do parâmetro analisado
                //d corresponde à 1% da diferença entre o mínimo e o máximo do parâmetro
                let controle = Simulation.simulationData.parameters[parameter] - d;
                //loop para verificar 3 pontos da Simulação
                for (var l = 1; l <= 3; l++) {
                    let vIndicator = await db.searchData({"indicator": indicator, "parameter": parameter, "valueParameter": controle}, type)
                    if (vIndicator != undefined) {
                        //PASSO 2: SE EXISTE O VALOR, RETORNA
                        this.valueIndicators.push(vIndicator.valueindicator)
                    } else {
                        //PASSO 3: SE NÃO EXISTE O VALOR, CHAMA A FUNÇAO DE CÁLCULO
                        this.valueIndicators.push(this._calculate(Simulation, type, indicator, parameter, controle))
                    }
                    controle += d; //incrementa o valor a ser analisado do parâmetro
                }
            }
            return this.valueIndicators
```

Figura 11: Função *calculation* que faz uso da técnica *Memoize*.

3.6 Formato dos dados

Durante o processo de cálculo do valor do indicador para cada ponto de um parâmetro selecionado, cada valor calculado é armazenado em uma tabela de pesquisa que tem a estrutura em cada linha de acordo com a figura 12.

Após a finalização do processo de cálculo de cada gráfico, um *JSON* é montado com os dados necessários para plotagem. Cada gráfico possui sua particularidade dos dados e são apresentados abaixo. Os gráficos de tornado do sistema e simulação possuem a mesma estrutura, visto que são gráficos semelhantes. Esta estrutura pode ser vista na figura 13.

O gráfico indicador X parâmetro precisa da informação dos 100 pontos calculados para plotagem. o formato final do *JSON* para o gráfico pode ser visto na figura 14. Já a

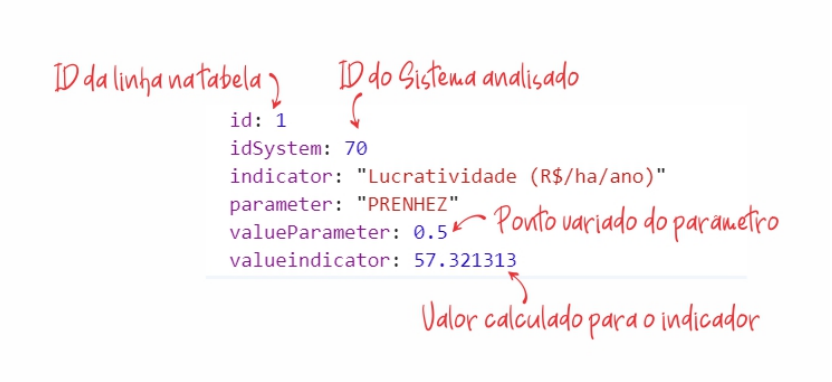

Figura 12: Formato de dados da tabela de pesquisa utilizada no *memoize*.


```
{
    "titleTornado":"Toneladas de Carne Produzidas por Ano - Simulação",
    "default":77,"defaultSystem":80,
    "indicator":"Toneladas de Carne Produzidas por Ano",
    "parameters":["CICLO","NATALIDADE","PARTO_MACHO","DIAGNOSE","CONFINAMENTO","MONTA"],
    "dadosInfo":{
        "Gráfico de Tornado da Simulacao":[
            {"param":"NATALIDADE","valueIndicator":77.10695225648799,"valueParam":0.51,"variation":"max"},
            {"param":"NATALIDADE","valueIndicator":76.88918933572799,"valueParam":0.5,"variation":"min"},
            {"param":"DIAGNOSE","valueIndicator":77,"valueParam":30.6,"variation":"max"},
            {"param":"DIAGNOSE","valueIndicator":76.58939395753457,"valueParam":30,"variation":"min"},
            {"param":"MONTA","valueIndicator":77.59316391636723,"valueParam":61.2,"variation":"max"},
            {"param":"MONTA","valueIndicator":77,"valueParam":60.6,"variation":"min"},
            {"param":"CONFINAMENTO","valueIndicator":77.365673021714,"valueParam":103.52,"variation":"max"},
            {"param":"CONFINAMENTO","valueIndicator":76.62466191087398,"valueParam":103,"variation":"min"},
            {"param":"PARTO_MACHO","valueIndicator":77.5357117515365,"valueParam":0.410003,"variation":"max"},
            {"param":"PARTO_MACHO","valueIndicator":76.45112735987651,"valueParam":0.4,"variation":"min"},
            {"param":"CICLO","valueIndicator":78.39868433808716,"valueParam":320,"variation":"max"},
            {"param":"CICLO","valueIndicator":77,"valueParam":320.45,"variation":"min"}
        ]
    }
}
```


Figura 13: JSON final para plotagem do gráfico de tornado da simulação.

estrutura de dados para plotagem do histograma pode ser vista na figura 15 onde, vinte faixas de valores são armazenadas (xField) com suas respectivas probabilidades (yField).

3.7 Os gráficos

Os gráficos foram desenvolvidos usando a biblioteca *D3.js* de manipulação de documentos com base em dados. Os gráficos de tornado, tanto de sistema quanto de simulação, possuem a mesma estrutura e mesma função implementada, sendo que a única diferença está no eixo central, onde um representa o valor do indicador com todos os parâmetros no valor padrão e, o outro representa o valor do indicador com todos os parâmetros no valor da simulação escolhida pelo usuário.

A figura 16 representa uma das partes principais da função de plotagem dos gráficos de tornado, onde é feita a verificação se o valor pertence à barra da esquerda ou direita do gráfico, posicionando as barras no lugar correto, conforme pode ser observado no resultado

```
{
    "titleLine":"Lucratividade (R$/ha/ano) X DIAGNOSE",
    "legendLineX":"DIAGNOSE",
    "legendLineY":"Lucratividade (R$/ha/ano)",
    "dadosLine":[
        {"yField":57.321313,"xField":"30.00"},
        {"yField":58.13401999912001,"xField":"30.60"},
        {"yField":58.94839502848001,"xField":"31.20"},
        {"yField":59.764438088080006,"xField":"31.80"},
        {"yField":60.582149177919995,"xField":"32.40"},
        {"yField":61.40152829800002,"xField":"33.00"},
        {"yField":62.22257544832001,"xField":"33.60"},
        {"yField":63.045290628880025,"xField":"34.20"},
        {"yField":63.869673839680026,"xField":"34.80"}
        //... 100 pontos no total
        {"yField":145.8709206071197,"xField":"89.40"}
    ],
    "defaultSimulation":["62.40",103],
    "defaultSystem":["60.60",100]
}
```


Figura 14: JSON final para plotagem do gráfico indicador x parâmetro.


```
{
    "titleBar": "Distribuição de Probabilidade - Lucratividade (R$/ha/ano) X CONFINAMENTO",
    "indicator": "Lucratividade (R$/ha/ano)",
    "parameter": "CONFINAMENTO",
    "legendXBar": "Lucratividade (R$/ha/ano)",
    "legendYBar": "Probabilidade",
    "dadosBar":[
        {"yField":"0.404","xField":"59.56"}, {"yField":"0.943","xField":"61.80"},
        {"yField":"2.425","xField":"64.04"}, {"yField":"3.638","xField":"66.28"},
        {"yField":"4.851","xField":"68.52"}, {"yField":"3.908","xField":"70.76"},
        {"yField":"5.919","xField":"73.00"}, {"yField":"5.703","xField":"75.24"},
        {"yField":"7.267","xField":"77.48"}, {"yField":"5.197","xField":"79.72"},
        {"yField":"6.593","xField":"81.96"}, {"yField":"4.692","xField":"84.20"},
        {"yField":"5.919","xField":"86.44"}, {"yField":"6.858","xField":"88.68"},
        {"yField":"5.053","xField":"90.92"}, {"yField":"5.775","xField":"93.16"},
        {"yField":"6.136","xField":"95.40"}, {"yField":"6.064","xField":"97.64"},
        {"yField":"6.064","xField":"99.88"}, {"yField":"6.593","xField":"102.12"}
    ],
}
```


Figura 15: JSON final para plotagem do histograma.

final apresentado na figura. Analisando a função de plotagem do gráfico XY (indicador x parâmetro), podemos destacar o trecho apresentado na figura 17 que é responsável por criar os pontos no gráfico e interligá-los por meio de linhas. Da mesma forma, a figura 18 representa o histograma com a distribuição de probabilidade de um determinado indicador x parâmetro, onde o trecho de código em destaque representa a plotagem das barras com sua porcentagem logo acima. Ainda sobre o histograma, o código disponibilizado na figura 19 busca garantir a distribuição triangular do gráfico.

```
133         //POSICIONAMENTO DAS BARRAS
134         bar
135         .enter().append("rect").attr("class", function (d) {return ("bar bar--"+(d.valueIndicator < eixoDefault ? "negative" : "positive"));}) /
136         .attr("x", function (d) { return x(Math.min(eixoDefault, d.valueIndicator));})
137         .attr("y", function (d) { return y(d.param);})
138         .attr("width", function (d) { return Math.abs(x(d.valueIndicator) - x(eixoDefault));}) //ONDE COMEÇA A DIVISÃO CENTRAL (EIXO)
139         .attr("height", y.bandwidth())
140         .on("mousemove", function (d) {
141             //ADICIONANDO TOOLTIP COM INFORMAÇÕES DA BARRA
142             tooltip
143               .style("left", d3.event.pageX - 50 + "px").style("top", d3.event.pageY - 70 + "px")
144               .style("display", "inline-block").html(title +": " + d.valueIndicator + "<br>" + d.param + ": " + d.valueParam);})
145               .on("mouseout", function () {tooltip.style("display", "none");}); //REMOVENDO TOOLTIP
```

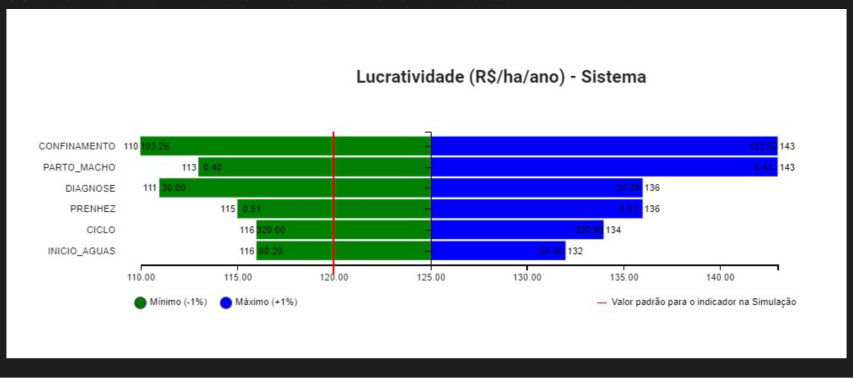

Figura 16: Função para gráfico de tornado.

```
154         // Adicionando pontos
155         chart
156           .selectAll("pontos").data(this.dados).enter().append("circle").attr("class", "pontos")
157           .attr("stroke", "none").attr("cx", function (d) { return x(d.xField); })
158           .attr("cy", function (d) { return y(d.yField); })
159           .attr("r", 3)
160           .on("mousemove", function (d) {
161             tooltip
162               .style("left", d3.event.pageX - 50 + "px")
163               .style("top", d3.event.pageY - 70 + "px")
164               .style("display", "inline-block")
165               .html(d.yField + " | " + d.xField);
166           })
167           .on("mouseout", function () {
168             tooltip.style("display", "none");
169           });
```

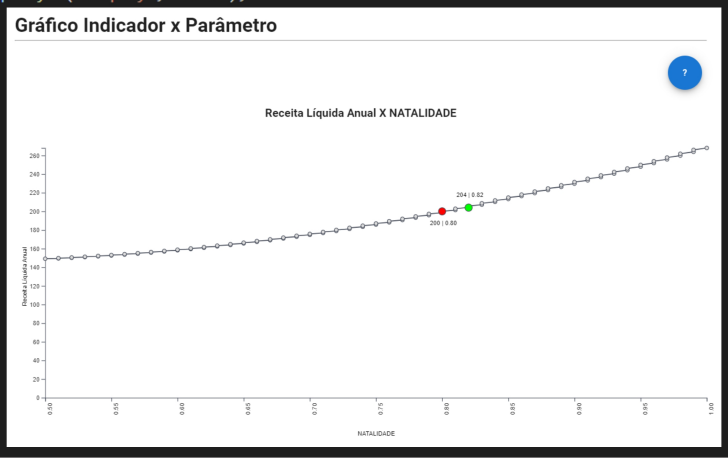

Figura 17: Função para gráfico indicador x parâmetro.

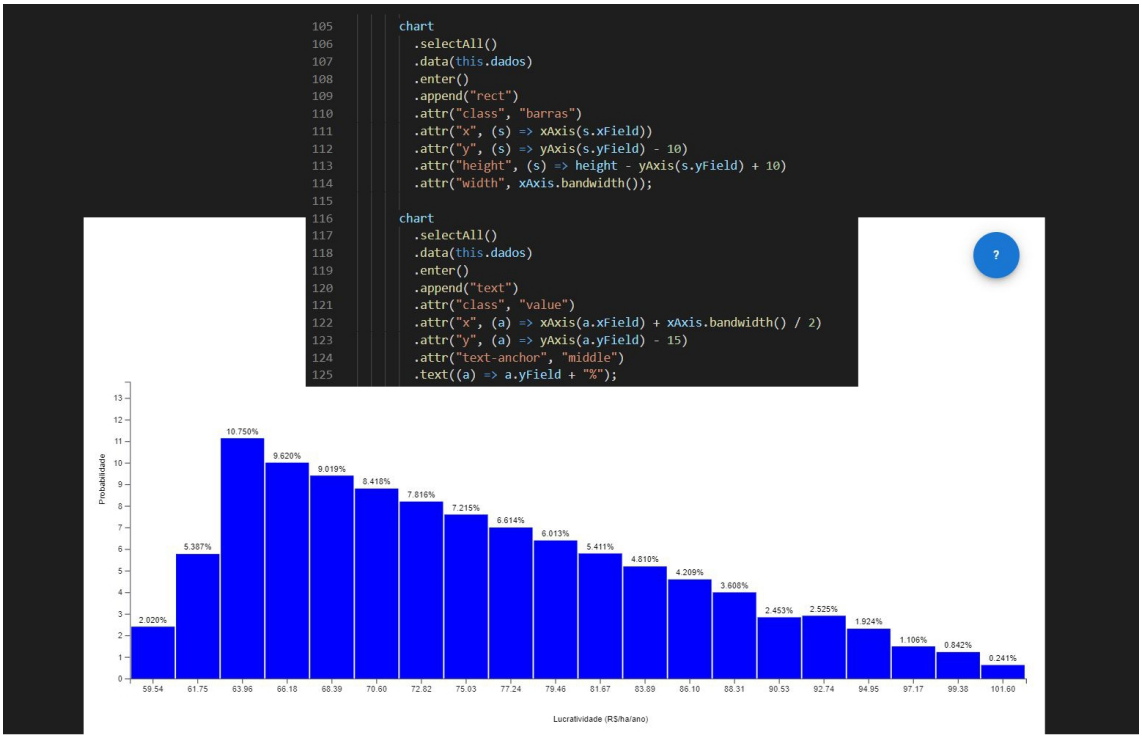

Figura 18: Função para histograma.

```
// O máximo da distribuição triangular ocorre no valor std do Parâmetro, aproximado para a posição xstd
// e a área do triângulo dá a probalidade total, i.e., 1, logo, usando a fórmula da área do triângulo para calcular a sua altura em xstd:
triang[xstd] = 2 / y.length;
// Para calcular a distribuição triangular do lado esquerdo, usamos a equação de reta ascendente começando em zero e indo até triang[xtsd]:
for (x = i = 0, ref = xstd - 1; (0 <= ref ? i <= ref : i >= ref); x = 0 <= ref ? ++i : --i) {
  triang[x] = triang[xstd] * x / xstd;
}
// Para calcular a distribuição triangular do lado direito, usamos a equação de reta descendente começando em triang[xtsd] e indo até zero:
for (x = j = ref1 = xstd + 1, ref2 = y.length - 1; (ref1 <= ref2 ? j <= ref2 : j >= ref2); x = ref1 <= ref2 ? ++j : --j) {
  triang[x] = triang[xstd] - triang[xstd] * (x - xstd) / (y.length - 1 - xstd);
}
// Encontra o menor e maior valores de y.
ymin = Math.min(...y);
ymax = Math.max(...y);

// Define a largura das colunas de probabilidade.
ygap = (ymax - ymin) / ncols;
// Acumula as probabilidades de y em colunas
for (x = k = 0, ref3 = y.length - 1; (0 <= ref3 ? k <= ref3 : k >= ref3); x = 0 <= ref3 ? ++k : --k) {
  c = Math.floor((y[x] - ymin) / ygap); // Descobre em que coluna y[x] deve cair
  // Na coluna do histograma em que y[x] caiu, acumula a probabilidade triagular do valor do Parâmetro que corresponde ao valor de y.
  if (c < ncols)
    prob[c] += triang[x];
}
// Ajusta as probabilidades para que o total seja 1 e salva pares com o centro da faixa de valores de y e a probabilidade respectiva.
total = 0
for (i = 0; i < prob.length; i++) {
  total += prob[i];
}

for (c = l = 0, ref4 = ncols - 1; (0 <= ref4 ? l <= ref4 : l >= ref4); c = 0 <= ref4 ? ++l : --l) {
  let v = ymin + (c + 1) * ygap / 2
  let p = (prob[c] / total) * 100
  hist.push({
    "yField": p.toFixed(3),
    "xField": v.toFixed(2)
  });
```

Figura 19: Parte da função de distribuição triangular.

4 A ferramenta

Este capítulo descreve o passo a passo da utilização do módulo de análise de sensibilidade, bem como apresenta cada tela disponível na aplicação.

4.1 Tela inicial

O módulo de análise de sensibilidade é iniciado a partir do momento em que o usuário realiza uma determinada simulação na P+P e lhe é apresentada a possibilidade de visualizar a análise de sensibilidade da simulação escolhida. Ao clicar em um determinado sistema e simulá-lo, é dada a opção do usuário escolher visualizar um dos quatro gráficos disponíveis (Figura 20).

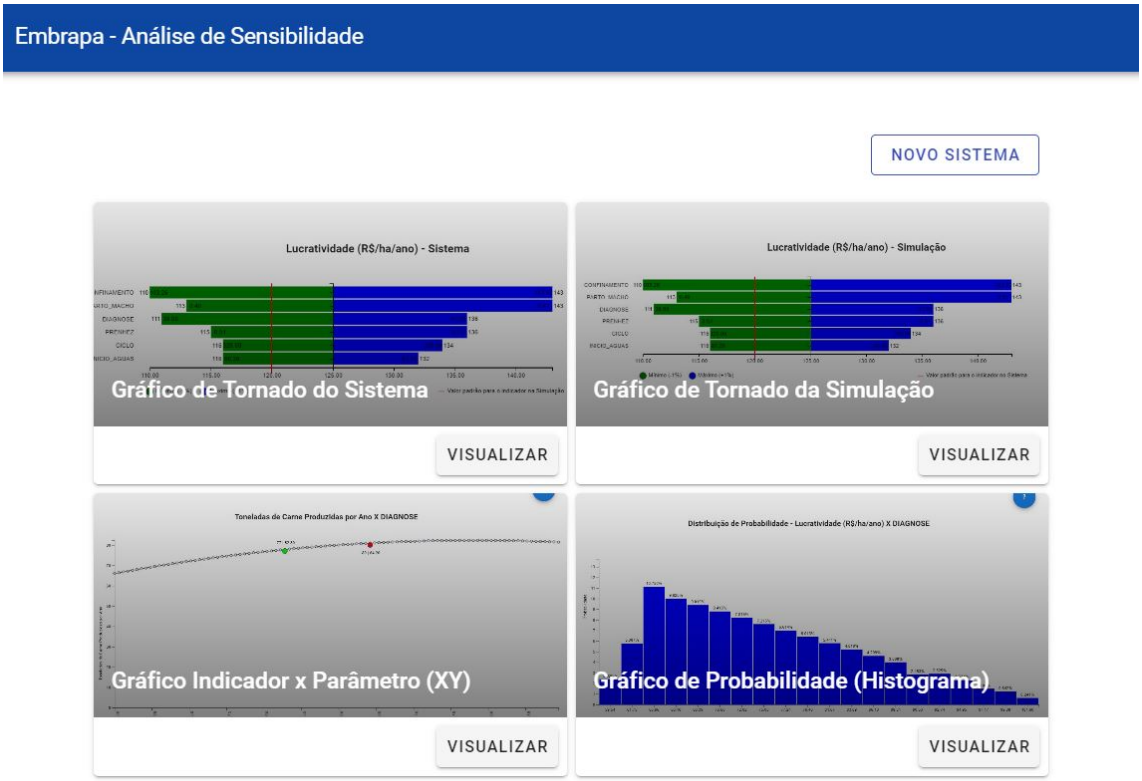

Figura 20: Tela inicial - Escolha dos gráficos.

4.2 Formulários de seleção

A partir da escolha do gráfico a ser produzido, o usuário é direcionado para visualização da escolha do indicador a ser visualizado e dos parâmetros. Para os gráficos de tornado é possível a escolha de até 6 parâmetros conforme figura 21 e, para os gráficos

indicadorXparâmetro e histograma apenas um parâmetro, como pode ser observado na figura 22.

Para os gráficos de tornado, tanto de sistema quanto de simulação, é possível visualizar uma lista de parâmetros sugeridos para um determinado indicador selecionado, definidos como *default* pelo administrador da P+P. A partir da escolha do indicador, estes parâmetros (se existirem) são disponibilizados no campo de seleção como já selecionados. O usuário pode completar sua lista, excluir algum parâmetro *default* ou até desabilitar a opção de visualização de sugestões. Como os gráficos indicadorXparâmetro e histograma possuem apenas um parâmetro, não existe uma lista de sugestões pré-selecionada.

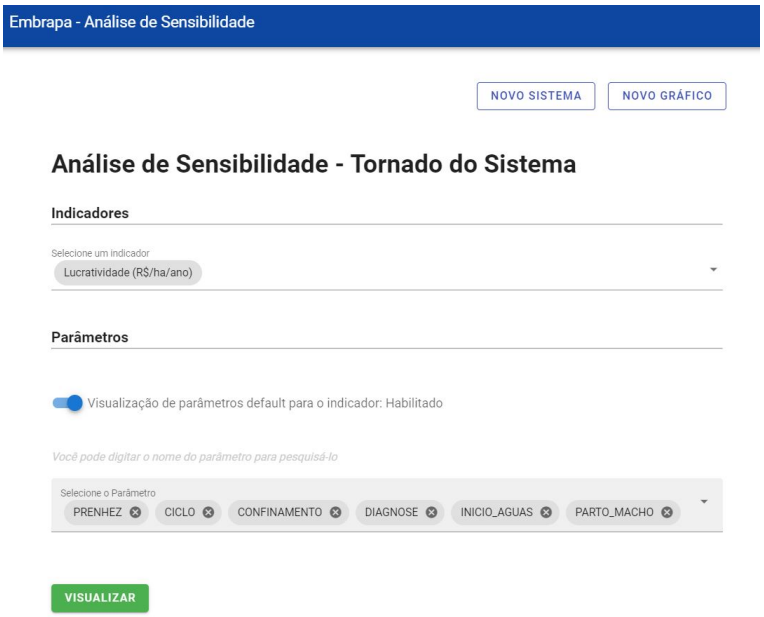

Figura 21: Formulário de escolha de indicadores e parâmetros - Gráfico de Tornado.

Figura 22: Formulário de escolha de indicadores e parâmetros - Gráfico indicadorXparâmetro.

4.3 Mensagem de Erro

Ao tentar selecionar mais de 6 parâmetros, um *modal* com uma mensagem de erro é apresentada (figura 23), impedindo que mais parâmetros sejam selecionados. O formulário também só é submetido caso tenha indicador selecionado e parâmetros suficientes (entre 1 e 6 para gráficos de tornado e apenas 1 para os demais gráficos) selecionados.

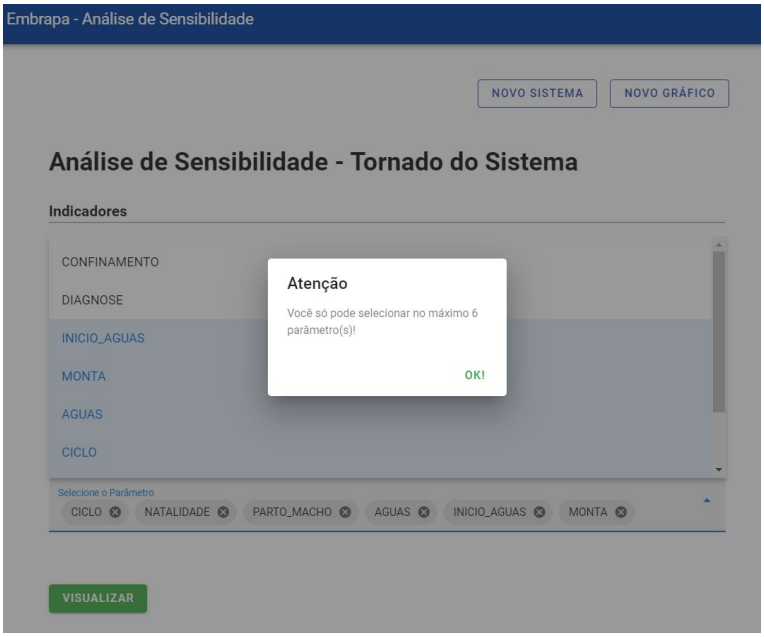

Figura 23: Mensagem de erro.

4.4 Apresentação dos gráficos

Após selecionar o indicador e os parâmetros a serem analisados, o usuário deverá clicar no botão visualizar. A partir deste clique o método "*searchRelation*"é chamado (este método já foi comentado no capítulo anterior), os cálculos e chamadas ao Simulador da P+P são realizados caso necessário, e os gráficos são apresentados a partir daí. As figuras 24, 25 e 26 representam os gráficos de tornado, indicadorXparâmetro (indicadorXparâmetro) e histograma, respectivamente.

4.5 Ajuda sobre os gráficos

Caso o usuário tenha alguma dúvida sobre a leitura do gráfico, é possível visualizar através de um modal a descrição e explicação do gráfico, como pode ser observado nas figuras 27 e 28

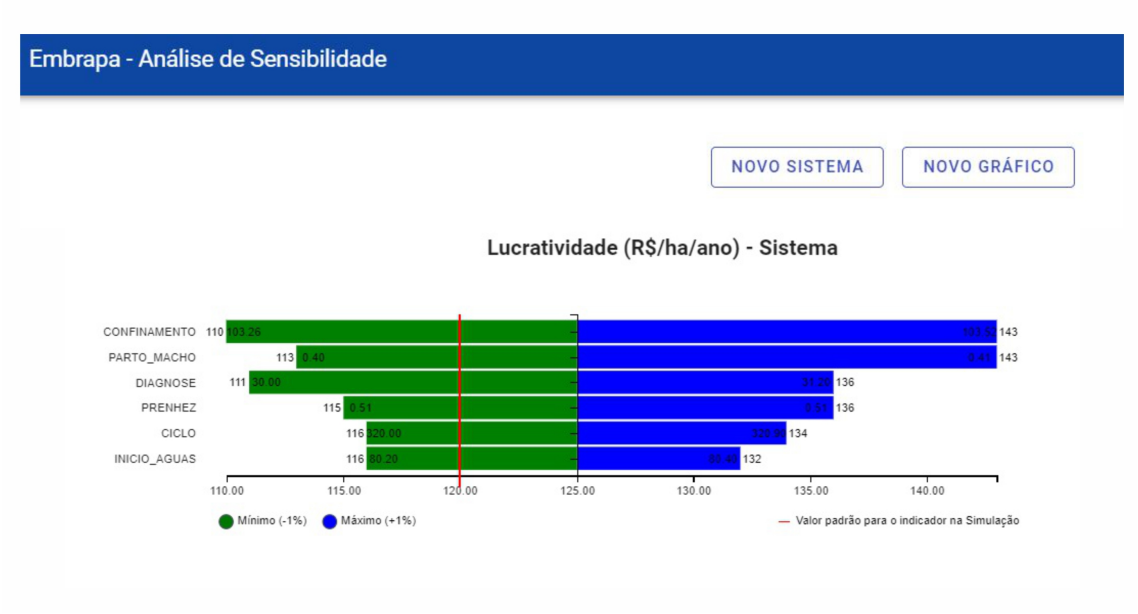

Figura 24: Gráfico de Tornado.

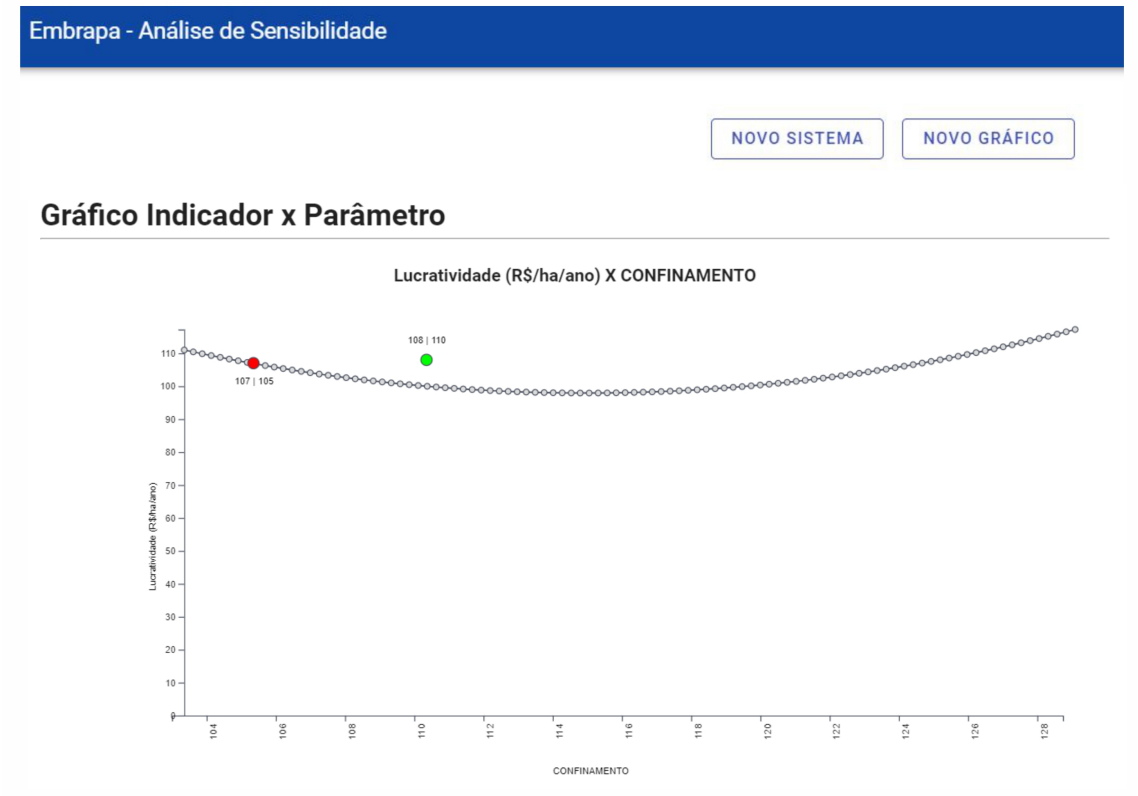

Figura 25: Gráfico indicadorXparâmetro (indicadorXparâmetro).

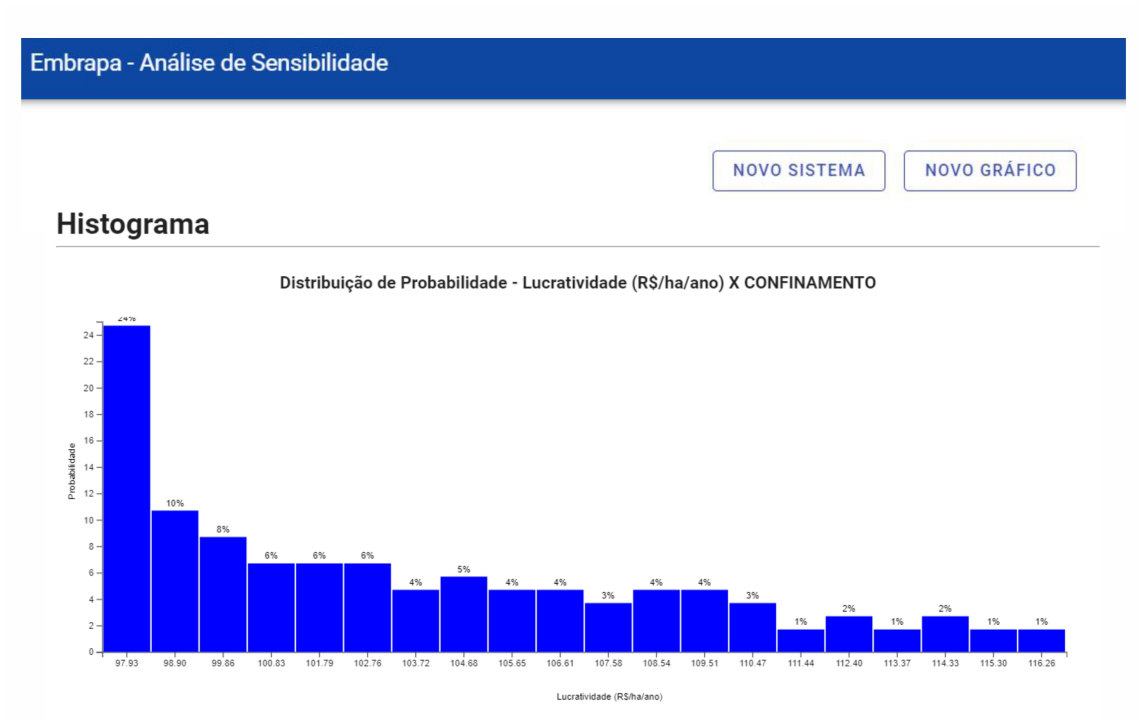

Figura 26: Histograma - Distribuição de Probabilidade.

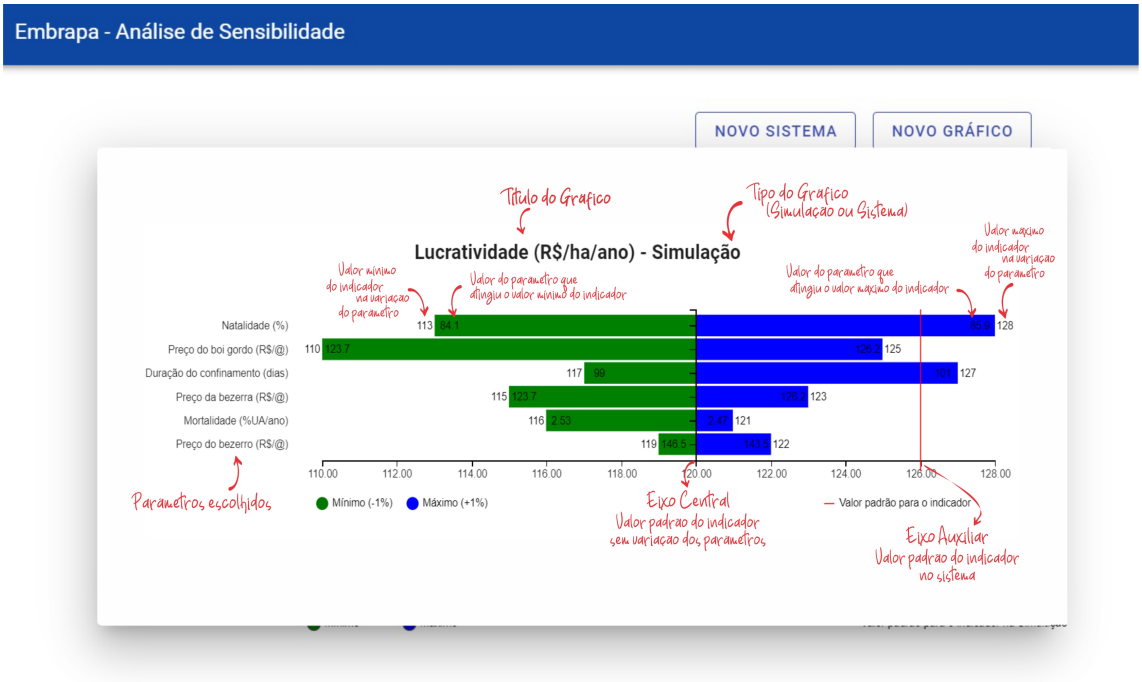

Figura 27: Explicação do gráfico de tornado.

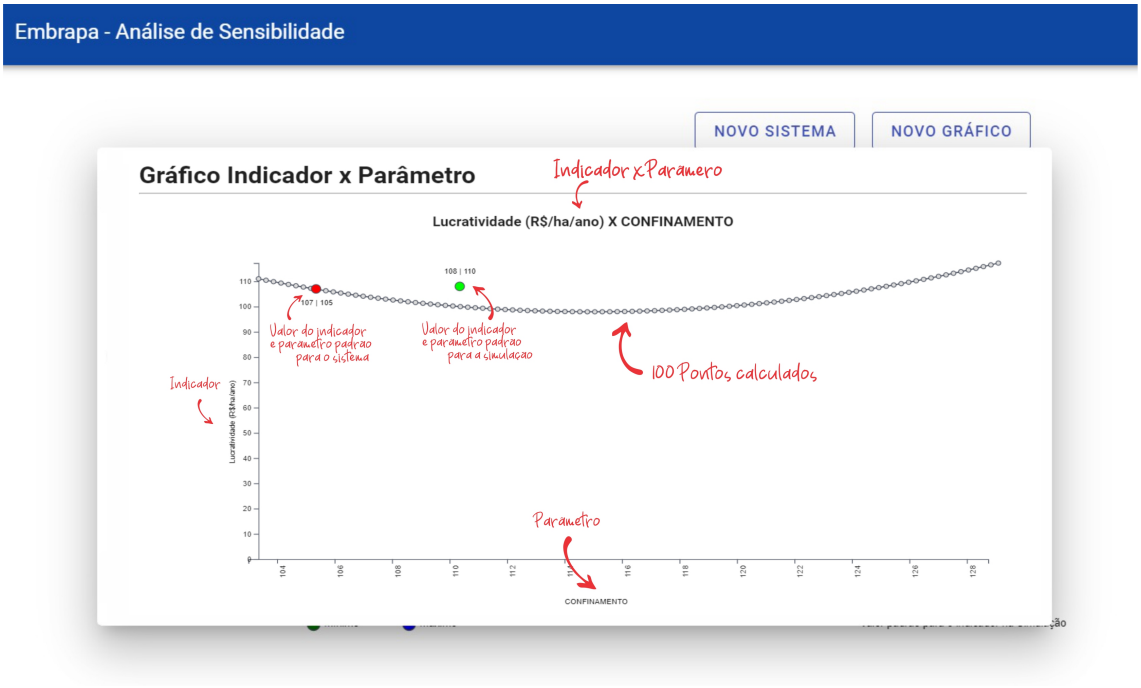

Figura 28: Explicação do gráfico indicadorXparâmetro.

5 Considerações Finais

Este trabalho consistiu em projetar e desenvolver uma aplicação web para análise de sensibilidade em sistemas de produção usados pela P+P, de forma que o usuário seja capaz de identificar o quanto um ou mais parâmetros influenciam no resultado de um indicador. Como resultado, foi desenvolvida a ferramenta descrita nos capítulos 3 e 4, cumprindo o proposto nos objetivos.

5.1 Limitações

Em paralelo ao desenvolvimento deste projeto, a P+P estava passando por uma reformulação, atualização de linguagem e otimização em seu módulo Simulador, e tal atualização não foi finalizada até o presente momento. Como o projeto foi desenvolvido baseado na chamada ao Simulador já baseada na linguagem atual, não foi possível realizar os testes com dados reais na P+P, o que será realizado assim que o Simulador estiver em pleno funcionamento.

Desta forma, um "pseudo-simulador"foi implementado com valores típicos possíveis para cada indicador, de modo que os testes pudessem ser realizados na ferramenta desenvolvida. Este algoritmo foi desenvolvido e os devidos testes realizados, cumprindo assim o proposto para finalização do presente projeto. Os dados já foram preparados corretamente para a chamada ao Simulador real e assim que este estiver implementado e em funcionamento, será integrado à P+P sem a necessidade de acréscimo ou implementação de novas funções ao módulo de análise de sensibilidade.

Espera-se que em breve a ferramenta possa ser totalmente integrada à P+P, de forma que possa ser utilizada por produtores rurais e técnicos para análise de parâmetros relevantes e possibilite auxiliar em suas tomadas de decisões.

5.2 Projetos futuros

Com o módulo de análise de sensibilidade integrado à P+P, é possível ampliá-lo para aprimorar funcionalidades já existentes ou criar novas funcionalidades com o objetivo de otimizar a ferramenta. Podemos destacar duas sugestões para trabalhos futuros.

1. Persistência, no servidor, dos cálculos para análise de sensibilidade de sistemas, a depender da avaliação do tempo de resposta de cálculo X tempo para trânsito dos dados de cálculo já persistidos.

2. Otimização da função responsável pelo *memoize*, de forma a melhorar ainda mais a performance.

3. Implementação do método de Efeitos Elementares, citado no capítulo 3, sobre a análise de triagem realizada em cada relação indicador x parâmetro com o objetivo de estudar as interações lineares entre os parâmetros, de modo a identificá-los por ordem de importância.

Referências

ABIEC. *Perfil da Pecuária no Brasil: Relatório Anual 2020*. [S.l.], 2020. Disponível em: <http://abiec.com.br/publicacoes/beef-report-2020/>. Citado 2 vezes nas páginas 11 e 17.

ARAÚJO, H. S.; SABBAG, O. J.; LIMA, B. T. M.; ANDRIGHETTO, C.; RUIZ, U. dos S. Aspectos econômicos da produção de bovinos de corte. *Pesquisa Agropecuária Tropical*, v. 42, p. 82 – 89, 2012. ISSN 1983-4063. Disponível em: <https://www.scielo.br/pdf/pat/v42n1/12.pdf>. Citado na página 17.

BARCELLOS, J. O. J.; OLIVEIRA, T. E. de; MARQUES, C. S. S. Apontamentos estratégicos sobre a bovinocultura de corte brasileira. 2016. Disponível em: <https://www.researchgate.net/publication/318795687>. Citado na página 23.

BARCELLOS, J. O. J.; SUNé, Y. B. P.; CHRISTOFARI, L. F.; SEMMELMANN, C. E. N.; BRANDãO, F. *A pecuária de corte no Brasil: Uma abordagem sistêmica da produção*. [S.l.], 2013. Disponível em: <http://cdn.fee.tche.br/jornadas/2/E13-03.pdf>. Citado 2 vezes nas páginas 17 e 23.

BASSO, T. *Plataforma +Precoce: Simulador de Sistemas de Produção de Novilho Precoce*. Dissertação (Mestrado) — Universidade Federal de Mato Grosso do Sul, Campo Grande, 2018. Citado na página 18.

BOLFE Édson. *Pecuária integrada e sustentável*. 2017. Disponível em: <https://www.embrapa.br/busca-de-noticias/-/noticia/24523612/artigo—pecuaria-integrada-e-sustentavel>. Acesso em: 18 mai. 2020. Citado 2 vezes nas páginas 17 e 18.

CEZAR, I. M.; QUEIROZ, H. P.; S.THIAGO, L. R. L. de; CASSALES, F. L. G.; COSTA, F. P. Sistemas de producão de gado de corte no brasil: Uma descricão com Ênfase no regime alimentar e no abate. *Embrapa Gado de Corte*, 2005. Disponível em: <https://www.embrapa.br/busca-de-publicacoes/-/publicacao/326307/sistemas-de-producao-de-gado-de-corte-no-brasil-uma-descricao-com-enfase-no-regime-alimentar-e-no-abate>. Citado na página 17.

D3. *D3 Wiki*. 2020. Disponível em: <https://github.com/d3/d3/wiki>. Acesso em: 30 apr. 2020. Citado na página 29.

DEXIE.ORG. *Dexie.js*. 2020. Disponível em: <https://dexie.org/>. Acesso em: 15 jan. 2021. Citado 2 vezes nas páginas 29 e 30.

DOCS, M. W. *IndexedDB*. 2020. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Basic_Concepts_Behind_IndexedDB>. Acesso em: 15 jan. 2021. Citado na página 29.

ENZ, L. de G. *Estudo para simulação a eventos discretos e contínuos para aplicação na pecuária*. Dissertação (Mestrado) — Universidade Federal de Mato Grosso do Sul, Campo Grande, 2019. Citado 2 vezes nas páginas 28 e 29.

FAHLANDER, D. *Dexie.js*. 2020. Disponível em: <https://github.com/dfahlander-/Dexie.js>. Acesso em: 01 fev. 2021. Citado na página 30.

FILHO, K. E. Bovinocultura de corte no brasil. *Revista de Política Agrícola*, v. 32, p. 121 – 129, 2007. Disponível em: <https://seer.sede.embrapa.br/index.php/RPA/article-/view/495/pdf>. Citado 2 vezes nas páginas 23 e 24.

FREY, H.; MOKHTARI, A.; DANISH, T. Evaluation of selected sensitivity analysis methods based upon applications to two food safety process risk models prepared by. 01 2003. Citado na página 20.

GOMES, R. da C.; FEIJÓ, G. L. D.; CHIARI, L. *Evolução e Qualidade da Pecuária Brasileira - Nota Técnica*. [S.l.], 2017. Disponível em: <https://www.embrapa.br-/documents/10180/21470602/EvolucaoeQualidadePecuaria.pdf/64e8985a-5c7c-b83e-ba2d-168ffaa762ad>. Citado na página 24.

HALL, M.; MCNAMEE, J. P. Improving software performance with automatic memoization. p. 254 – 260, 02 1997. Disponível em: <https://www.jhuapl.edu/content-/techdigest/pdf/V18-N02/18-02-Hall.pdf>. Citado na página 27.

IBGE. *Produção da pecuária municipal*. 2019. Disponível em: <https://biblioteca.ibge-.gov.br/visualizacao/periodicos/84/ppm_2019_v47_br_informativo.pdf>. Acesso em: 10 fev. 2021. Citado na página 17.

JÚNIOR, M. A. S. N. *Análise de sensibilidade das variáveis de influência em afundamentos de tensão utilizando projeto e análise de experimentos (DOE)*. Dissertação (Mestrado) — Universidade Federal de Itajubá, Itajubá, 2012. Citado 2 vezes nas páginas 24 e 25.

LANDSKRON, B. F.; KESSLER, J. D. Bovinocultura de corte: Bem-estar animal e produtividade em confinamento. *Caderno Rural*, v. 214, 10 2018. Citado na página 17.

LESSA, M. F. de V. *Metodologias para análise de incertezas paramétricas em conversores de potência*. Tese (Doutorado) — Universidade Federal de Minas Gerais, Belo Horizonte, 2013. Citado 3 vezes nas páginas 20, 24 e 31.

NUNES, D. F. *Procedimento para análise de sensibilidade do programa HDM-4*. Dissertação (Mestrado) — Universidade de São Paulo, São Paulo, 2012. Citado 5 vezes nas páginas 24, 25, 26, 31 e 32.

PEREIRA, C. R. *Aplicações web real-time com Node.js*. São Paulo: Editora Casa do Código, 2013. ISBN 978-85-66250-14-5. Citado na página 27.

PONTES, G. *Progressive Web Apps: Construa aplicações progressivas com react*. São Paulo: Editora Casa do Código, 2018. ISBN 978-85-94188-54-0. Citado na página 28.

RODRIGUES, L. G.; LAGES, S. L.; SILVA, H. L. da. *Código Sanitário de Animais Terrestres – OIE 2014 - Tradução Livre do Capítulo 7 - versão inglês*. [S.l.], 2014. Disponível em: <https://www.gov.br/agricultura/pt-br/assuntos/sustentabilidade/bem-estar-animal/arquivos/capitulo7_9BEABOVINOCORTE.pdf/view>. Citado na página 23.

SILVA, A. S.; GHISI, E. Análise de sensibilidade global dos parâmetros termo físicos de uma edificação residencial de acordo com o método de simulação do rtq-r. *Associação Nacional de Tecnologia do Ambiente Construído*, p. 135 – 148, 2013. ISSN 1678-8621. Disponível em: <https://www.scielo.br/pdf/ac/v13n4/v13n4a10.pdf>. Citado 3 vezes nas páginas 24, 25 e 26.

SILVA, L. R. S. da. *Análise de incertezas e avaliação dos fatores influentes no desempenho de modelos de simulação de bacias hidrográficas*. Tese (Doutorado) — Universidade de Brasília, Brasília, 2010. Citado 3 vezes nas páginas 24, 25 e 26.

SOARES, D. de M. A.; SANTOS, C. L. A. dos; LIMA, P. M. F. de; SANTOS, V. da C.; SOUZA, K. A. de; ABRANTES, R. S. X.; LOIOLA, M. V. do C.; SANTOS, E. L. A. dos. *Noções básicas sobre bovinocultura de corte*. [S.l.], 2016. Disponível em: <https://www.gvaa.com.br/revista/index.php/INTESA/article/view/4571>. Citado na página 23.

SWENSON, L.; LITTLE, J.; MANNING, J.; KROGT, R. Using sensitivity analysis to illustrate and improve schedule robustness. 11 2010. Citado na página 35.

TSUMURA, T.; SUZUKI, I.; IKEUCHI, Y.; MATSUO, H.; NAKASHIMA, H.; NAKASHIMA, Y. Design and evaluation of an auto-memoization processor. 02 2007. Citado na página 27.

VUEJS.COM. *Vuetify JS - Guide*. 2020. Disponível em: <https://vuetifyjs.com/en-/introduction/guide/>. Acesso em: 29 apr. 2020. Citado na página 29.

VUEJS.ORG. *Vue JS - Guide*. 2020. Disponível em: <https://vuejs.org/v2/guide/>. Acesso em: 29 apr. 2020. Citado na página 28.