Licenciatura em Sistemas e Tecnologias da Informação

BluePrism: Interface de Monitorização e Controlo

Projeto Final de Licenciatura

Elaborado por Deyvison Silva de Souza

Aluno nº 20172124

Orientador: Mário José Costa de Macedo

Barcarena

dezembro de 2020

Atlântica Instituto Universitário

Licenciatura em Gestão de Sistemas e Computação

O autor é o único responsável pelas ideias expressas neste relatório

Agradecimentos

O alcançar desta etapa não poderia ser possível sem a colaboração e auxílio por parte de várias pessoas ao longo de todo o percurso da minha formação. Assim sendo, quero aproveitar esta oportunidade para agradecer a todos aqueles que, direta ou indiretamente, contribuíram de alguma forma para o meu sucesso.

Agradeço primeiramente a todos os professores que tanto me ensinaram no decorrer da minha Licenciatura, especialmente ao Mário Macedo, que também contribuiu para a realização deste trabalho.

A todos os meus colegas de curso, pela ajuda e companhia nas inúmeras horas de estudos que tivemos juntos.

À minha mãe, Madalena Araújo, pela educação e sacrifício ao desempenhar o papel difícil de mãe. Obrigado por tudo.

Aos meus irmãos, David Souza e Sângela Souza, pelo apoio ao longo desta trajetória.

Quero agradecer ainda a minha namorada, Ana Lopes, por toda a paciência e compreensão durante estes 3 anos.

Por último, mas não menos importante, agradeço ao meu cão Kog'Maw, por ter sido uma fantástica companhia, o meu muito obrigado.

Resumo

BluePrism: Interface de Monitorização e Controlo

Este projeto tem a intenção de disponibilizar aos utilizadores um *dashboard* intuitivo, ágil e seguro, desenvolvido para comunicar diretamente com a base de dados do Software *Blue Prism*, e disponibilizar dados relevantes para os utilizadores.

A primeira parte deste trabalho aborda as motivações para o desenvolvimento do mesmo, o tema, a justificativa e os objetivos.

Já na segunda parte descreve-se o desenvolvimento do trabalho, apresenta-se a Arquitetura e Protótipo do Sistema de forma direta, as ferramentas e tecnologias utilizadas durante o desenvolvimento são apresentadas, bem como *screenshots* das opções e funcionalidades do *Blue Prism: Dashboard*.

Na parte final apresentam-se as considerações finais sobre o desenvolvimento do *Blue Prism: Dashboard* e sobre o objetivo inicialmente traçado.

Palavras-chave: *Visualização da Informação, Python, Blueprism, Dashboard, Flask.*

Abstract

BluePrism: Monitoring and Control Interface

This project has the intention to provide users an intuitive, fast and secure dashboard, having its 'design' based on good practices in dashboard development. The Blue Prism: Dashboard was developed to communicate directly with the Blue Prism Software database and provide relevant data.

The first part of this work addresses the motivations for its development. The theme, the justification and the objectives.

In the second part the work development is described, the System Architecture and Prototype is presented in a simple and direct way. The tools and technologies used during development are presented, as well the screenshots of the options and features present on Blue Prism: Dashboard.

In the final part, the final considerations about the Blue Prism: Dashboard developed and about the objective initially outlined.

Palavras-chave: *Data visualization, Python, Blueprism, Dashboard, Flask.*

Índice

Índice de figuras

Índice de tabelas

Lista de abreviaturas e siglas

RPA – *Robot Process Automation*

BP – *Blue Prism*

BPP – *Blue Prism Process*

BPR – *Blue Prism Resource*

VM - *Virtual Machine*

CLI - *Command-line Interface*

GUI - *Graphical User Interface*

CRUD - *Create, Read, Update, and Delete*

SSO - *Single Sign-On*

OS – *Operating system*

ODBC - *Open Database Connectivity*

1. Introdução

1.1. Contexto e Motivação

Atualmente, cada vez mais há a necessidade de que as organizações tenham que trabalhar com volume cada vez maior de informações, reunir as mais relevantes e, através da análise dos dados, extrair resultados e disponibilizá-los de forma ágil e transparente, passou a ser um dos principais desafios nos últimos anos.

Todavia, foi-se observado na organização BNP Paribas, que as equipas que trabalham com a ferramenta de automação *Blue Prism*, experienciam dificuldades em disponibilizar informações da própria ferramenta aos utilizadores que fazem parte do grupo de clientes.

O *software* Blue Prism é uma ferramenta de automação Robótica de Processos (*RPA*), permite que as operações sejam ágeis e econômicas, automatizando processos repetitivos, manuais, baseados em regras e melhorando a precisão através do desenvolvimento de uma *Digital Workforce*.

Uma vez que a ferramenta Blue Prism não possui uma interface destinada a visualização informação e estado dos robôs desenvolvidos, para os utilizadores que não fazem parte das equipas de desenvolvimento e suporte, houve a necessidade de disponibilizar estas informações através de um *Dashboard*, assim evitando que os utilizadores abram *tickets* para as equipas de suporte disponibilizarem tal informação.

Posto isto, este projeto tem a finalidade de desenvolver um *dashboard,* na qual disponibiliza informações do *Blue Prism* através de uma página *web*, e tendo como base as boas práticas de desenvolvimento de *dashboards.*

Uma vez que esta solução estiver implementada, o número *tickets* criados pelas operações para as equipas de suporte, irão diminuir drasticamente, pois toda a informação estará em uma única plataforma.

1.2. Definição do Problema

As equipas/clientes que utilizam os BPPs, não possuem uma interface para consultar informações pertinentes a respeito do trabalho desempenhado pelos BPPs. Por limitações do próprio BP, não é possível disponibilizar as informações requeridas pelas equipas de forma ágil e segura através da própria ferramenta.

Quando um BPP inicia a sua atividade até o término da mesma, existem inúmeras informações que são importantes para as equipas. Assim, o principal problema identificado consiste na impossibilidade da ferramenta BP fornecer informações dos BPPs. Em concreto, descreve-se um conjunto de questões associadas a este problema:

1) A equipa não têm visibilidade do estado do BPP;
2) A equipa não têm visibilidade do estado da BPR utilizada pelo BPP;
3) A equipa não têm visibilidade do estado da *queue* de trabalho; e
4) A equipa não possui controlo sobre o BPP.

1.3. Objetivos de Investigação

Considerando o contexto supracitado, foram definidos os seguintes objetivos de investigação (*RG – Research Goals*):

- RG 1 – Efetuar a revisão de literatura de modo a compreender as principais técnicas de visualização efetiva de dados e desenvolvimentos de *Dashboards*.
- RG 2 – Elaborar a especificação de requisitos da aplicação *Blue Prism: Dashboard*;
- RG 3 – Definir a Arquitetura do Sistema;
- RG 4 – Definir os principais Casos de Uso;
- RG 5 – Definir e desenvolver o Modelo de Dados; e
- RG 6 – Desenvolver o protótipo *Blue Prism: Dashboard.*

1.4. Método de Investigação

O método de investigação abordado no desenvolvimento deste projeto foi *Design Science Research* (DSR), com a finalidade de definir uma solução ao problema definido no capítulo anterior.

A metodologia DSR, foi desenvolvida como modelo que fosse consistente com a literatura anterior, criar um modelo padrão para informação científica assim como fabricar um modelo intuitivo mentalmente para pesquisar informação científica relativamente a sistemas de informação. (Peffers, 2006)

Para a adoção apropriada deste método, este trabalho foi desenvolvido seguindo o processo de 6 fases proposto por Ken Peffers e Tuure Tuunanen (Peffers, et al., The Design Science Research Process, 2006):

1. **Identificação do problema e motivação**. Definir as especificidades de problema, e elucidar o mérito de se encontrar a solução. Por conseguinte, encontrar a forma mais eficaz para que o problema seja sanado, procurando englobar todos os desdobramentos, para que seja uma solução integral e eficiente. Ao evidenciar os méritos de solucionar o problema, produzirá tanto um efeito motivacional ao pesquisador, quanto servirá de estímulo para o leitor buscar informações adicionais, e por compreender a perspetiva do autor, dar continuidade a investigação;

2. **Objetivos de uma solução**. Ao definir um problema, deverá ser apontado os objetivos ao solucionar o problema. Essa solução poderá ser quantitativa, se a solução for melhor que a vigente, ou qualitativos, caso seja uma solução inédita para um problema que também ainda não foi abordado. Sendo assim, a solução deverá ser apresentada de forma lógica e racional consoante ao problema, para que dessa forma, os recursos sejam despendidos com eficácia;

3. **Design e desenvolvimento**. para desenvolver uma solução, deverá ser definido em termos gerais, aspetos da construção, modelagem, método ou instalação. Dessa forma, o objetivo é determinar a funcionalidade do artefacto, seguido da sua arquitetura, e por

fim, criá-lo de fato. Entre os recursos utilizados para criar o design e desenvolvimento, deverá ser levado em consideração o conhecimento teórico que possam ser convenientes ao determinar a solução;

4. **Demonstração**. Demonstrar a eficiência do artefacto, no intuito de resolver o problema inicial. Podendo abranger a aplicação em forma de experimentação, simulação, estudo de caso, prova ou outra atividade apropriada;

5. **Avaliação**. Deverá ser aferido a compatibilidade da solução consoante ao problema, onde será comparado os objetivos da solução, de acordo com os resultados reais apreciados em uma demonstração, sustendo-se no conhecimento das mais relevantes métricas e técnicas. Conforme o conteúdo do problema, a avaliação poderá abranger comparativos de funcionalidade, com medidas quantitativas de desempenho, com pesquisas e simulações, de forma que melhor satisfaça a apreciação do cliente. Por fim, ao final desta etapa, será avaliado pelo pesquisador a necessidade de retornar a terceira etapa, caso queira aperfeiçoar algum ponto, ou dar seguimento, deixando melhorias adicionais para projetos futuros.

6. **Comunicação**. É importante a comunicação do problema e a sua relevância, além de demais pontos pertinentes, como o rigor do design, ou mesmo a eficácia aos pesquisadores e aos seus demais interessados. No âmbito académico, os pesquisadores podem usar esse processo na estrutura das suas publicações, já que é comum aos trabalhos de pesquisa empírica a estrutura de definição de problema, revisão de literatura, desenvolvimento de hipóteses, coleta de dados, análise, resultados, discussão e conclusão; e

7. **Demonstração.** Demonstrar a eficiência do artefacto, no intuito de resolver o problema inicial. Podendo abranger a aplicação em forma de experimentação, simulação, estudo de caso, prova ou outra atividade apropriada. Nessa etapa, o recurso imprescindível é o conhecimento eficaz no emprego do artefacto na resolução do problema.

1.5. Estrutura do Documento

Este documento encontra-se segmentado em 6 capítulos, nos quais descrevem todo o trabalho desenvolvido.

No primeiro capítulo, apresenta-se o contexto e motivação, definição do problema, objetivos de investigação e os resultados alcançados;

No segundo capítulo, é efetuada a revisão de literatura e apresentados os principais conceitos e tecnologias abordados neste trabalho;

No terceiro capítulo, descrevem-se os casos de uso, modelo de dados, assim como a arquitetura do sistema implementado;

No quarto capítulo descreve-se o protótipo do sistema de forma geral, o diagrama da estrutura da interface, e finalmente a interface com o utilizador;

No quinto capítulo, por sua vez são apresentados os testes com realizados com os utilizadores e a descrição do cenário; e

Por fim, o sexto capítulo descreve a análise crítica ao trabalho desenvolvido durante todo este projeto e as conclusões extraídas do mesmo.

Esta página em branco foi inserida propositalmente.

2. Revisão da Literatura

2.1. Dados e Informação

No contexto da sociedade atual, a sociedade aprendeu a conviver com a massificação de dados. Com a tecnologia acessível a todo o tipo de utilizador, a produção de informação e seu armazenamento é feito em larga escala, o que deixa ainda mais evidente o porquê. Segundo Manuel Castells (2009), a sociedade encontra-se na "Era Informacional", o último recanto do capitalismo, onde dados são mais valiosos que cédulas de papel-moeda. Consequentemente, com o inestimável número de dados disponível, surge a necessidade de organizá-los, para que possamos perceber o que lhe é de maior relevância, sem perder tempo com uma infinidade de informações insignificantes.

Nesse contexto, uma solução viável e eficiente para a análise de dados, principalmente no contexto de gestão de informações em organizações, é a utilização de interfaces gráficas. Atualmente, devido ao grande volume de informação em que se vive, a análise de dados brutos é impossível ser feita manualmente, pois vai além da capacidade humana de processamento, além de ser pouco atrativa e laboriosa, o que acaba por gerar uma maior tendência ao erro. Ao nível das organizações, onde se preza a eficiência, já que o tempo perdido é sinónimo de custos, a apresentação de dados de forma clara e objetiva pode ser crucial para a qualidade na tomada de decisões, de forma ágil e eficiente. Os custos iniciais para implementar um sistema centralizado pode até ser alto, mas desde que as tarefas são mecanizadas, e as decisões são mais precisas, os custos do serviço torna-se menos dispendioso.

Com os dados apresentados de forma clara, é possível que dados sejam transformados em informação, pois é assim que eles se consolidam, respeitando uma hierarquia assim defendido por Milan Zeleny em 1987, no livro "*Management Support Systems*" (Zeleny, 1987). Na obra, o autor é o primeiro que defende que a gestão do conhecimento é arquitetado no formato de uma pirâmide, onde os dados estariam em sua fundação, e se elevando a informação, conhecimento o ponto mais alto, a sabedoria (DIKW em inglês - *data, information, knowledge* e *wisdom*). Zeleny vai mais adiante ao dizer que no topo da pirâmide estaria a iluminação, e só após passar por todos os níveis, seria possível encontrar a verdade. Abaixo

ilustra-se uma representação da teoria feita pela empresa AGT, que tem seu foco de atuação em IoT.

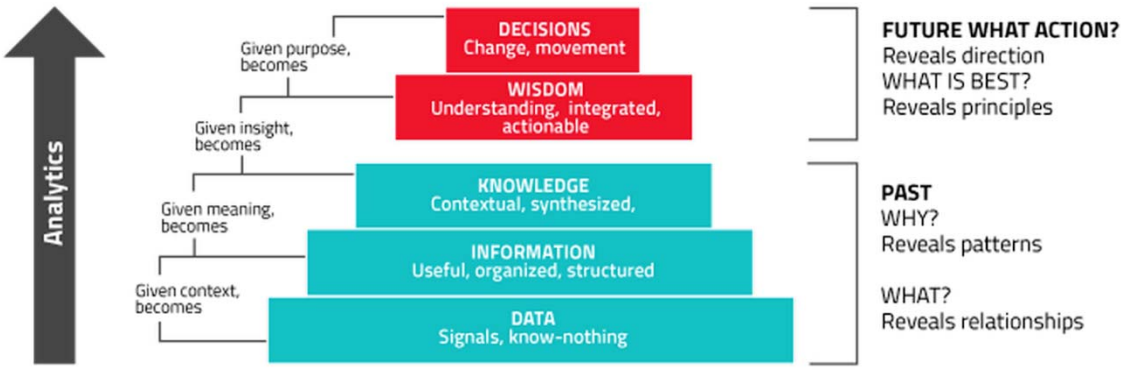

Figura 1 - Modelo de Zeleny pela ótica da empresa AGT, extraído de (Mannion, 2015)

Uma forma muito difundida de interface gráfica é a utilização de *dashboards* de performance, que nada mais é do que um painel visual, cujo principal objetivo é enquadrar informações relevantes, de forma que seja possível acompanhar o desempenho da organização. Em suma, um dashboard de performance permite monitorizar processos e atividades críticas; analisar a origem causadora de problemas, explorando informações relevantes e oportunas de várias perspetivas, em vários níveis de detalhes; gerir pessoas e processos para melhorar a tomada de decisões, otimizar o desempenho e orientar a organização na direção certa.

2.2. Tipos de *dashboards* e aplicações

Há três diferenciações principais de *dashboards*, do qual cada um possui diferentes propósitos, de acordo com o uso que lhe será empregado. O *Dashboard* Operacional, na sua maioria, exibe métricas que deverão ser acompanhadas periodicamente para garantir o bom desempenho das operações. Dessa forma, os seus indicadores deverão ser atualizados a medida que os eventos e atividades ocorrem, o que pode ser em tempo real, a cada minuto, hora ou dia. Dessa forma, é possível perceber rapidamente falhas e erros que possam ter ocorrido e corrigi-los, como, por exemplo, se todos os computadores de um setor estão operacionais. Outra forma de empregar um *dashboard* operacional, é com a função de medir

o desempenho dos funcionários, como exemplo num *call center*, onde é necessário monitorizar quais os atendentes receberam mais chamadas.

O segundo tipo de *dashboard* é o tático, que permite o planeamento, a médio prazo da mobilização de recursos. Esse recurso analisa o desempenho da atividade pelo todo, além de processos e objetivos departamentais. Um exemplo, seria de uma companhia de transporte pode ter um *dashboard* que monitora a média de ocupação das suas linhas, o que permite gerir, se é viável manter aquele trajeto disponível.

Por último, há o *dashboard* estratégico, que acompanha o processo a longo prazo, sobre as metas a serem alcançadas, o que permite empregar táticas mais eficientes. Sendo assim, um *dashboard* estratégico deve conter indicadores de desempenho a longo prazo, assim são usados como parâmetro para aferir *KPIs* vigentes, além de planear a evolução dos negócios. Um exemplo de dados que pode conter esse recurso, é o acompanhamento de meta anual de vendas, com a visibilidade de dados financeiros de abrangência macro.

Devido ao caráter único e complementar de cada *dashboard*, não é incomum que uma empresa possua os três tipos em simultâneo. De acordo com Wayne Eckerson (Eckerson, 2011), autor do livro "*Performance Dashboards - Measuring, Monitoring, and Managing Your Business*", é afirmado que na sua pesquisa para o livro em questão, que das empresas consultadas, 59% utilizavam *dashboards* operacionais, 80% táticos e 64% têm estratégicos. Portanto, se conclui que por volta de dois terços têm, ao mesmo tempo, os três tipos, sendo os táticos os mais populares.

A seguir, também é possível observar a distribuição de tipos de utilizadores, que segundo o mesmo autor, afirma que os executivos utilizam primariamente os *dashboards* estratégicos, embora também possa fazer uso dos *dashboards* táticas para ter uma maior visibilidade do processo. Ao nível médio, estão as táticas, utilizadas por gerentes, para que se consiga acompanhar os processos departamentais, que, entretanto, também podem utilizar as estratégicas e operacionais, dependendo da necessidade de gerir "para cima" ou "para baixo". Por fim, os trabalhadores, utilizam principalmente os *dashboards* operacionais, pois assim podem monitorizar os processos locais, mas também, por vezes, podem utilizar os táticos quando precisam exibir dados departamentais.

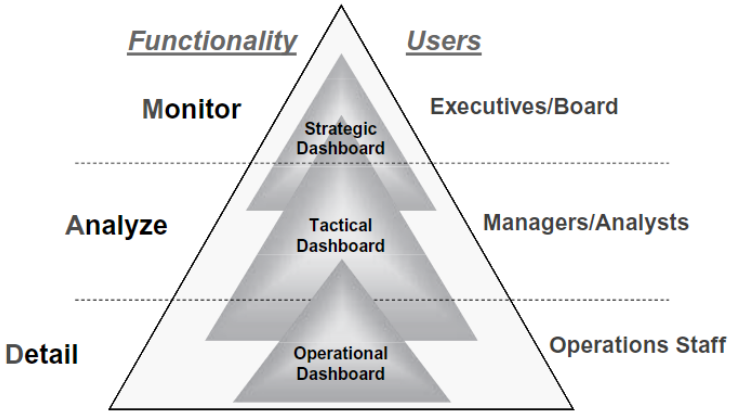

Figura 2 - Esquema de funcionalidade do dashboard em decorrência ao tipo de utilizadores extraído de (ECKERSON, 2011)

2.3. Visualização da Informação

Segundo Trassi (2016), a Visualização da Informação é uma área inovadora, pois se propõe a estudar métodos para obter uma visualização efetiva, para que seja extraído o máximo de informação de um grande volume de dados de forma rápida e clara. A metodologia utilizada é a de metáforas visuais, que busca nos traços humanos, dentre eles, físicos, psicológicos e cognitivos, para aplicar uma técnica de visualização adequada ao que se espera do resultado. A habilidade que a visão tem de assimilar as informações se explica pela sensibilidade do sistema ocular à fatores como contraste, padrão, orientação, entre outros, e a Visualização da Informação desenvolve a sua pesquisa no âmbito de encontrar técnicas para uma interação eficiente entre dados e a capacidade humana de visualizar e processar dados. Advindo desse novo campo de estudo, a Visualização da Informação, são concebidos alguns conceitos, a Visualização Científica, que direciona o seu estudo a dados formados por uma estrutura geométrica orgânica, como os padrões de ondas de calor na atmosfera, que em contraposição a Visualização Analítica, incumbe-se de atribuir significado e raciocínio analítico por meio de interfaces gráficas.

Dito isso, em todos os exemplos de *dashboard* citados, para que sejam eficientes, há algumas constantes, que vai de encontro com os princípios de Visualização da Informação.

Atualmente, a heurística mais difundida no campo do design de interface de utilizador, é a proposta pelo Ph.D. na interação humano-computador, Jakob Nielsen. Os seus estudos propõem 10 heurísticas (Nielsen, 1994a) de usabilidade, e estão sumariamente explicadas na Figura 3.

Figura 3 - Heurísticas de Jakob Nielsen, por Scott Klemmer e Janaki Kumar extraído de (Klemmer, 2017)

A heurística não é uma regra na qual é garantido que a apreciação será perfeita, no entanto, o método direciona o caminho mais eficiente para o resultado satisfatório na experiência do utilizador. Assim sendo, essas regras são categorizadas de acordo com a sua função: compreensão, ação, *feedback*, como é possível observar na Figura 4. Assim, esse recurso é usualmente a base para o desenvolvimento eficiente de interface, que tem como objetivo a interação, experiência e boa navegação. Dada a sua relevância, essas diretrizes também são utilizadas para testar as interfaces depois de prontas, através da "Avaliação Heurística".

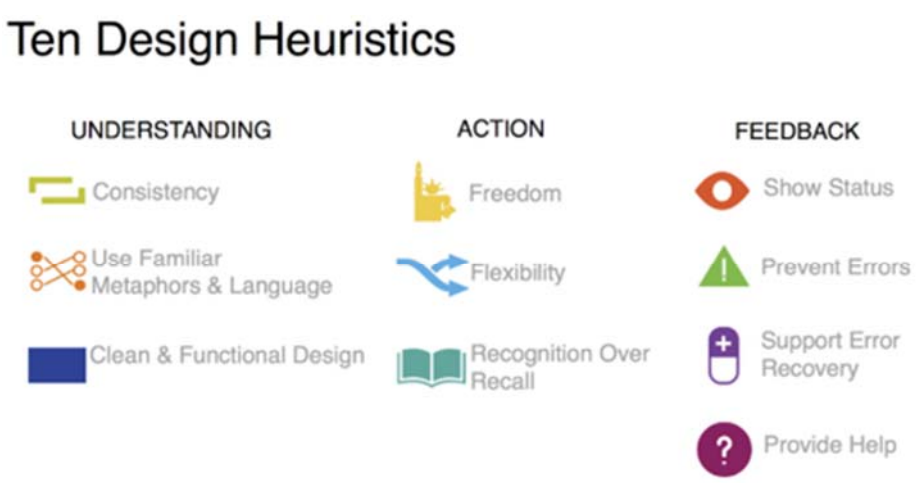

Figura 4 - Classificação das Heurísticas de Jakob Nielsen, por S. Klemmer e J. Kumar extraído de (Klemmer, 2017)

2.4. Research Design

Nos últimos anos, temos testemunhado e participado da luta em que várias instituições acadêmicas tentaram integrar o design com a tecnologia e as ciências comportamentais em apoio ao ensino e à pesquisa de *Human-Computer Interaction* (HCI). Embora tenha havido um grande entusiasmo sobre os benefícios que a integração do design pode trazer, rapidamente percebemos que não existia nenhum modelo de pesquisa acordado para que os designers de interação fizessem contribuições de pesquisa além do desenvolvimento e avaliação de novos métodos de design.

Inicialmente, o termo "design" dentro da de HCI significava engenharia de usabilidade, ou o que Jonas Löwgren (1995) "*o processo de modelar usuários e sistemas e especificar o comportamento do sistema de forma que se encaixasse nas tarefas dos usuários, era eficiente, fácil de usar e fácil de aprender*". Com o tempo, designers começaram a trabalhar com desenvolvedores de software, trazendo habilidades em hierarquia visual, navegação, cor e tipografia que haviam desenvolvido para conteudos impressos. Löwgren rotulou o processo que eles trouxeram para o design de interação como “design criativo” para distingui-lo da abordagem de engenharia. No design de engenharia, os desenvolvedores criaram software para atender a uma especificação e, no design criativo, os designers reformularam

continuamente o problema, questionando constantemente as suposições subjacentes durante o processo de design.

2.5.Critical Design

A área emergente de *design research* ou design construtivo, de acordo com Bardzell (2013), "refere-se à pesquisa em design em que a construção - seja produto, sistema, espaço ou mídia - ocupa o lugar central e se torna o meio-chave na construção do conhecimento". Uma forma de design construtivo é design crítico. *Critical Design* foi o termo atribuído por Anthony Dunne e Fiona Raby no *London Royal College of Art*, usado pela primeira vez em "Contos Hertzianos", como parte da dissertação de Dunne de 1999. Mais tarde, Dunne & Raby desenvolveram o termo em seu livro de 2001 "Design Noir: A Vida Secreta de Objetos Eletrônicos", onde escreveram: "*Em vez de pensar em aparência, facilidade de uso ou identidade corporativa, designers industriais poderiam desenvolver propostas de design que desafiar os valores convencionais*".

No entanto, apesar de seu aparente potencial para *Human-Computer Interaction* (HCI), o design crítico não é muito usado nesta área. A razão pode ser que os pesquisadores de HCI não sabem como fazer, além também aparentar haver uma confusão geral sobre o que é design crítico, e sobre a noção do que é "crítico" sobre o design crítico. O design crítico é adequado para muitas pesquisas contemporâneas de HCI, mas sua aceitação é inesperadamente limitada, sendo sua literatura ainda muito subdesenvolvida para oferecer o suporte prático necessário para sua adoção mais ampla.

2.6.*Design* de *Dashboards*

O *design* é parte crucial na interpretação da informação, e deve servir de aliado, por isso devem ser considerados diferentes aspetos na sua composição. O layout deverá ser intuitivo e organizado, sendo uma tendência entre os autores a afirmar que "menos é mais". A começar pela decoração, que deverá ser evitada, de modo a deixar o foco da atenção do utilizador nos dados importantes. O autor Eckerson vai mais longe nesse conceito, e afirma na sua obra que "nada significa algo" (Eckerson, 2011), o que ele explica que enquanto alguns autores dizem

que ao invés de colocar um semáforo, um único círculo alarmístico ao lado da métrica já seria suficiente, ele aponta que a ausência dum objeto deve ser sinónimo de um desempenho aceitável, ou como ele mesmo cita “nenhuma notícia é uma boa notícia”.

Outro aspeto em relação ao design, é a utilização indevida de efeitos visuais, pois esses recursos, além de gerarem estresse ao utilizador devido às constantes mudanças, acarreta também o desvio da atenção, pois retira o foco dos dados. A complexidade de um *dashboard* não é sinônimo de qualidade, e por esse mesmo motivo, deverão ser evitados enfeites, como bordas esfumaçadas ou marcas d'água, pois isso desvia a atenção, por isso entende-se que quanto mais simples for a visualização, mais facilmente passará a informação ao utilizador (TRASSI, 2016). Também deve ser evitado o excesso de informação, pois um *dashboard* eficiente deve ser objetiva, se restringindo apenas a informações relevantes. Um *dashboard* não é um relatório, e deve conter apenas informações úteis, pois as informações irrelevantes fazem com que o *dashboard* perca a sua razão, já que prejudicar o senso de prioridade.

O que também facilita na monitorização de um *dashboard* é o recurso de alertas, caso algum dado apresente alguma alteração acima do esperado, pois dessa forma, se torna mais eficiente na resolução de problemas. Esse recurso vai de encontro com a primeira regra da heurística apontado por Jakob Nielsen, pois os alertas auxiliam na maior visibilidade do sistema.

Um grande aliado também, podem ser os gráficos e símbolos, que ajudam a perceber em instantes a mensagem, diferentemente do que seria um texto, e por essa razão, devem ser escolhidos com muito cuidado. As identidades desses elementos precisam convergir entre si, por isso não é recomendado utilizar um gráfico muito complexo, com várias informações contidas, e de difícil interpretação. Os gráficos devem ser simples, e se for necessária mais informação, é preferível que seja dividida em diferentes gráficos. Também é vital que faça a escolha criteriosa do modelo do gráfico, pois cada um possui o seu propósito de utilização, ou seja, é necessário entender que um gráfico de pizza, que tem o objetivo de evidenciar partes sobre o todo, não deve ser empregado na função de um gráfico de linha, que tem o objetivo de visualizar a evolução de um evento em decorrência a um espaço de tempo determinado. Por fim, os gráficos devem evitar ao máximo serem repetitivos, pois isso torna a visualização cansativa. Deve-se dar preferência a gráficos diversificados, pois assim permite a visualização de um mesmo todo por diferentes ângulos, pois há mais hipóteses de

se chegar a resultados mais positivos num sistema bem construído, do que numa série cansativa de gráficos de barra.

A matiz, saturação e brilho são aspetos importantes acerca das cores, que deverão ser evitadas as vibrantes, e também o seu excesso, pois isso causa um cansaço visual ao utilizador, além de gerar um aspeto agressivo. Também deve se ter em conta o uso das cores de alarme (verde, amarelo e vermelho), que se forem usadas em informações que não tem o objetivo alarmístico, pode causar confusão ao utilizador. Uma ferramenta que será de uso contínuo, muitas vezes visualizado por longas horas, além de ser clara, deve ser de confortável aos olhos do utilizador. A mesma regra se aplica para o uso de fontes rebuscadas, que deverão ser evitadas, dando preferência a fontes de fácil leitura (Ferreira, 2012).

É importante para um *dashboard*, para que tenha um *design* eficiente, que ele apresente todas as suas finalidades uma única página, pois assim permite uma visualização mais prática, e ágil na interpretação dos dados. As guias são a forma apropriada para diferenciar o conteúdo de acordo a necessidade do utilizador, o que permite alterar o conteúdo funcional de forma mais eficiente, do que pela estrutura hierárquica, que o obrigaria a rolar o conteúdo até encontrar o que lhe é necessário. As guias agrupam conteúdos que deverão se manter juntos, o que torna o aspeto mais organizado, já que não há excesso de informação, além de permitir que o administrador delimite os dados que o utilizador terá acesso, exibindo apenas o que é apropriado a função, o que garante melhor privacidade da informação (Ferreira, 2012).

Até mesmo a maneira de posicionar os elementos em um *dashboard* influencia a forma e a ordem de que o utilizador visualiza o mesmo. Os elementos da parte superior esquerda é o quadrante que recebe maior atenção do utilizador, que em seguida a visão se voltará ao canto superior direito, posteriormente ao canto inferior esquerdo, e por último o canto inferior direito. Seguindo essa tendência humana (Eckerson, 2011), para um maior aproveitamento da atenção do utilizador, os dados mais relevantes no quadrante mais visualizado, e as menos importantes no quadrante que receberá menos atenção. Outra técnica utilizada para uma visualização efetiva, é a agrupar itens correlacionados, pois ao colocá-los afastados faz com que os olhos tenham que se movimentar mais, o que pode ser cansativo, além de facilitar o desvio de atenção. Além disso, é necessário encontrar um equilíbrio no número de elementos, pois uma quantidade limitada de elementos dispersos pode confundir tanto o utilizador

quanto o excesso de informação. Alguns especialistas recomendam que haja algo entre três e sete métricas, mas Eckerson deixa claro que não há uma métrica exata, e que tudo irá variar com o utilizador e a preferência pessoal. O autor afirma que, de modo geral, os trabalhadores que utilizam *dashboards* operacionais preferem um layout mais denso, com uma maior quantidade de texto e gráficos. Em contrapartida, os utilizadores de *dashboards* estratégicos, corresponde aos gerentes e executivos, preferem métricas destacadas com semáforos, e com uma menor quantidade de gráficos associados.

Um recurso primário de interatividade que deverá ser utilizado são os filtros, pois eles permitem refinar a pesquisa ao consultar um conjunto de dados, buscando o pormenor de uma hierarquia, o que permite uma análise detalhada e concisa de um dado agregado a outros, como, por exemplo, a quantidade de irregularidades de um processo versus quantidade de irregularidades deste mesmo processo em decorrência ao operador. Ao marcar e desmarcar os filtros, é permitido ao utilizador explorar os mesmos dados, porém com múltiplas perspetivas.

A navegação estrutural, comummente conhecida como *breadcrumb (*Eckerson, 2011), também é uma ferramenta que auxilia a navegação, pois ela demonstra o caminho percorrido pelo utilizador dentro de um sistema que possui camadas, ou seja, mostra em que subpáginas o utilizador acedeu para chegar ao resultado atual, ou mesmo quais os filtros aplicados. Com isso, torna-se mais fácil retornar a uma pesquisa já feita, e mesmo saber o caminho que o levou até ali, permitindo uma visão macro da organização da informação.

Deste modo, o aspeto citado acima compõe, de forma sintetizada, as principais diretrizes a serem aplicadas no desenvolvimento de um *dashboard*. Em resumo, *dashboards* de performance fornecem as informações certas aos utilizadores certos, no momento certo, para otimizar decisões, aumentar a eficiência e acelerar os resultados. Abaixo são destacados alguns benefícios chave e que reverberam resultados positivos em todos os níveis da organização.

- Comunicar estratégia;
- Refinar a estratégia;
- Aumentar a visibilidade

- Aumentar a coordenação;
- Aumentar a motivação;
- Visão consistente dos negócios;
- Reduzir custos e redundância;
- Capacitar utilizadores; e
- Fornecer informações acionáveis.

2.7. Aplicação Blue Prism

Blue Prism é o nome comercial do Blue Prism Group, uma empresa multinacional de software do Reino Unido que foi pioneira em cria software de automação de processo robótico corporativo (RPA) que fornece uma digital workforce projetada para automatizar atividades operacionais complexas de ponta a ponta (Blue Prism, 2020).

O Software Blue Prism foi criado no Microsoft .NET Framework. Ele automatiza qualquer aplicativo e suporta qualquer plataforma (mainframe, Windows, WPF, Java, web, etc.) apresentada de várias maneiras (terminal emulator, thick client, thin client, web browser, Citrix and web services). O software Blue Prism RPA inclui uma interface de gerenciamento de liberação centralizada e um modelo de distribuição de mudanças de processo, proporcionando altos níveis de visibilidade e controle (Blue Prism, 2020).

Ao longo deste trabalho são abordados termos que fazem parte unicamente do universo Blue Prism, estes mesmos possuem os seguintes significados no ecossistema Blue Prism:

- **Blue Prism** (BP), refere-se à aplicação *Blue Prism,* desenvolvida pela Blue Prism Group*;*
- **Blue Prism Process** (BPP), refere-se aos processos de automação criados através da aplicação *Blue Prism; e*
- **Blue Prism Resource** (BPR), refere-se as máquinas aonde os BPPs são executados em tempo real, obrigatoriamente máquinas com o sistema operativo *Microsoft Windows*.

2.8. Flask microframework

Flask é um microframework desenvolvido em Python e baseado na biblioteca de aplicativos Web WSGI Werkzeug.

É chamado de microframework porque mantem um núcleo simples, mas extensível. Não há uma camada de abstração do banco de dados, validação de formulários, ou qualquer outro componente onde bibliotecas de terceiros existem para prover a funcionalidade, porém Flask suporta extensões capazes de adicionar tais funcionalidades na aplicação final (Flask, 2020).

Com quase nenhuma dependência e layout de projeto necessário, o Flask permite com menos de 3 linhas de código, obtenha um servidor. O Flask não é apenas uma opção conveniente para pequenos projetos, pois também pode ser escalonado para aplicações relativamente maiores e mais complexas.

Flask é uma das Frameworks construídas em Python mais utilizada, como exibe uma pesquisa feita pela companhia JetBrains no ano de 2019.

Como ilustra a Figura 5, o mesmo alcançou o primeiro lugar no ranking, ultrapassando até mesmo o Framework Django, que por muito tempo foi a primeira escolha entre os desenvolvedores de Python, quando o assunto era Desenvolvimento Web.

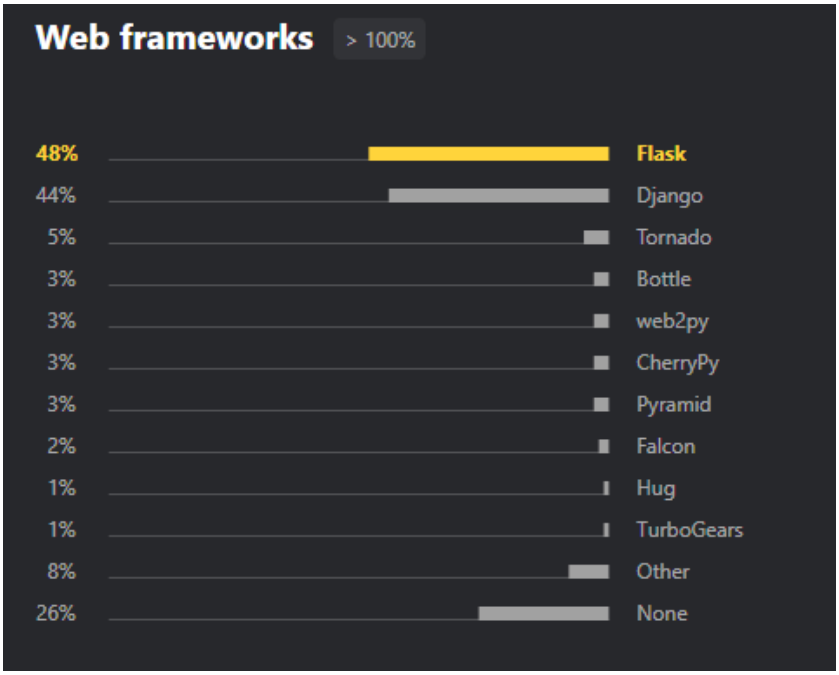

Figura 5 – Web *frameworks survey* extraído de (*JetBrains Survey*, 2019)

2.9. Especificação de Requisitos de Software

A Especificação de Requisitos de Software (SRS) deve descrever com precisão e clareza cada um dos requisitos (funções, performances, design restrições e atributos) do *software* e interfaces externas. Cada requisito deve ser definido de tal forma que sua realização seja capaz de ser objetivamente verificada e validada por um método: por exemplo, inspeção, análise demonstração ou teste (IEEE, 1998).

Encontram-se dois tipos de requisitos, são eles: Requisitos Funcionais (RF) e Requisitos Não-Funcionais (RNF), eles podendo variar quanto aos propósitos e quanto às propriedades que apresentam (Kececi, 2001):

- Requisitos funcionais: tem como objetivo de indicar o que o sistema deve oferecer e como o sistema deve atuar. Estes requisitos dependem do tipo de *software* e do ambiente em que o mesmo esteja implementado;
- Requisitos não funcionais: tem como objetivo de definir restrições ou qualidades para o sistema. As qualidades são particularidades que satisfazem positivamente os utilizadores do sistema e as restrições são limitações que a aplicação possui em uma determinada fase.

O objetivo deste processo é determinar as características do sistema conforme observadas pelo cliente, direcionando o desenvolvimento do software para a direção correta. O Primeiro passo é capturar os requisitos apresentados pelos *stakeholders* (utilizador, clientes, marketing, etc.) compostos por solicitações e desejos. O resultado será um documento com a *Visão do Sistema*, que apresenta as suas características técnicas e funcionais. Posteriormente, estes requisitos são traduzidos em requisitos detalhados, através das ferramentas UML, de modo que possa ser criada uma arquitetura para o sistema (MARTINS, 2007).

3. Especificação de Requisitos

Nesta secção são descritos os requisitos funcionais e não funcionais do projeto Blue Prism: Dashboard. Os mesmos serão apresentados em tabelas, cujas colunas possuem as seguintes informações:

- **Código -** Exibe o código identificado do requisito.
- **Prioridade** - Exibe o nível de prioridade do requisito, podendo contem os valores Alta (*implementação indispensável*), Média (*incremento à aplicação*) e baixa (*implementação optativa*); e
- **Descrição -** Descrição do requisito.

3.1. Requisitos Funcionais

Os requisitos demonstrados a seguir, foram definidos com base as principais necessidades reportadas pelas equipas responsáveis pelos BPPs. As informações foram levantadas por meio de uma ferramenta de Ticketing, na qual todas as equipas utilizam quando há necessidade de reportarem incidentes, duvidas e problemas.

A Tabela 1 exibe os principais requisitos funcionais para o desenvolvimento do Blue Prim: Dashboard.

Tabela 1 - Requisitos funcionais do projeto (Elaborado pelo Autor, 2020)

Requisitos Funcionais		
Código	**Prioridade**	**Descrição do requisito**
RF1	Alta	Permitir que os utilizadores se autentiquem com as credenciais de acesso na página de login.
RF2	Alta	Permitir que os utilizadores alterem a própria palavra passe.
RF3	Alta	Permitir que os utilizadores visualizem e interaja com o Gráfico.
RF4	Alta	Permitir que os utilizadores visualizem o número de vezes que o BPP completou a *run* sem sucesso, com erros.
RF5	Alta	Permitir que os utilizadores corram o BPP.

RF6	Média	Permitir que os utilizadores visualizem os itens disponíveis na *Queue* de trabalho.
RF7	Média	Possibilitar que os Administradores do sistema, possa criar grupos de acesso.
RF8	Média	Permitir que os utilizadores visualizem o estado do BPR.
RF9	Média	Permitir que os utilizadores visualizem o estado do BPP.
RF10	Média	Possibilitar que os Administradores do sistema criem de perfis de acesso
RF11	Baixa	Possibilitar que os Administradores do sistema visualizem os *logs* do sistema.

3.2. Requisitos Não Funcionais

Os requisitos demonstrados a seguir, foram definidos com base em regras de segurança e usabilidade, pré-estabelecidas pela organização onde o projeto foi desenvolvido. Tal como, o requisito **RNF1** que impõe a necessidade de encriptar as *passwords* armazenadas no banco de dados do sistema.

Outros requisitos tiveram como base as limitações técnicas do próprio *Blue Prism*. Tal como, os requisitos **RNF5** e **RNF9**, que por sua vez limitam o *Blue Prism: Dashboard* trabalhar com diferentes tipos de sistemas operativos e banco de dados.

A Tabela 2 exibe os principais requisitos não funcionais para o desenvolvimento do *Blue Prim: Dashboard*.

Tabela 2 - Requisitos não funcionais do projeto (Elaborado pelo Autor, 2020)

Requisitos Não Funcionais		
Código	**Prioridade**	**Descrição do requisito**
RNF1	Alta	A *password* do utilizador deve ser encriptada, antes de ser armazenada na base de dados do sistema.
RNF2	Alta	O sistema deve ser escalável.
RNF3	Alta	Os gráficos devem atualizarem-se em tempo real.
RNF4	Alta	O *Dashboard* deve ser compatível com o *browser* Google Chome.
RNF5	Alta	O sistema deve interagir com o banco de dados SQL Server.
RNF6	Alta	Somente utilizadores autorizados devem ter acesso ao Banco de dados SQL Server.
RNF7	Média	O *Dashboard* deve dispor de uma Interface gráfica responsiva e intuitiva.
RNF8	Média	A aplicação deve estar disponível no idioma Inglês.

RNF9	Média	O sistema deve interagir através da CLI com o sistema operativo *Microsoft Windows*.

4. Modelo de Dados e Arquitetura do Sistema

4.1. Casos de Uso

A Figura 6 ilustra o Diagrama de Casos de Uso, o mesmo apresenta de forma objetiva e introdutória as principais funcionalidades do sistema associadas a interações realizadas pelos atores.

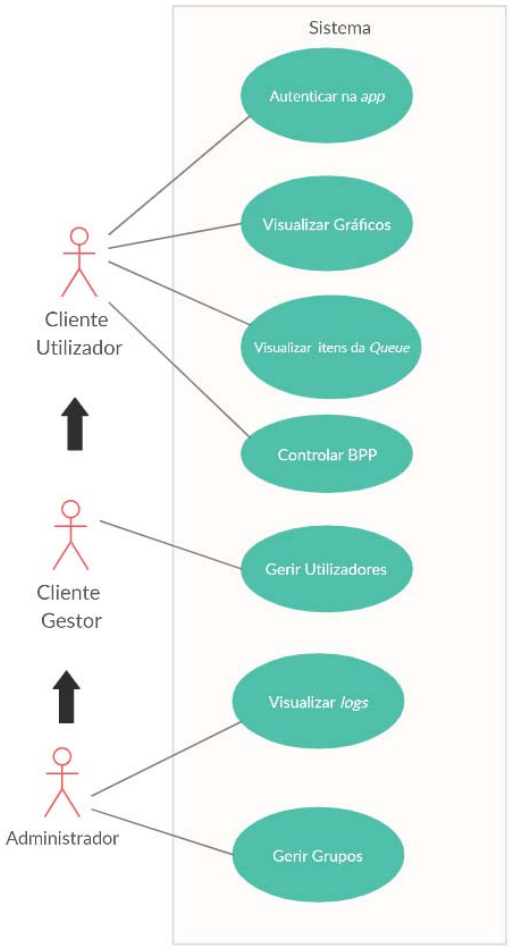

Figura 6 - Diagrama de Casos de Uso (Elaborado pelo Autor, 2020)

A Tabela 3 descreve os casos de uso presentes no sistema.

Tabela 3 - Descrição Casos de Uso (Elaborado pelo Autor, 2020)

UserTable		
Nome	**Ator**	**Descrição**
Autenticar na Página	Cliente Visualizador	*Login* na plataforma com *Email* e palavra-passe
Visualizar Gráficos	Cliente Visualizador	Gráficos e painéis que disponibilizam Informações relativas ao BPP
Visualizar itens da Queue	Cliente Visualizador	Lista de itens que estão na *queue* do BPP
Controlar BPP	Cliente Visualizador	Botão possibilita inicializar e suspender o BPP
Gerir Utilizadores	Cliente Gerente	Formulário para a adição de utilizadores
Visualizar logs	Administrador	Tabela com *logs* de eventos do sistema
Gerir Grupo	Administrador	Formulário para a adição e exclusão de Grupo
Autenticar na Página	Cliente Visualizador	*Login* na plataforma com *Email* e palavra-passe
Visualizar Gráficos	Cliente Visualizador	Gráficos e painéis que disponibilizam Informações relativas ao BPP

A Tabela 4 descreve os Atores dos Casos de uso, bem como seus papéis.

Tabela 4 - Papéis dos Atores (Elaborado pelo Autor, 2020)

Atores e Papeis	
Ator	**Descrição dos papéis**
Cliente Utilizador	Possui somente acesso as funcionalidades de visualização e controlo do *dashboard.*
Cliente Gestor	Possui accesso a todas funcionalidades, excepto as funcionalidades globais do sistema, nomeadamente **Visualizar Logs** e **Criar grupos.**
Administrador	Possui accesso a todas funcionalidades do *dashboard.*

4.2. Modelo de dados

A Figura 7 fornece uma visão geral do banco de dados do *dashboard*, gerada pela aplicação *SQL Server*. O modelo de dados gerado inclui as entidades, as relações entre as tabelas e os atributos.

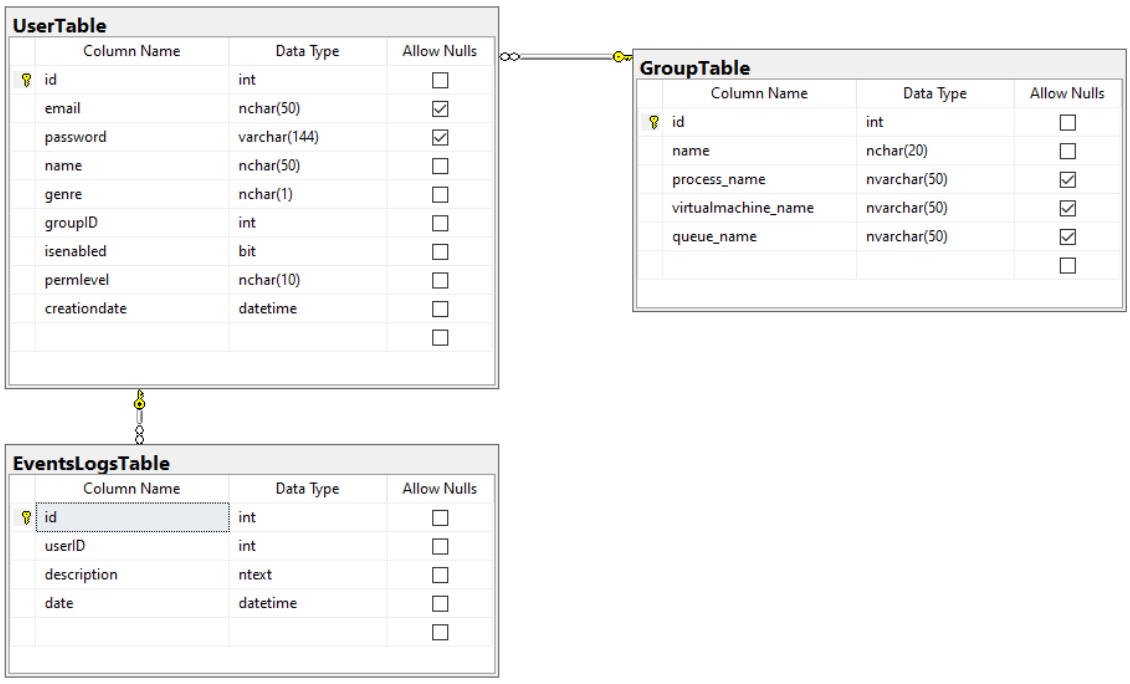

Figura 7 - Diagrama Físico da Base de Dados (Microsoft SQL Server,2020)

As tabelas a seguir descrevem os atributos, tipos de dados, entidades e uma descrição das colunas utilizadas em cada uma das entidades, as seguintes tabelas são:

- Tabela 5 - Tabela UserTable;
- Tabela 6 - Tabela GroupTable; e
- Tabela 7 - Tabela EventsLogsTable.

Tabela 5 - Tabela UserTable (Elaborado pelo Autor, 2020)

UserTable		
Atributo	**Tipo**	**Descrição**
id	int	ID do utilizador (Chave Primária)
email	nchar(50)	Endereço de Email do utilizador
password	varchar(144)	Hash (SHA-512) da Palavra-passe do utilizador
name	nchar(50)	Nome do utilizador
genre	nchar(1)	Género do utilizador representado por uma letra, “M” ou “F”
groupID	int	ID do grupo do utilizador (Chave Estrangeira)
isenabled	bit	Estado do Utilizador representado por um número, “0” ou “1”
permlevel	nchar(10)	Nível de permissão do utilizador
creationdate	datetime	Data de criação do utilizador

Tabela 6 - Tabela GroupTable (Elaborado pelo Autor, 2020)

GroupTable		
Atributo	**Tipo**	**Descrição**
id	int	ID do grupo (Chave Primária)
name	nchar(20)	Nome do grupo
process_name	nvarchar(50)	Nome do processo atribuído ao grupo
virtualmachine_name	nvarchar(50)	Nome da Máquina atribuída ao grupo
queue_name	nvarchar(50)	Nome da *Queue* atribuída ao grupo

Tabela 7 - Tabela EventsLogsTable (Elaborado pelo Autor, 2020)

EventsLogsTable		
Atributo	**Tipo**	**Descrição**
id	int	ID do evento (Chave Primária)
userID	int	ID do utilizador (Chave Estrangeira)
description	ntext	Descrição do evento

date	datetime	Data em que o evento ocorreu

4.3. Arquitetura do Sistema

A Figura 8 fornece uma visão geral da arquitetura do sistema através de uma *tech stack*, observa-se a combinação de linguagens de programação, estruturas, bibliotecas, servidores, software e ferramentas usadas no desenvolvimento do *Blue Prism: Dashboard.*

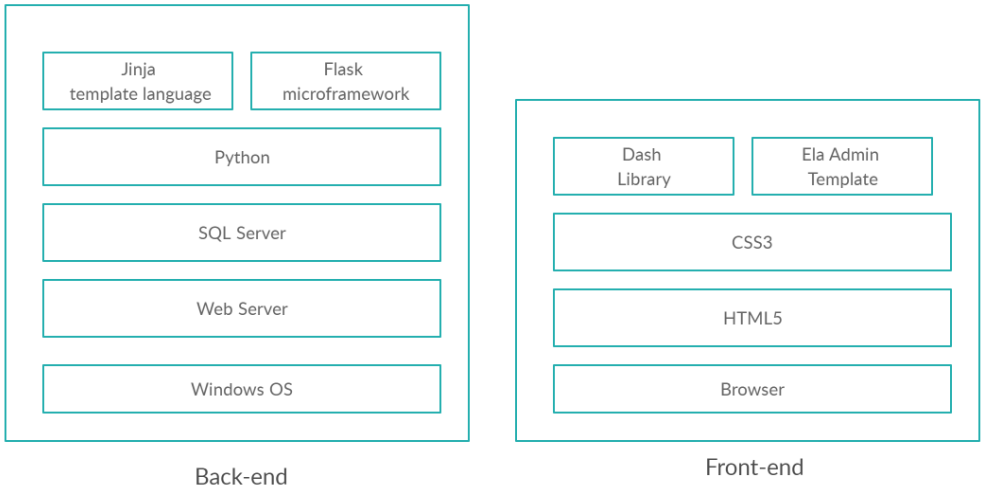

Figura 8 - Arquitetura do Sistema *Blue Prism: Dashboard* (Elaborado pelo Autor, 2020).

Observa-se na pilha de *Back-end*, as seguintes tecnologias implementadas:

Windows OS é o sistema operativo que compõe a arquitetura do sistema, a versão Windows 10 esta implementada no ambiente de testes, enquanto no ambiente de produção há a versão Windows Server.

SQL Server é o Banco de dados que compõe a arquitetura do sistema, visto que a ferramenta *Blue Prism* trabalha exclusivamente com o SQL Server, houve a necessidade do projeto *Blue Prism: Dashboard* também trabalhar com a mesma.

Web server, esta a ser utilizador o *web server* de desenvolvimento da biblioteca Flask.

Pyton, é a linguagem de programação que compõe a arquitetura do sistema, a versão implementada no ambiente de desenvolvimento é a versão 3.6, que por sua vez é a mesma versão que esta implementada nos servidores em produção.

Flask e *Jinja2*, são as principais bibliotecas do projeto, são responsáveis pelas criações de rotas, gestão de pedidos GET/POST, operações CRUD na base de dados e renderizações *templates* HTML.

A biblioteca *Jinja2* trabalha com *Templates*, que são arquivos "*.html*". Por convenção, eles residem no diretório pré-definido "/templates" no projeto Flask. O *Jinja2* possibilita a utilização de variáveis e expressões, que são substituídas por valores quando um *Template* é renderizado; e *tags*, que controlam a lógica do mesmo, possibilitando páginas estáticas se tornarem dinâmicas e interativas aos utilizadores.

Por sua vez, as conexões e interações a base de dados são realizadas pelo *Flask* através do módulo de código aberto *Pyodbc 4.0*, que possua vez simplifica o acesso aos bancos de dados através do padrão *ODBC*.

Observa-se na pilha de *Front-end*, as seguintes tecnologias implementadas:

Browser, o Google Chrome é o navegador que compõe a arquitetura do sistema, o mesmo foi selecionado para o desenvolvimento do projeto visto que em produção o mesmo é o navegador padrão do sistema.

HTML5 e *CSS3,* são as linguagens de marcação e a folha de estilo que compõe a arquitetura do sistema, as versões mais recentes da mesma estão implementadas no projeto.

Ela Admin é um *html template* gratuito desenvolvido em *Bootstrap 4*, disponibilizado pela plataforma *colorlib.com*. No intuito de adequar o *template* ao projeto final, houve a necessidade de customizar as páginas html e folhas de estilo *css*, assim como remover widgets e trechos de código não utilizados, como por exemplo o *Chart.js*, *gmaps*, loadWheather e *FullCalendar*.

Dash Library, é o modulo escrito em *Python* que compõe a arquitetura do sistema, o mesmo é responsável por criar os gráficos em tempo real e disponibilizar aos utilizadores ferramentas de interação.

Outros pacotes e livrarias em *Python* foram utilizados ao longo do projeto, ver anexo A. Python Packages.

5. Protótipo do Sistema

5.1. Descrição Geral

O protótipo *Blue Prism: Dashboard* é uma ferramenta que se comunica com a aplicação BP e disponibiliza informações não unicamente referentes ao BP, mas também referentes ao BPP, BPR e *Queue* de trabalho. Estas informações são disponibilizadas ao utilizador por meio de Gráficos e Painéis informativos, gerados e atualizados em tempo real através da plataforma *open source* Dash, os mesmos são apresentados via *web,* numa interface responsiva desenvolvida em HTML5 e CSS3, que tem como base o *Template* "Ela Admin", adquirido na plataforma colorlib.com.

Quando se refere em interface *web*, não se pode deixar de fora o *JavaScript* e suas *frameworks*, os mesmos foram aplicados em diversas partes do projeto no intuito de facilitar algumas funcionalidades do site. Deste modo o *Blue Prism: Dashboard* é capaz de apresentar e recolher os menus de forma dinâmica e atualizar os Gráficos e Painéis de informações em tempo real, sem a necessidade de carregar a página por completo.

No que diz respeito ao *back-end*, a linguagem Python juntamente com a *microframework* Flask desempenham as funções primordiais como descreve a Arquitetura do Sistema, nomeadamente, criações de rotas, gestão de pedidos *GET/POST*, operações *CRUD* na base de dados, assim como renderizações através de sua biblioteca *Jinja2*, que possibilita a criação de *templates* HTML e com isto a reutilização de código das páginas *web*.

5.2. Estrutura geral dos componentes da interface

Na Figura 9 é possível visualizar o Diagrama da estrutura da interface

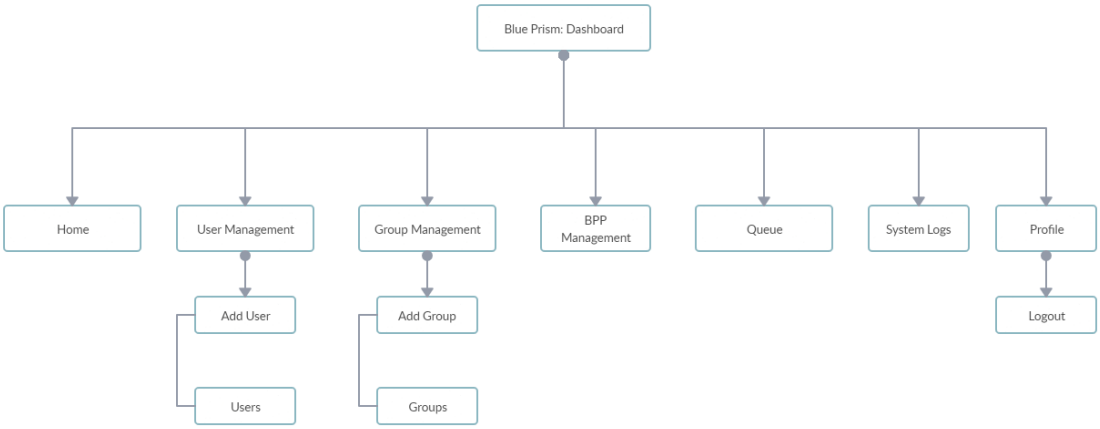

Figura 9 - Diagrama da estrutura da interface (Elaborado pelo Autor, 2020)

5.3. Interface com o Utilizador

5.3.1. Página de *Login*

Para iniciar sessão no Blue Prism: Dashboard, os utilizadores têm de introduzir credenciais válidas, ou seja, endereço de Email previamente cadastrados no sistema por um **Cliente Gestor** ou **Administrador**, e uma Palava-passe valida.

Na Figura 10 é possível visualizar a página de autenticação da aplicação Blue Prism: Dashboard.

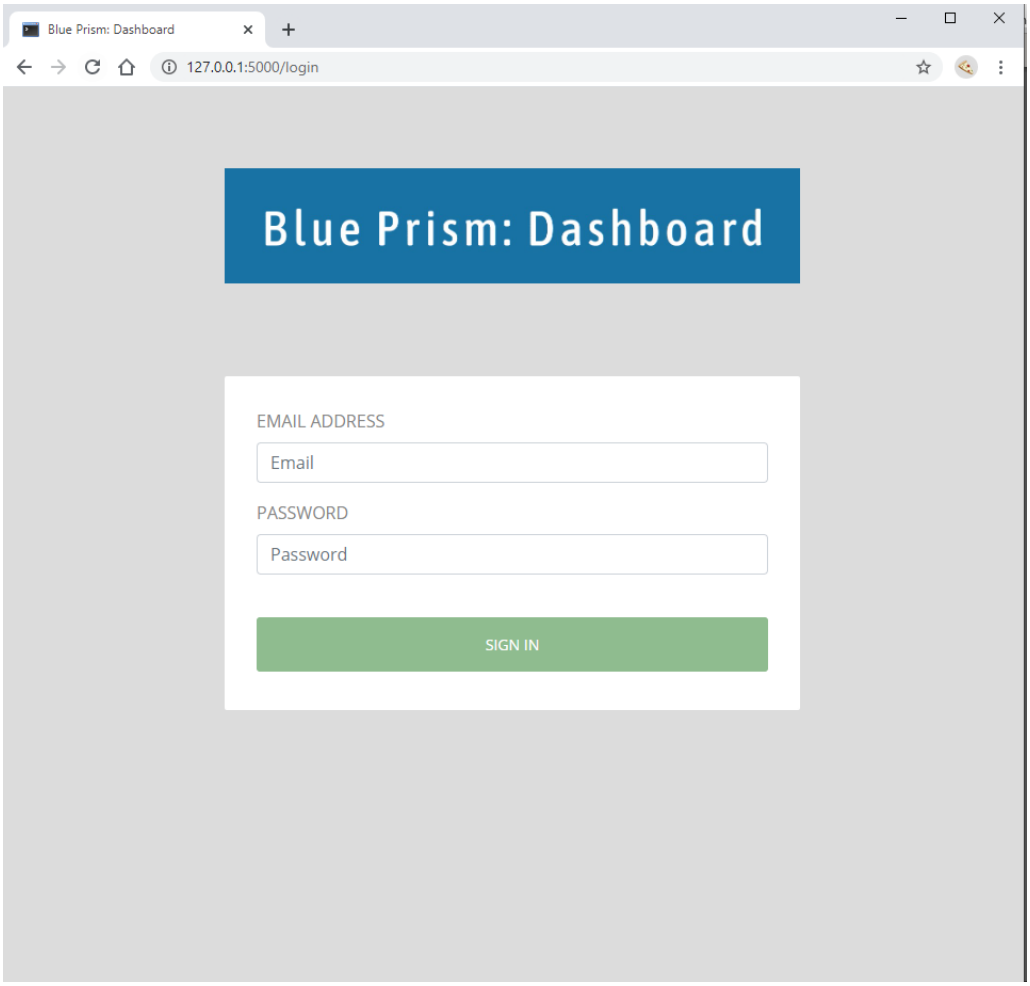

Figura 10 - Página de Autenticação (*Blue Prism: Dashboard*, 2020)

A Página de autenticação irá apresentar mensagens de alertas, caso ocorra algum erro por parte do utilizador na fase de autenticação, tais como:

Formato de endereço de *e-mail* incorreto

Uma vez que o utilizador escreva incorretamente o endereço de *e-mail*, a página de autenticação por meio da linguagem *JavaScript,* identifica o erro e apresenta um *pop-up* com indicações ao utilizador de como o resolver.

Na figura 11, podemos observar um dos erros mais comuns que possam ocorrer por parte do utilizador na fase de autenticação com o enderenço de email, a ausência do "@".

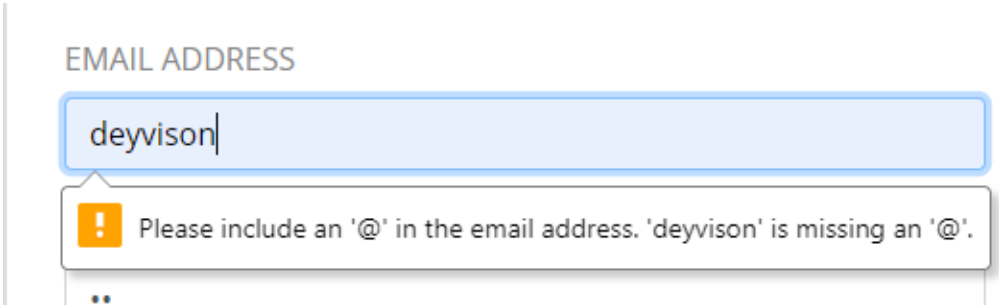

Figura 11 – *Pop-up* formato do endereço de *e-mail* incorreto (*Blue Prism: Dashboard*, 2020)

Endereço de email não registado

Uma vez que o utilizador tente efetuar a autenticação com um *e-mail* não previamente registado por um **Cliente Gestor** ou **Administrador**, o formulário de *login* envia o *e-mail* do utilizador através de um *Post Request,* e após ser validado pelo *backend*, é apresenta a mensagem de erro ilustrada na Figura 12.

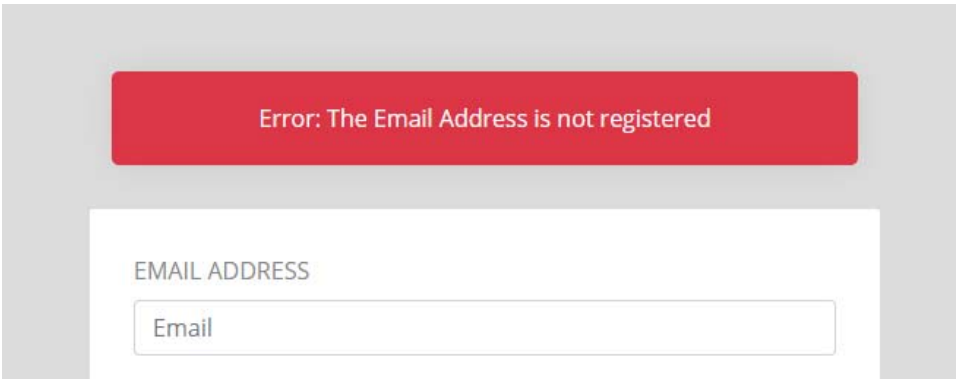

Figura 12 - *Pop-up* endereço de *e-mail* não registado (*Blue Prism: Dashboard*, 2020)

Palavra-passe invalida

Por sua vez, uma vez que que o utilizador tente efetuar a autenticação com uma palavra-passe invalida, o formulário de *login* envia o *e-mail* do utilizador através de um *Post*

Request, e após ser validado pelo *backend*, é apresenta a mensagem de erro ilustrada na Figura 13.

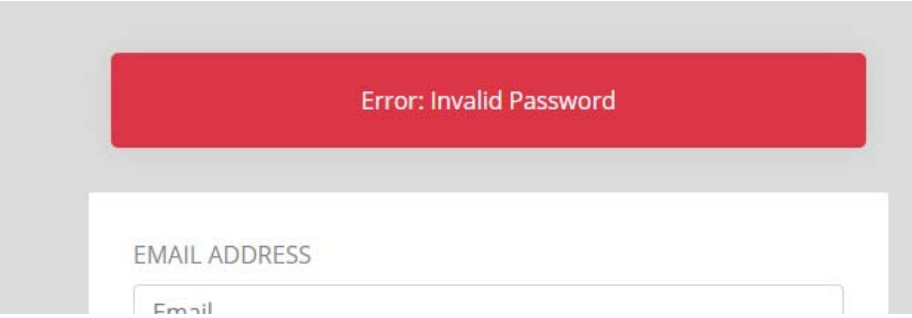

Figura 13 – *Pop-up* palavra-passe invalida (*Blue Prism: Dashboard*, 2020)

5.3.2. Reset your password

Sempre que o utilizador se autentica com uma palavra-passe temporária, ou seja uma palavra-passe que foi gerada aleatoriamente e definida automaticamente ao perfil, na data em que o mesmo foi criado, o utilizador é redirecionado para a página *Reset your Password*, para que o mesmo possa redefinir a palavra-passe para uma mais segura e confortável para si.

Contudo, uma vez que o utilizador faça login com a palavra-passe já redefinida, o mesmo é redirecionado para Página Inicial do sistema. As palavras-passes são encriptada com a função *sha512 hash*, através da livraria *werkzeug.security*.

A Figura 14 ilustra a página *Reset your Ppassword* da aplicação Blue Prism: Dashboard.

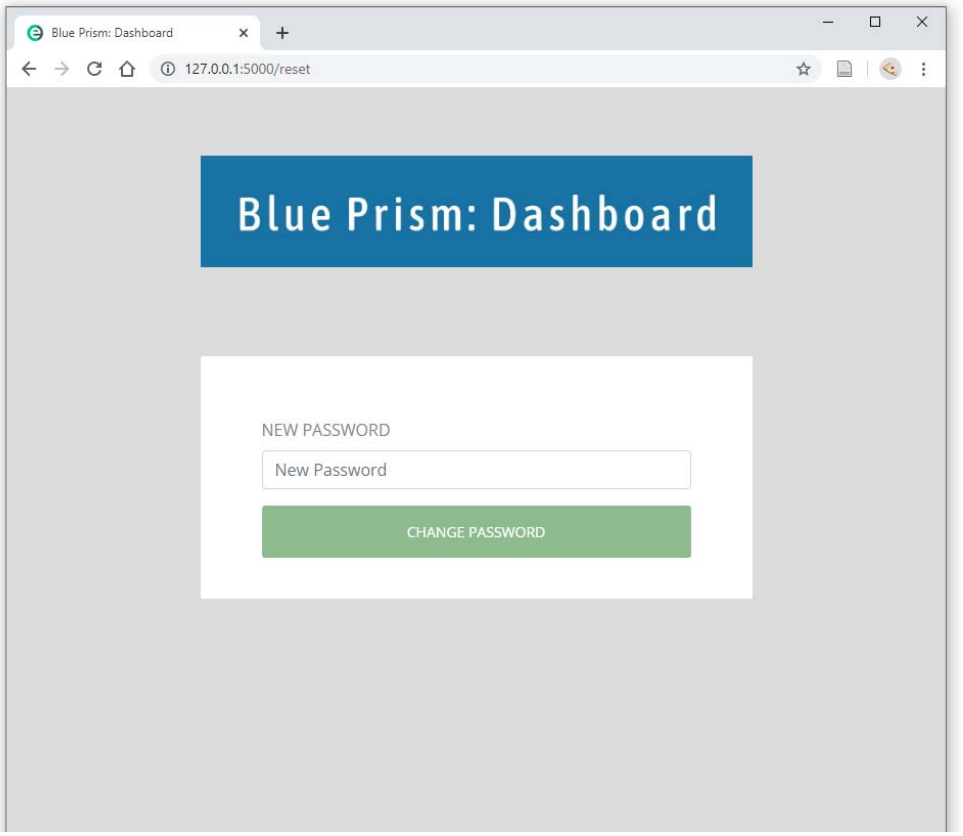

Figura 14 – Página *Reset your Password* (*Blue Prism: Dashboard*, 2020)

Uma vez que o utilizador faça o *login* pela primeira vez no sistema, a Figura 14 será apresentada, para que o utilizador defina uma palavra-passe de sua escolha, que irá ser utilizada nas próximas vezes que o mesmo se autenticar.

4.3.3. Página Inicial

A Figura 15 ilustra a Página inicial do sistema, a mesma contém Gráficos e Painéis informativos que disponibilizam ao utilizador o estado do BPP, BPR e a *Queue* de trabalho, assim como algumas funcionalidades no *menu* do lado esquerdo, e as demais na foto de perfil do utilizador.

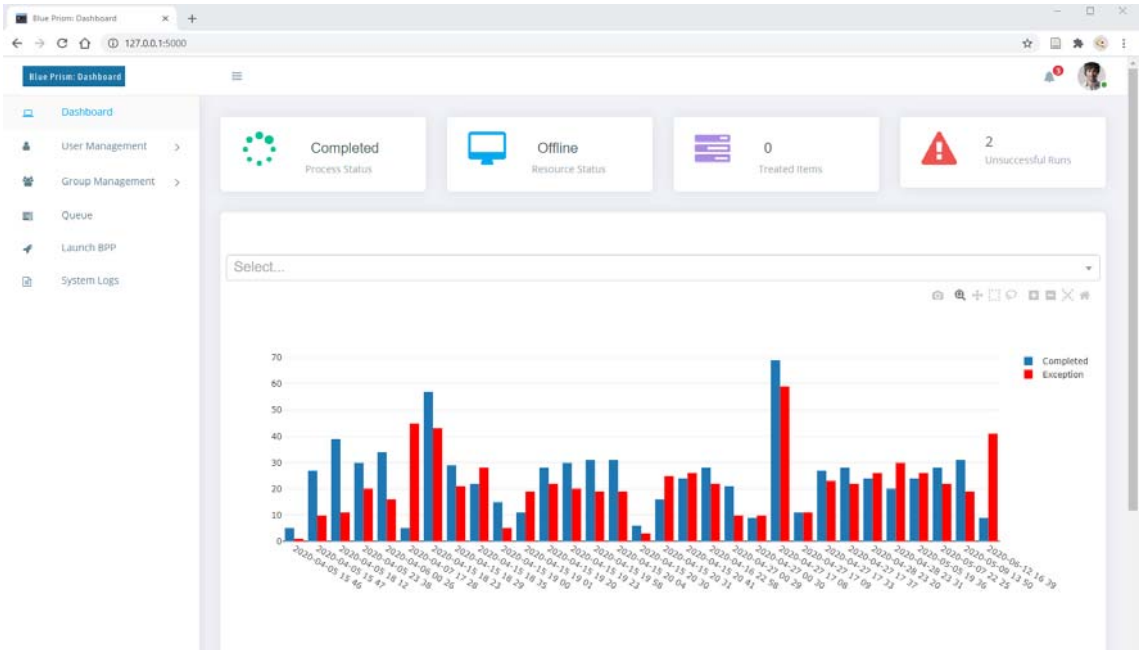

Figura 15 - Página Inicial (*Blue Prism: Dashboard*, 2020)

1) Menu principal
2) Painéis informativos
3) Opções do perfil

Painéis Informativos

A Figura 16 ilustra os painéis interativos, os mesmos têm o objetivo de disponibilizar ao utilizador uma série de informações, de forma objetiva e em tempo real. As informações presentes nos painéis foram estipuladas com base nos problemas definidos na Introdução deste trabalho, na secção **Definição do Problema**. A seguir, é possível visualizar os diferentes *status* que cada painel exibe ao utilizador. De igual modo, a tecnologia adotada segue as boas práticas identificadas na revisão da literatura ao nível da visualização da informação e construção de *dashboards*.

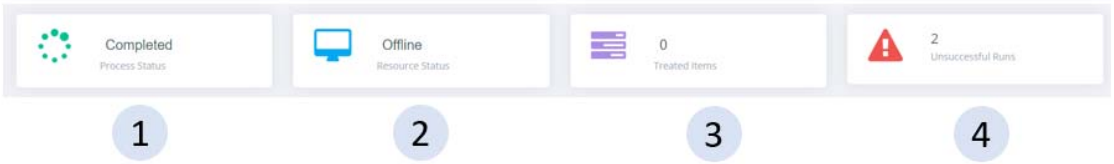

Figura 16 – Painéis Informativos (*Blue Prism: Dashboard*, 2020)

A Tabela 8 descreve todos os estados possíveis, para cada um dos painéis disponíveis no sistema, ilustrado anteriormente na Figura 16.

Tabela 8 - Estados dos Painéis Informativos (Elaborado pelo Autor, 2020)

Painéis Informativos			
Nº	**Tipo**	**Estado**	**Descrição**
1	Process Status	*Completed*	O processo (BPP) completou a última *run* com sucesso.
		Terminated	O processo (BPP) completou a última *run* com erros.
		Running	O processo (BPP) esta correr.
		Debugging	O processo (BPP) esta a correr em *debug mode*.
2	Resource Status	*Online*	A Máquina (BPR) esta online.
		Offline	O processo (BPR) esta offline.
3	Treated Items	Contador	Número total de itens tratados no dia atual.
4	Unsuccessful Runs	Contador	Número de vezes que o BPP completou a *run* sem sucesso, com erros.

Gráfico Interativo

O Gráfico interativo foi desenvolvido através da biblioteca Dash em Python, o mesmo é atualizado em tempo real, responsivo e possui ferramentas que auxiliam o utilizador a manipular o gráfico que esta a ser visualizado.

A Figura 17 ilustra o Gráfico da página inicial, e as ferramentas de manipulação do mesmo.

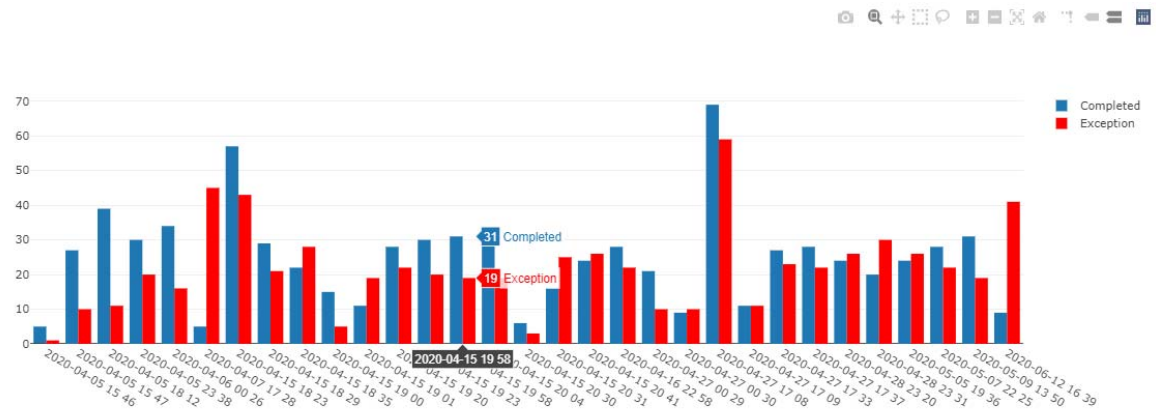

Figura 17 - Gráfico Interativo (*Blue Prism: Dashboard*, 2020)

Barra de ferramentas

A Figura 18 ilustra as ferramentas disponíveis no Gráfico, as mesmas podem ser removidas individualmente da interface, podem também ter o tamanho e a descrição alterada.

Figura 18 – Barra de ferramentas do Gráfico (*Blue Prism: Dashboard*, 2020)

A Tabela 9 descreve a funcionalidade de cada item disponível na barra de ferramentas.

Tabela 9 – Opções da barra de ferramentas (Elaborado pelo Autor, 2020)

Barra de ferramentas		
Nº	**Nome**	**Descrição**
1	*Download Plot as a png*	Descarrega uma imagem do Gráfico em formato PNG.

2	*Zoom*	Zoom em uma determinada parte do Gráfico.
3	*Pan*	Move o Gráfico.
4	*Box Select*	Seleciona uma parte do Gráfico em formato de caixa.
5	*Lasso Select*	Seleciona uma parte do Gráfico em formato "laço".
6	*Zoom in*	Mais *Zoom.*
7	*Zoom Out*	Menos *Zoom.*
8	*Auto Scale*	*Reset* no *Zoom.*
9	*Reset Axes*	*Reset* nas coordenadas do gráfico.

Em anexo há um exemplo de como é desenvolvido e configurado um gráfico com a biblioteca *Dash*, ver anexo B. *Create_Dash_Queue Class*.

4.3.4. User Management

O *User Manager* possui duas opções, nas quais serão definidas a seguir:

1) Add User

A opção *Add User* possui níveis de restrições diferentes de acordo com o tipo de permissão do utilizador.

Utilizadores com permissões de **Administrador**, conseguem criar perfis e atribuí-los a grupos, estes perfis possuem permissões do tipo **Gerente**.

Por sua vez, os utilizadores do tipo **Gerente**, podem criar perfis, mas os mesmos são atribuídos automaticamente ao atual grupo do criado, estes perfis possuem permissões do tipo **Visualizador**.

Por fim, os utilizadores do tipo **Visualizador**, não possuem permissões para adicionar novos perfis.

A Figura 19 ilustra a página *Add User*, que esta localizada dentro da opção *User Management*.

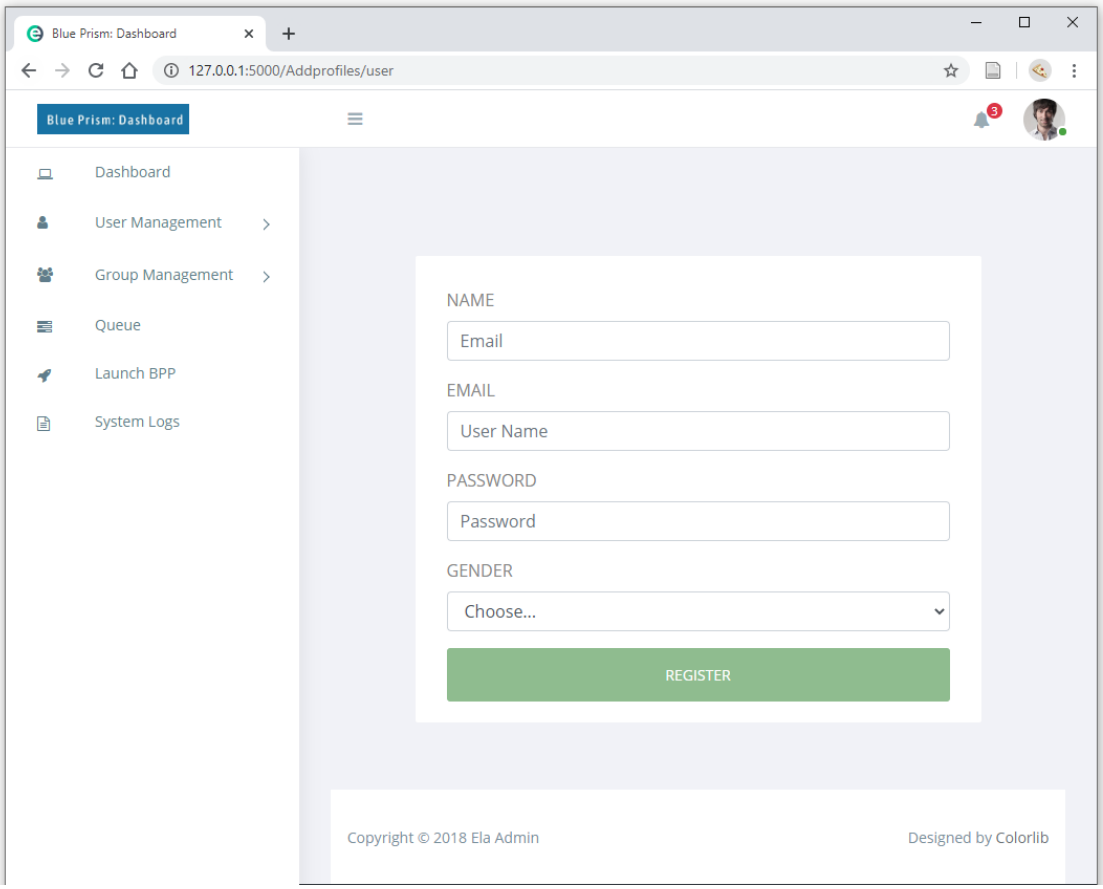

Figura 19 – Opção *User Management*, página *Add User* (*Blue Prism: Dashboard*, 2020)

2) Users

A opção *Users* disponibiliza ao utilizador uma lista dos utilizadores que fazem parte do mesmo grupo.

A Figura 20 ilustra a página *User*, que esta localizada dentro da opção *User Management*, a mesma esta disponível a todos os utilizadores do sistema.

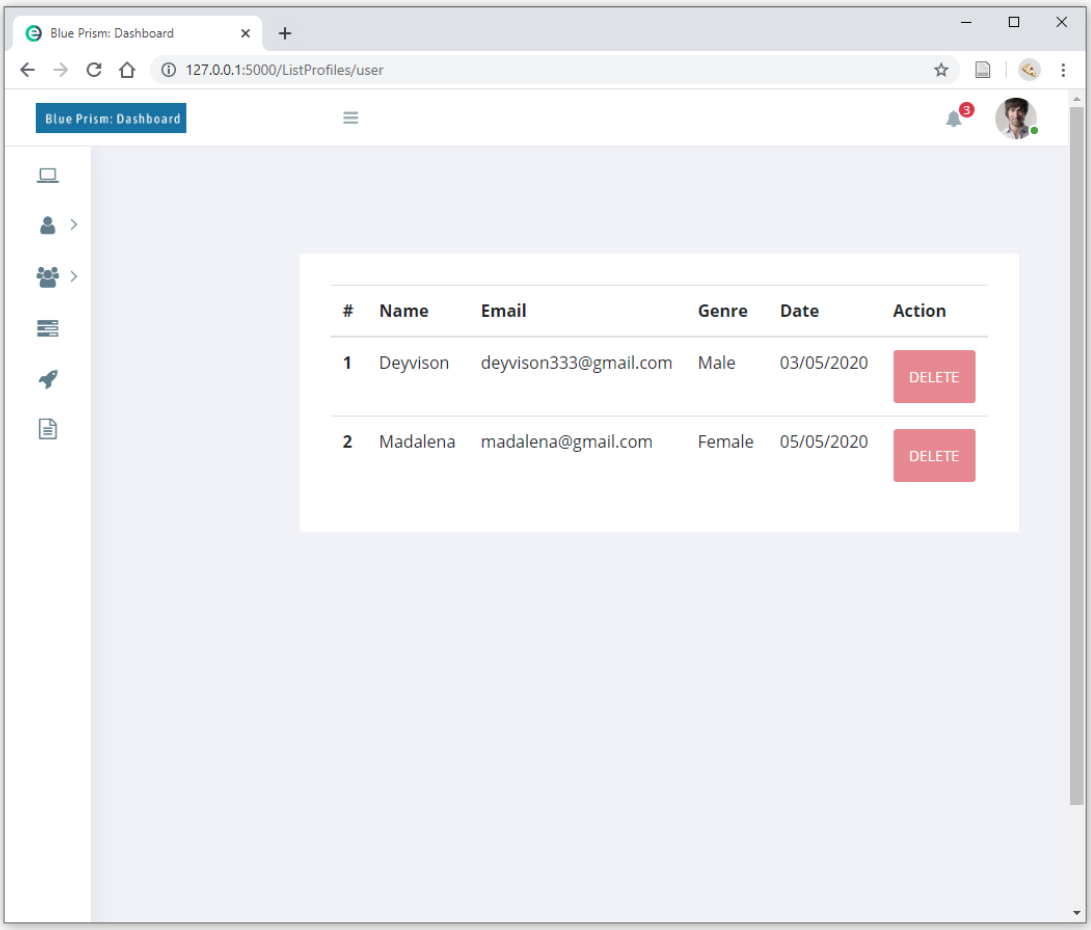

Figura 20 - Opção *User Management*, página *Users* (*Blue Prism: Dashboard*, 2020)

4.3.5. Group Management

O *Group Manager* possui duas opções, nas quais serão definidas a seguir:

1) Add Groups

A opção *Add Group* possui níveis de restrições diferentes de acordo com o tipo de permissão do utilizador.

Utilizadores com permissões de **Administrador**, são os únicos que possuem permissões para criar novos grupos.

A Figura 21 ilustra a página *Add Group*, que esta localizada dentro da opção *Groups Management*.

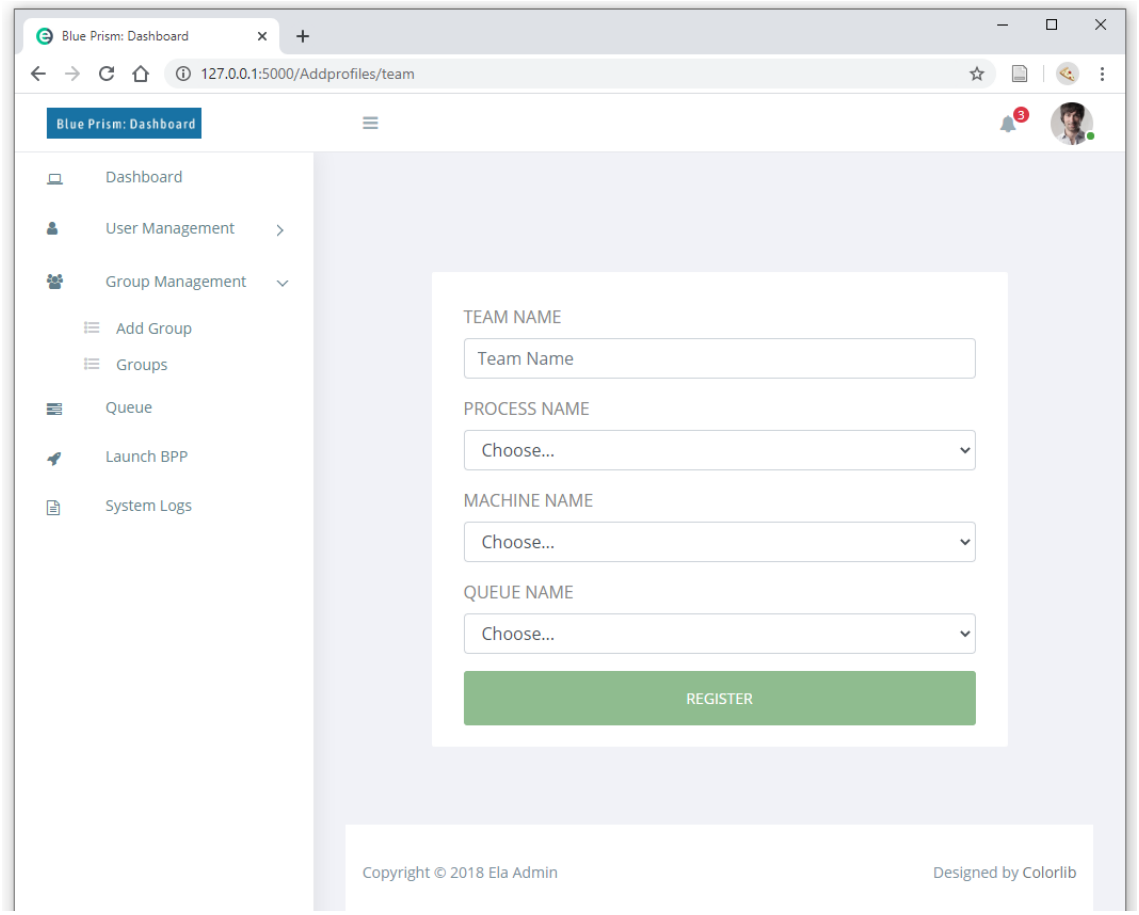

Figura 21 - Opção *Group Management*, página *Add Groups* (*Blue Prism: Dashboard*, 2020)

2) Groups

A opção *Groups* disponibiliza ao utilizador uma lista dos utilizadores que fazem parte do mesmo grupo.

A Figura 22 ilustra a página *Groups*, que esta localizada dentro da opção *User Management*, a mesma esta disponível a todos os utilizadores do sistema.

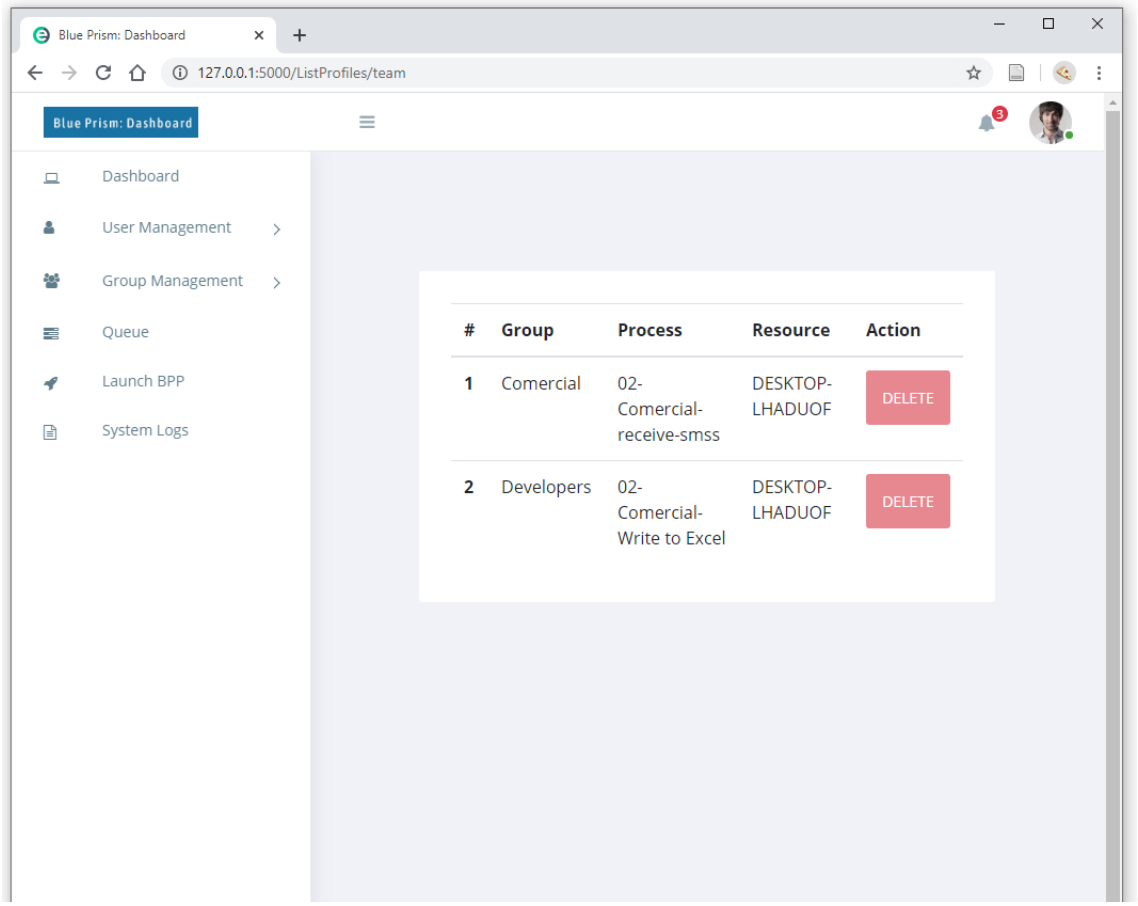

Figura 22 - Opção *User Management*, página *Groups* (*Blue Prism: Dashboard*, 2020)

4.3.6. System Logs

A Página *System Logs*, disponibiliza aos administradores do sistema, uma tabela com histórico de cada ações desempenhadas por todos os utilizadores no sistema, tais como Horário de autenticação e *logout*, perfis e grupos criados e etc.

A Figura 23 ilustra a página *System Logs*, esta página esta disponível exclusivamente aos utilizadores com permissões do tipo **Administrador**.

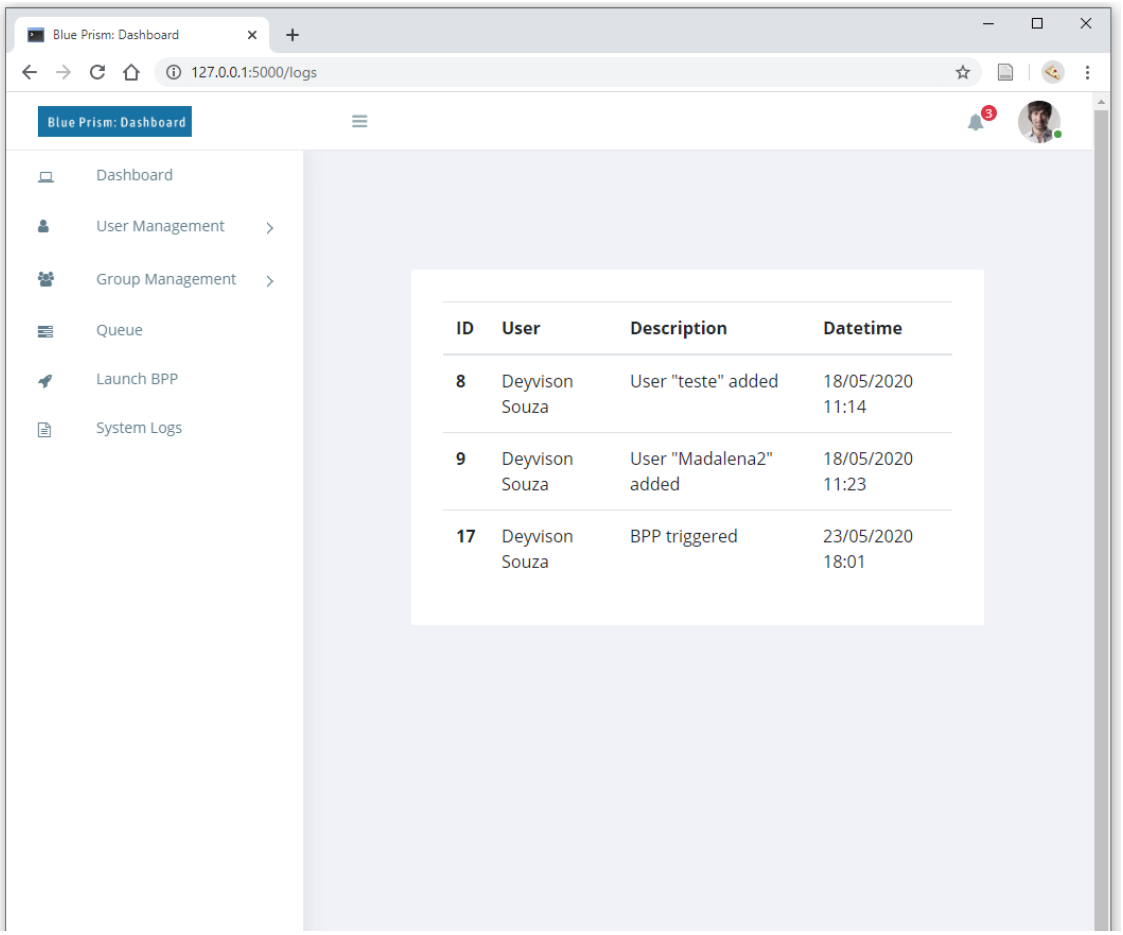

Figura 23 - Página *System logs* (*Blue Prism: Dashboard*, 2020)

4.3.7. *Launch* BPP

A página *Launch BPP* disponibiliza uma interface de comunicação entre o utilizador e o BPP. A Interface apresenta um botão *"Stop"* caso o BPP esteja a correr, ou *"Start"* caso o BPP esteja parado.

O Controlo oferecido ao utilizador através do botão *Trigger*, é feito através de pedidos encaminhados ao Blue Prism via CLI.

A Figura 24 ilustra a página *Trigger BPP*, esta página esta disponível a todos os utilizadores do sistema.

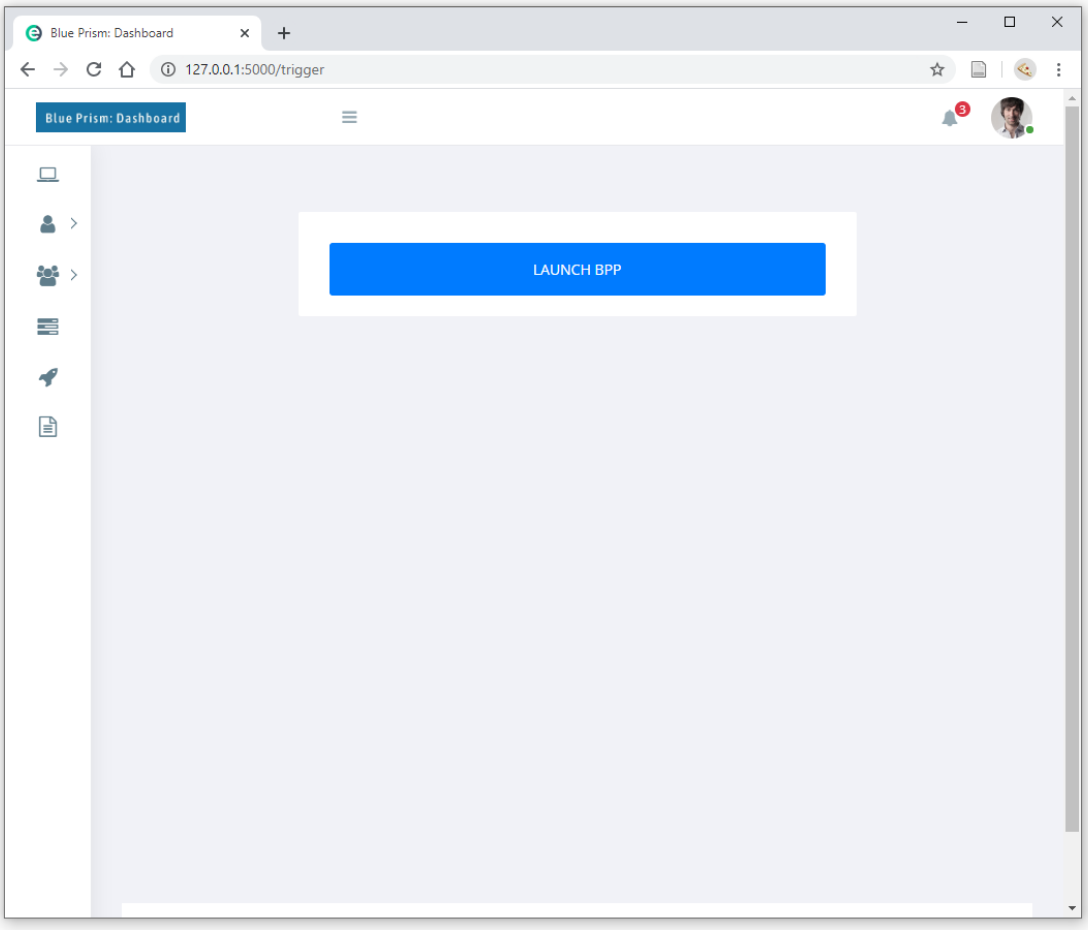

Figura 24 - Página *Trigger BPP* (*Blue Prism: Dashboard*, 2020)

5.4. Cenário de Testes

Para garantir que o *Blue Prism: Dashboard* satisfaz critérios como performance e tempo de resposta, foi necessário efetuar testes num ambiente semelhante ao de produção. Os testes foram realizados no intuito de verificar o tempo de resposta da aplicação, determinando assim a sua escalabilidade e confiança.

A Tabela 10 exibe os principais programas e suas versões utilizados na fase de desenvolvimento e teste do *Blue Prim: Dashboard*, no intuito de comparar as versões utilizadas no ambiente de testes e as de produção, tal como as justificativas para as escolhas feitas.

Tabela 10 – Aplicações de testes (Elaborado pelo Autor, 2020)

		Ambiente de Testes e Produção	
Aplicação	**Versão de Teste**	**Versão de Produção**	**Justificativa**
Microsoft Windows	Windows 10	Windows Server 2016	O computador utilizado para o desenvolvimento e testes não possui configurações minimas para correr o Windows Server 2016. Por este motivo, tivemos que optar por uma versão mais recente.
Blue Prism	Blueprism 6.7	Blue prism 5.1	O Site oficial do Blue Prism não disponibiliza de forma gratuita as versões anteriores, somete a versão Blueprism 6.7 *trial* é oferecida pela plataforma. Por este motivo, tivemos que optar por uma versão mais recente.
SQL Server	SQL Server 2019	SQL Server 2016	A versão mais recente do Blue Prism (6.7) tem como requesitos obrigatorios o SQL Server 2019. Por este motivo, tivemos que optar por uma versão mais recente.

Esta página em branco foi inserida propositalmente

5. Conclusões

Com base na definição inicial do problema:

"As equipas/clientes que utilizam os BPPs, não possuem uma interface para consultar informações pertinentes a respeito do trabalho desempenhado pelos BPPs. Por limitações do próprio BP, não é possível disponibilizar as informações requeridas pelas equipas de forma ágil e segura através da própria ferramenta.

Quando um BPP inicia a sua atividade até o término da mesma, existem inúmeras informações que são importantes para as equipas. Assim, o principal problema identificado consiste na impossibilidade da ferramenta BP fornecer informações dos BPPs. Em concreto, descreve-se um conjunto de questões associadas a este problema:

1. *A equipa não têm visibilidade do estado do BPP;*
2. *A equipa não têm visibilidade do estado da BPR utilizada pelo BPP;*
3. *A equipa não têm visibilidade do estado da queue de trabalho; e*
4. *A equipa não possui controlo sobre o BPP."*

foram definidos os seguintes objetivos de investigação (RGs):

- RG 1 – Efetuar a revisão de literatura de modo a compreender as principais técnicas de visualização efetiva de dados e desenvolvimento de *Dashboards*.
- RG 2 – Elaborar a especificação de requisitos da aplicação *Blue Prism: Dashboard*;
- RG 3 – Definir a Arquitetura do Sistema;
- RG 4 – Definir os principais Casos de Uso;
- RG 5 – Definir e desenvolver o Modelo de Dados; e
- RG 6 – Desenvolver o protótipo *Blue Prism: Dashboard.*

Deste modo, relativamente a:

- RG 1, foi realizada a revisão da literatura no Capítulo 2;

- RG 2, foram definidos os requisitos funcionais e não funcionais da aplicação, como se demonstrou no Capítulo 3;

- RG 3 – foi definida a Arquitetura do sistema por meio de uma *tech stack*, como se demonstrou no Capítulo 3;

- RG 4, foram ilustrados os Casos de Uso do Sistema, como ilustrado no Capítulo 3;

- RG 5, foram desenvolvidos os Modelo de Dados necessário ao Sistema, apresentado também no Capítulo 3;

- RG 6, o desenvolvimento do *Blue Prism: Dashboard* foi realizado, como se demonstrou no Capítulo 4;

Por estas razões, como conclusão preliminar, considera-se que o problema identificado foi resolvido e os objetivos de investigação cumpridos.

Este trabalho permitiu demonstrar competências técnicas ao nível da integração de sistemas de informação. A integração de componentes pré-fabricados em soluções existentes é um desafio complexo que pressupõe o domínio de diversas vertentes tecnológicas. Nesse sentido, o autor considera que esse desafio foi ultrapassado.

Como trabalho futuro, pretende-se expandir o número de gráficos e painéis disponíveis no *Dashboard*, acrescentar novos meios de autenticação como por exemplo o *Single sign-on (SSO),* alterações avançadas nos perfis de utilizadores, download dos *logs* em formato Excel, assim como disponibilizar *logs* aos utilizadores não estejam no grupo administradores.

Bibliografia

10 Heurísticas de usabilidade para o design de interface do usuário. (sem data). UX Lib. Obtido 26 de Maio de 2020, de https://uxlib.net/content/method/topic/nielsen.html#%E5%90%AF%E5%8F%91%E5%BC%8F%E8%AF%84%E4%BC%B0-heuristic-evaluation-jakob-nielsen

ÁVILA, R. (2014, Dezembro 8). Dicas importantes para elaborar um dashboard útil e profissional. LUZ Planilhas Empresariais. https://blog.luz.vc/excel/dicas-importantes-para-elaborar-um-dashboard-util-e-profissional/

BANNAN, B. (2013). *The Integrative Learning Design Framework: 114 - 133 An Illustrated Example from the Domain of Instructional Technology* (pp. 114–133).

BARBOSA, A. L. dos S. (2017). DASHBOARD. Centro Universitário de Brasília.

BARDZELL, J., & BARDZELL, S. (2013). What is «critical» about critical design? *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 3297–3306. https://doi.org/10.1145/2470654.2466451

Blue Prism. (2020, maio 19). Wikipedia. Obtido 20 de Junho de 2020, de https://palletsprojects.com/p/flask/

CARVALHO, E. S., & MARCOS, A. F. (2009). Visualização da Informação. Publicações CCG.

CASTELLS, M. (2009). COMUNICACiÓN Y PODER. Alianza Editorial.

ECKERSON, W. W. (2011). Performance Dashboards: Measuring, Monitoring, and Managing Your Business (2.a ed.). John Wiley & Sons, Inc.

FERREIRA, N. M. A. (2012). Visualização de Informação e Visualização Analítica: Mapa de visualização gráfica da informação agregada do país, um sistema de apoio à decisão. ISCTE-IUL.

Flask (framework web). (2019, dezembro 19). Obtido 20 de Junho de 2020, de https://www.fullstackpython.com/flask.html

FOURNIER, D. (2016, Abril 14). Heurísticas de Nielsen — Avaliando a usabilidade de interfaces. https://medium.com/vivareal-ux-chapter/heur%C3%ADsticas-de-nielsen-avaliando-a-usabilidade-de-interfaces-e96f9801cd5

GOMES, P. C. T. (2017, Outubro 16). O QUE É UM DASHBOARD? O GUIA COMPLETO E DEFINITIVO. OpServices. https://www.opservices.com.br/o-que-e-um-dashboard/

KECECI, N., & ABRAN, A. (2001). *An Integrated Measure for Functional Requirements Correctness*.

KLEMMER, S., & KUMAR, J. (2017). Programa de cursos integrados Design de interação

Design de Interação. https://www.coursera.org/specializations/interaction-design

IEEE. (1998). IEEE Guide for Developing System Requirements Specifications.

JAKOBSONE, L. (2017). Critical design as approach to next thinking. *The Design Journal*, *20*(sup1), S4253–S4262. https://doi.org/10.1080/14606925.2017.1352923

JetBrain Survey. (2019). Obtido 20 de Junho de 2020, de https://www.jetbrains.com/lp/python-developers-survey-2019/

JOHANNESSEN, L. K. (sem data). The Young Designer's Guide to Speculative and Critical Design. *Norwegian University of Science and Technology*, 12.

MANNION, P. (2015, Janeiro 12). Optimal Analysis Algorithms are IoT's Big Opportunity. Electronics360. https://electronics360.globalspec.com/article/4890/optimal-analysis-algorithms-are-iot-s-big-opportunity

MARTINS, J. C. C. (2007). *Técnicas Para Gerenciamento de Projetos de Software*.

NIELSEN, J. (1994, Abril 24). 10 Usability Heuristics for User Interface Design. Nielsen Norman Group. https://www.nngroup.com/articles/ten-usability-heuristics/

NIELSEN, J. (1994a). Enhancing the explanatory power of usability heuristics. Proc. ACM CHI'94 Conf. (Boston, MA, April 24-28), 152-158

PEFFERS, K., TUUNANEN, T., GENGLER, C., ROSSI, M., HUI, W., VIRTANEN, V., & BRAGGE, J. (Fevereiro de 2006). The design science research process: A model for producing and presenting information systems research.

SHARMA, N. (2008). The Origin of Data Information Knowledge Wisdom (DIKW) Hierarchy. Google Inc.

SHAW, A. (2016). NAVEGAÇÃO ESTRUTURAL. Usabilidade.gov.pt. https://usabilidade.gov.pt/navegacao-estrutural

TRASSI, V. A. (2016). Visualização da informação: análise de ferramentas web para auxílio a tomada de decisão. UNIVERSIDADE FEDERAL DO PARANÁ.

ZELENY, Milan (1987) Management Support Systems: Towards Integrated Knowledge Management, Human Systems Management

ZIMMERMAN, J., FORLIZZI, J., & EVENSON, S. (2007). Research through design as a method for interaction design research in HCI. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '07*, 493–502. https://doi.org/10.1145/1240624.1240704

Anexos

A. Bibliotecas *Python* utilizadas

A seguir, encontra-se os pacotes utilizados ao longo do projeto, e suas correspondentes versões.

Pacotes responsáveis por criarem, e disponibilizarem os gráficos em tempo real ao utilizador.

dash 1.10.0

dash-core-components 1.9.0

dash-html-components 1.0.3

dash-renderer 1.3.0

dash-table 4.6.2

plotly 4.6.0

Pacotes responsáveis por integrar o *front-end* e *back-end* do sistema, criações de rotas e gestão de pedidos *GET/POST*.

Flask 1.1.1

Flask-Session 0.3.1

Jinja2 2.11.1

jsonschema 3.2.0

requests 2.22.0

Pacotes responsáveis por facilitar as operações *CRUD* realizadas pelo *Dashboard* na base de dados.

Flask-SQLAlchemy 2.4.1

pylint-flask-sqlalchemy 0.1.0

B. Classe “Create_Dash_Queue”

A figura a seguir, exibe a classe “Create_Dash_Queue”, responsável por construir um dos gráficos disponíveis no sistema.

```
def Create_Dash_Queue(self):

    self.dashs.layout = html.Div(children=[
        html.H1(children=''),
            html.Div(id='my-div'),

            -
            dcc.Dropdown(
                    id='graph_select',
                    options=[{'label': i, 'value': i} for i in ['Process', 'Queue']],
                )
        ,
            dcc.Graph(
                id='example-graph',
                figure={

                    'layout': {
                        'BluePrism Queue': 'Completed and Exception Items',
                        'uirevision': True
                    }
                },
                config={
                            'displayModeBar': True,
                            'displaylogo': False,
                            'modeBarButtonsToRemove': [ 'hoverCompareCartesian', 'hoverClosestCartesian', 'toggleSpikelines']
                        },
            ),
            dcc.Interval(
                id='interval-component',
                interval=4*1000, # in milliseconds
                n_intervals=0
            )

    ])

    @self.dashs.callback(Output('example-graph', 'figure'),
    [Input('interval-component', 'n_intervals')]
    )

    def update_metrics(n):

        sql = SQLqueries().Create(session['queue_name'])
        self.cursor.execute(sql)
        self.QueueItems = [item for item in self.cursor]
        self.Completed_Items = list(filter(lambda item: item[1] != None,self.QueueItems))
        self.Exception_Items = list(filter(lambda item: item[1] == None,self.QueueItems))
        self.Completed = Dashboard_Queue(self.Completed_Items).Get_Graph_data()
        self.Exceptions = Dashboard_Queue(self.Exception_Items).Get_Graph_data()

        Graph_data =  [ {'x': self.Completed[0], 'y': self.Completed[1], 'type': 'bar', 'name': 'Completed'},
                        {'x': self.Exceptions[0], 'y': self.Exceptions[1], 'type': 'bar', 'name': 'Exception','marker': { 'color': 'red' }},]
        return {'data': Graph_data,'layout':{'uirevision': True}}
```