FELIPE CARDOSO MAIA NOEL

DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES WEB EM LOW-CODE

UNIVERSIDADE DO ALGARVE
Instituto Superior de Engenharia
2023

FELIPE CARDOSO MAIA NOEL

DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES WEB EM LOW-CODE

Mestrado em Engenharia Eletrotécnica e de Computadores

(Tecnologias de Informação e Telecomunicações)

Trabalho realizado sob a orientação de: Professor Pedro Jorge Sequeira Cardoso

UNIVERSIDADE DO ALGARVE
Instituto Superior de Engenharia
2023

DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES WEB EM LOW-CODE

Declaração de autoria da obra

Declaro ser o autor desta obra, que é original e inédito. Os autores e obras consultados são devidamente citados no texto e constam da listagem de referências

_____

(Felipe Cardoso Maia Noel)

AGRADECIMENTOS

Ao Fernando Guedes orientador na empresa, pelo acompanhamento, apoio, transmissão de conhecimentos, amizade e, aconselhamento profissional e pessoal para o seguir os estudos no Mestrado em Engenharia Eletrotécnica e de computadores (MEEC) e no mercado desenvolvimento web, através das plataformas *low-code*.
Aos meus colegas de mestrado, Raphael Jório, Sergio Brito, Breno Garcia e Annie Louise, por partilharem seus conhecimentos, experiências e por incentivarem a minha continuidade no MEEC.
Aos meus parceiros de empresa e trabalho Andre Guerreiro, Pedro Pires e Cristiana Louro, pelo acompanhamento, apoio, transmissão do conhecimento, amizade, e por passar a experiência de anos de trabalho na área e pelo aconselhamento pessoal.
Aos companheiros de projeto da empresa multinacional de engenharia (que por motivos de confidencialidade, não possível revelar o nome), por dividirem sua experiência, conhecimento sobre as estruturas das aplicações e lógicas, sobre o desenvolvimento de aplicações na plataforma Mendix.
Ao professor Pedro Jorge Sequeira Cardoso orientador académico, pelo apoio, aconselhamento e acompanhamento.
Ao Instituto Superior de Engenharia da Universidade do Algarve, pela opção de escolha de estágio, que dá aos alunos uma visão do mundo corporativo e os encaminha a nível profissional.
Aos meus familiares, em especial aos meus pais, que sempre me apoiaram nas minhas escolhas e nos meus sonhos.
A minha namorada Julia Ramos que acompanhou meus últimos anos na licenciatura, essa caminhada pela pós-graduação e agora a conclusão do mestrado.
Aos meus amigos que sempre me apoiaram e estiveram comigo na luta dos estudos para chegarmos aonde chegamos.

RESUMO

Durante meu estágio na empresa Atos tive a oportunidade de trabalhar com tecnologias inovadoras, incluindo a plataforma de desenvolvimento *low-code* Mendix e a metodologia de e desenvolvimento rápido de aplicações (*Rapid Application Development*, RAD). O meu trabalho envolveu o desenvolvimento web utilizando a referida plataforma Mendix.

Neste contexto, pude adquirir habilidades significativas em criação de aplicativos de negócios eficientes e personalizados, utilizando a abordagem *low-code* para acelerar o processo de desenvolvimento. Os principais pontos do trabalho desenvolvidos durante o estágio foram:

1. **Desenvolvimento de Aplicativos Web**: Trabalhei no desenvolvimento de aplicações web personalizadas para clientes de uma empresa multinacional de engenharia (que por motivos de confidencialidade, não possível revelar o nome), utilizando a plataforma Mendix. Esses aplicativos atendiam a diversas necessidades, desde gerenciamento de processos de negócios até soluções de automação.

2. **Metodologia RAD**: Fui exposto à metodologia RAD, que enfatiza ciclos de desenvolvimento curtos e interativos. Isso me permitiu colaborar de perto com os *stakeholders* e adaptar rapidamente os aplicativos às suas necessidades em constante evolução.

3. **Colaboração Interdisciplinar**: Trabalhei em estreita colaboração com uma equipa interdisciplinar, incluindo desenvolvedores, analistas de negócios e designers de UX/UI. Essa experiência me proporcionou uma visão abrangente do ciclo de vida completo de desenvolvimento de *software*.

4. **Customização e Integração**: Fui responsável por personalizar os aplicativos Mendix para atender aos requisitos específicos dos clientes. Além disso, integrei esses aplicativos com sistemas legados e outras ferramentas, garantindo uma operação suave e eficiente.

Em resumo, o estágio na empresa Atos foi uma oportunidade valiosa para adquirir conhecimento e experiência no desenvolvimento web utilizando a plataforma Mendix e a metodologia RAD. Aprendi a colaborar eficazmente em uma equipa multidisciplinar, a adaptar rapidamente os aplicativos às necessidades dos clientes e a entregar soluções de alta qualidade. Esta experiência fortaleceu minha base de conhecimento em desenvolvimento de software e preparou-me para desafios futuros na área das tecnologias da informação.

Palavras-chave: RAD, plataformas *low-code*, Mendix, Desenvolvimento Web**.**

ABSTRACT

During my internship at the company Atos, I had the opportunity to work with innovative technologies, including the low-code development platform Mendix and the Rapid Application Development (RAD) methodology. My work involved web development using the mentioned Mendix platform.

In this context, I was able to acquire significant skills in creating efficient and customized business applications, using the *low-code* approach to accelerate the development process. The main aspects of the work carried out during the internship were:

1. Web Application Development: I worked on the development of customized web applications for clients of a multinational engineering company (which, for confidentiality reasons, I cannot disclose the name of), using the Mendix platform. These applications addressed various needs, from business process management to automation solutions.
2. RAD Methodology: I was exposed to the RAD methodology, which emphasizes short and interactive development cycles. This allowed me to collaborate closely with stakeholders and quickly adapt applications to their constantly evolving needs.
3. Interdisciplinary Collaboration: I worked closely with an interdisciplinary team, including developers, business analysts, and UX/UI designers. This experience provided me with a comprehensive view of the entire software development lifecycle.
4. Customization and Integration: I was responsible for customizing Mendix applications to meet specific client requirements. Additionally, I integrated these applications with legacy systems and other tools, ensuring smooth and efficient operation.

In summary, the internship at Atos was a valuable opportunity to gain knowledge and experience in web development using the Mendix platform and the RAD methodology. I learned to collaborate effectively in a multidisciplinary team, quickly adapt applications to client needs, and deliver high-quality solutions. This experience strengthened my foundation in software development and prepared me for future challenges in the field of information technology.

Keywords: RAD, *low-code* platforms, Mendix, Web Development.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÓNIMOS

AC	*Acceptance Criteria*
AI	*Artificial intelligence*
API	*Application Programming Interface*
BDD	*Behavior Driven Development*
BPMN	*Business Process Model and Notation*
CASE	*Computer Aided Software Engineering*
CI/CD	*Continuous Integration/Continuous Delivery*
CX	*Customer experience*
DSLs	*Digital Subscriber Lines*
DSMLs	*Domain-Specific Modeling Language*
DXP	*Digital Experience Platforms*
ERP	*Enterprise Resource Planning*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
IT	*Information Technology*
LCDPs	*Low-code software development platforms*
MBE	*Model-Bases Engineering*
MDA	*Model-Driven Architecture*
MDD	*Model Driven Development*
MDE	*Model Driven Engineering*
OMG	*Object Management Group*
ORM	*Object-Relational Mapping*
PaaS	*Platform as a Service*
RTIM	*Real-Time Interaction Management*
UX/UI	*User Experience/User Interface*

1

INTRODUÇÃO

Este capítulo ilustra um breve enquadramento do presente relatório de estágio, através de uma descrição do contexto, da empresa, bem como abordagem principal do trabalho realizado ao longo do estágio.

1.1 CONTEXTO DO ESTÁGIO

O estágio realizado na empresa Atos representou um marco significativo na minha jornada acadêmica e profissional, oferecendo uma experiência rica no campo do desenvolvimento web em *low-code*, com foco na plataforma Mendix. Este período de estágio foi enriquecido pela imersão nas metodologias ágeis, como Scrum e Agile, e pela oportunidade de explorar as complexidades do *outsourcing*.

Durante essa experiência, tive o privilégio de contribuir para o desenvolvimento de aplicações que abrangiam diversas áreas cruciais para as operações empresariais. Estas incluíam sistemas de gestão de recursos, ferramentas para controle de encomendas e contratos, soluções interativas de *live chat*, sistemas de gerenciamento de recursos humanos, controle de hierarquia organizacional e otimização de processos. Cada projeto era um quebra-cabeça único, desafiando minha capacidade de aplicar conhecimentos teóricos em situações do mundo real.

Essa experiência prática foi profundamente inspiradora e catalisou minha pesquisa de mestrado. Ela forneceu a base sólida e a compreensão aprofundada necessárias para abordar questões complexas relacionadas à transformação digital e ao desenvolvimento de software num ambiente empresarial dinâmico. Como resultado, estou confiante de que este estágio

enriqueceu consideravelmente minha formação acadêmica e preparou-me de forma abrangente para os desafios futuros na área de tecnologia e negócios.

1.2 EMPRESA

Como referido anteriormente, o estágio apresentado neste relatório decorreu na Atos, líder em segurança digital e descarbonização, através de uma gama de soluções digitais, juntamente com um vasto leque de serviços de consultoria [1]. Sendo referência na Europa em cibersegurança, *cloud* e computação de alto desempenho, a empresa oferece soluções personalizadas de ponta a ponta para todas as indústrias em 71 países. Entre as várias localizações a nível mundial, a Atos conta com um escritório localizado no Campus da Penha da Universidade do Algarve, no edifício UAlgTech.

A Atos tem nos objetivos, ajudar a projetar o futuro da tecnologia da informação, tendo as suas competências e serviços como suportes para o desenvolvimento do conhecimento, educação e investigação, numa abordagem multicultural e contribuindo para o desenvolvimento da excelência científica e tecnológica.

A Atos é uma empresa global líder em serviços de transformação digital, com uma equipa de mais de 110 mil funcionários em todo o mundo. Em Portugal, fazemos parte dessa presença global com uma equipa de 292 profissionais dedicados. Dentro desse contingente, 132 fazem parte da divisão do centro de desenvolvimento local (*Local Development Center*, LDC) presente na Universidade do Algarve, contribuindo ativamente para o nosso compromisso de excelência e inovação em Portugal e além-fronteiras. Esta dimensão e diversidade de talentos são essenciais para o nosso sucesso contínuo e a capacidade de atender às necessidades dos nossos clientes em todo o mundo.

Além disso, a secção da Atos que opera em Faro, oferece uma ampla gama de serviços de tecnologia da informação e soluções de transformação digital para clientes locais e internacionais. Esses serviços incluem consultoria em tecnologia da informação (TI), desenvolvimento de software, *cloud*, gerenciamento de infraestrutura, segurança cibernética e muito mais. O escritório da Atos em Faro desempenha um papel importante na entrega desses serviços, contribuindo para o sucesso da empresa na região e além.

1.3 OBJETIVOS DO ESTÁGIO

Foi no referido contexto que o estágio descrito neste relatório se realizou. Durante o período de estágio na empresa, foi apresentada uma ampla gama de desafios e metas empolgantes. Uma das tarefas cruciais que desempenhei foi a programação *low-code* na plataforma Mendix.

Isso envolveu a criação de aplicativos de maneira eficiente e ágil, aproveitando ao máximo as capacidades avançadas dessa tecnologia. A minha experiência com o Mendix permitiu-me entregar soluções mais rapidamente do que seria possível com métodos tradicionais de desenvolvimento de software.

Além disso, tive a valiosa oportunidade de adotar a metodologia Agile/Scrum na nossa equipa de projeto. Essa abordagem altamente colaborativa e ágil revolucionou nossa forma de trabalhar. Com ciclos de desenvolvimento curtos, revisões frequentes e priorização flexível, conseguimos entregar resultados de alta qualidade de maneira mais eficaz e alinhada com as necessidades em constante evolução dos nossos clientes e do mercado. Essa experiência contribuiu significativamente para o meu crescimento profissional e para a eficácia da equipa como um todo.

Podemos então resumir os objetivos do estágio nos seguintes itens:

- Adquirir competências de desenvolvimento web e mobile, através de plataforma de programação *low-code* Mendix.
- Adquirir competências nas áreas de metodologias de trabalho, como o Scrum, para acompanhamento do progresso no desenvolvimento do software em relação aos requisitos.
- Adquirir competências dentro da plataforma Mendix direcionada ao *Front-end*, através de CSS, JavaScript e jQuery, para uma melhor interface para o usuário.

1.4 ORGANIZAÇÃO DO RELATÓRIO

A organização do relatório é fundamental para apresentar de forma clara e lógica todas as informações relevantes sobre o estágio. Deste modo o relatório foi definido nas seguintes secções:

- Secção 1 - Introdução: São abordados o contexto do estágio, detalhando a empresa onde ocorreu, seu setor e relevância. Além disso, é apresentado o objetivo do estágio, destacando sua importância na formação e desenvolvimento profissional adquirido no estágio.
- Secção 2 - Ferramentas de Desenvolvimento Rápido: É realizada a identificação das ferramentas de desenvolvimento *low-code*. Além disso, são abordados os critérios que devem ser considerados na escolha das plataformas para projetos de desenvolvimento de software, bem como os processos de desenvolvimento, destacando a simplificação e agilidade que o ambiente de *low-code* proporciona. A análise aprofundada da arquitetura dessas soluções e das plataformas abordadas fornece uma visão abrangente

das ferramentas essenciais para o desenvolvimento eficiente no contexto *low-code*, tendo como exemplos OutSystems, PegaSystems e Mendix.

- Secção 3 - Metodologias de Gerenciamento de Desenvolvimento de Software: São explorados diversos modelos de desenvolvimento de software, como o Modelo em Cascata, Modelo em Espiral, RAD, Kanban e Scrum. Cada modelo é detalhado em termos das características, metodologias e aplicações. A escolha entre esses modelos é discutida com base nos requisitos de projetos, prazos e flexibilidade necessária para adaptação às mudanças, proporcionando aos leitores informações valiosas para tomar decisões informadas.
- Secção 4 - Casos Desenvolvidos: São apresentados casos de estudo nos quais são feitas as caracterizações detalhadas do negócio abordado. Para cada caso, são analisados os requisitos funcionais, que incluem funcionalidades, e os requisitos não funcionais, como desempenho e segurança. Essa análise abrangente serve como base para o desenvolvimento de soluções adequadas a cada cenário de negócios.
- Secção 5 - Implementação: São apresentados os conceitos da plataforma Mendix, descrevendo seus princípios fundamentais e funcionalidades chave. Posteriormente, são apresentadas as soluções desenvolvidas com Mendix para os casos de estudo mencionados anteriormente, analisando como a plataforma foi aplicada para atender aos requisitos. Também é realizada uma análise da eficácia e adaptabilidade do Mendix nos cenários abordados.
- Secção 6 - Avaliação Crítica do Estágio: É realizada uma avaliação aprofundada da experiência de estágio, abordando as competências adquiridas, a correspondência com os objetivos estabelecidos, desafios enfrentados e estratégias de superação. Essa análise fornece *insights* cruciais para o desenvolvimento profissional futuro.
- Secção 7 – Conclusão: Finalizo o relatório com um breve resumo dos principais pontos a destacar, as lições aprendidas e os resultados alcançados durante o estágio.

2

FERRAMENTAS DE DESENVOLVIMENTO RÁPIDO

Este capítulo descreve os princípios gerais das ferramentas *low-code*, a sua arquitetura, o processo de desenvolvimento aplicado a essas ferramentas e três das principais plataformas *low-code* do mercado.

Nos últimos anos, houve uma notável mudança de foco na pesquisa de software no contexto do RAD. O enfoque central passou a ser a criação de abstrações que possibilitam uma ênfase maior na modelagem do sistema operacional, em contraposição à ênfase tradicional em questões como memória e dispositivos de rede. Essas abstrações carregam um potencial significativo para simplificar o processo de desenvolvimento.

Essa concepção de abstração aplica-se tanto a linguagens de programação quanto a ferramentas. À medida que o software evolui, torna-se evidente que as linguagens de programação também evoluíram, capacitando os desenvolvedores para se afastarem da complexidade associada à programação direta em plataformas específicas e linguagens de programação tradicionais.

Ao nível de abstração de ferramentas, a evolução começou na década de 1980 com o método de engenharia de software assistida por computador (*Computer-Assisted Software Engineering*, CASE) para implementar dispositivos e critérios, facilitando o desenvolvimento de software [2]. As ferramentas CASE são grandes aliadas na implementação do método RAD. De facto, como será visto no decorrer do presente relatório, o método RAD define que as ferramentas de desenvolvimento são consideradas um elemento essencial no processo de desenvolvimento rápido de aplicação. No entanto, esses programas evoluíram e com isso permitem que este tipo de desenvolvimento seja aplicado às “ferramentas de baixo código”.

O termo "ferramentas de baixo código" refere-se a plataformas ou sistemas que permitem criar aplicativos ou softwares com um mínimo de programação tradicional, i.e., com linhas de código. Essas ferramentas visam simplificar o processo de desenvolvimento, fornecendo uma interface visual e componentes pré-construídos com funções pré-definidas, facilitando a sua utilização na criação de lógicas de programação e páginas. O nome "baixo código" reflete a ideia de que menos codificação tradicional é necessária, tornando o desenvolvimento de aplicativos mais acessível a usuários com menos experiência em programação.

No restante deste capítulo exploraremos com mais profundidade os aspetos fundamentais relacionados à identificação, origem, seleção, processos, arquitetura e plataformas de desenvolvimento no contexto do desenvolvimento de software com a abordagem *low-code*. Vamos examinar como esses elementos se integram na prática para impulsionar o desenvolvimento rápido e eficiente de aplicações, alinhando-se com a evolução das abstrações e abordagens de modelagem que mencionamos anteriormente. Isso permitirá compreender melhor como o *low-code* está transformando a maneira como desenvolvemos software e como ele pode ser aplicado de maneira eficaz em diversos cenários.

2.1 IDENTIFICAÇÃO DE FERRAMENTAS DE DESENVOLVIMENTO

A identificação das ferramentas de desenvolvimento deve seguir alguns critérios importantes para a classificação dos projetos. De acordo com a *Dzone community* [3] a escolha da melhor ferramenta para o desenvolvimento de um projeto baseia-se nos seguintes critérios:

- Aplicabilidade da ferramenta;
- Velocidade de desenvolvimento;
- Flexibilidade e capacidade de gestão.

A aplicabilidade da ferramenta está diretamente ligada ao tipo de negócio ou ao objetivo que se pretende atingir com o software. A velocidade de desenvolvimento está relacionada com o prazo definido para o projeto ficar pronto para uso do cliente final. A flexibilidade e capacidade de gestão tem como objetivo que o software possa ser melhorado e gerido por uma ou mais pessoas, de acordo com o planejamento definido para a aplicação.

Tendo em conta os critérios citados anteriormente, é apresentada na Figura 1 abaixo uma classificação para as diversas plataformas. Sendo uma imagem retirada do *Dzone community* [3], teve o objetivo de fazer uma análise imparcial para informar e categorizar ferramentas a partir dos requisitos. É possível observar que uma das plataformas citadas nesta imagem é a plataforma em questão do presente relatório (Mendix).

De acordo com o esquema é possível concluir que aumentar a velocidade de desenvolvimento implica necessariamente diminuir a flexibilidade e a capacidade de gestão. Essa relação intrínseca entre velocidade, flexibilidade e gestão é uma das principais considerações em qualquer projeto de desenvolvimento de software. Por outro lado, aumentar a flexibilidade e a capacidade de gestão amplia consideravelmente o raio de aplicabilidade de qualquer projeto. A capacidade de se adaptar rapidamente às mudanças do ambiente de negócios e às demandas do mercado é uma vantagem competitiva inegável. No entanto, quando se busca acelerar o processo de desenvolvimento de software, é importante estar ciente de que essa abordagem pode resultar em uma redução do raio de aplicabilidade do produto final.

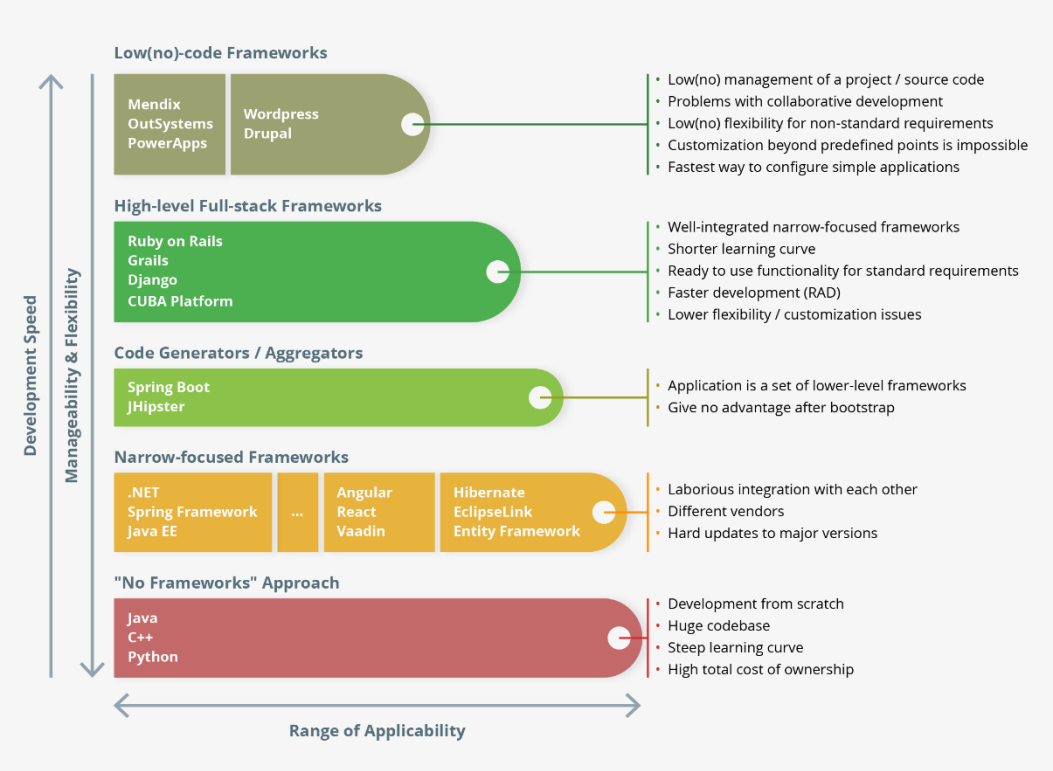


Figura 1. Classificação de ferramentas segundo a *DZone community* (Fonte: [3])

Dentro do contexto da identificação das ferramentas de desenvolvimento, o *Dzone community* separou em 5 categorias as plataformas de desenvolvimento de software, a partir dos critérios mencionados [3]:

- ***"No Frameworks" Approach***: A abordagem "No Framework" possui que ampla aplicabilidade e a mais alta capacidade de gerenciamento e flexibilidade que se pode alcançar. Obviamente, esta é a maneira mais lenta de desenvolver aplicativos corporativos, porque todas as "rodas precisam ser reinventadas". Essa abordagem envolve a utilização de programação tradicional de software para construção de

estruturas, bibliotecas e conjuntos de ferramentas que serão utilizados para facilitar o desenvolvimento de software em menor tempo. O JavaScript "puro" pode ser citado como exemplo de utilização "No Framework".

- ***Narrow-focused Frameworks***: Esta classe tem o maior número de linguagens ou plataformas de desenvolvimento web. Tudo isso visa simplificar o desenvolvimento de uma pequena área, aumentar o nível de abstração e fornecer uma API clara para o seu domínio. Agrupar mapeamento objeto-relacional (*Object-relational mapping*, ORM), *middleware*, interface de usuário (*User Interface*, UI), mensagens e outros tipos de estruturas não é uma tarefa trivial, no entanto, permanecer nessa categoria tem uma combinação satisfatória de velocidade de desenvolvimento e alta flexibilidade. O .Net (plataforma de desenvolvimento de código aberto) é um exemplo de aplicabilidade do "*Narrow-focused Frameworks*" [3].
- ***Code Generators/Aggregators***: No contexto de desenvolvimento de software, os geradores de código (*Code Generators*) e agregadores (*Aggregators*) são ferramentas ou abordagens que automatizam a criação de código ou a geração de estruturas de código. Na verdade, eles configuram seus aplicativos de acordo com o *Narrow Focus Frameworks*. Tendo como exemplo de gerador de código o *Swagger Codegen* [4]. O *Swagger* é uma ferramenta popular para documentar interface de programação de aplicativos (*Application Programming Interface*, APIs), sendo que o *Swagger Codegen* permite gerar automaticamente código-fonte em várias linguagens de programação com base na especificação da API criada. Por exemplo, se tiver definido uma API no *Swagger* usando notação de objeto JavaScript (*JavaScript Object Notation*, JSON), o *Swagger Codegen* pode gerar automaticamente o código-fonte em Java, Python, C#, entre outras linguagens. Isso economiza tempo e esforço, pois o desenvolvedor não precisa escrever manualmente o código de integração da API, uma vez que ele é gerado automaticamente.
- ***High-Level Full-Stack Frameworks***: Esta classe apresenta o próximo nível de abstração relativamente ao *Narrow Focus Frameworks*. Neste caso, a infraestrutura está pronta para criar aplicativos de negócios em escala muito mais rapidamente. Por exemplo a plataforma CUBA [5], oferece uma arquitetura completa para a construção de aplicações de três camadas, incluindo características comuns amplamente utilizadas em aplicações corporativas, que contenham:

componentes sensíveis a dados, armazenamento de arquivos e módulos de alto nível de autenticação e autorização. Esta categoria é mais adequada para o desenvolvimento de sistemas semelhantes ao planejamento de recursos empresariais (*Enterprise Resource Planning*, ERP), aplicativos de linha de negócios ou software personalizado para unidades de negócios, devido a sua estrutura em camadas.

- ***Low/No Code Frameworks**:* Nesta categoria perde-se totalmente o controlo sobre a base de código, desenvolvendo apenas pequenos pedaços de lógica nos pontos de extensão pré-definidos. Assim, se tal estrutura satisfaz os seus requisitos em termos de desempenho, opções de implementação, componentes UI, integrações, etc., esta seria a forma mais rápida de configurar a sua aplicação. Contudo, podem acontecer limitações arquitetónicas que não sejam possíveis de ultrapassar em desenvolvimentos mais complexos. PegaSystems pode ser citado como um exemplo de plataforma que possui limitações no que se refere a sua aplicabilidade e adaptação a diferentes tipos de mercados.

A identificação destas ferramentas possibilita concluir que a escolha dos recursos do *low-code* permite uma maior rapidez no desenvolvimento. No contexto das empresas de TI, a decisão pela utilização deste tipo de ferramenta está, em parte, ligada à necessidade na redução no tempo de desenvolvimento de aplicações.

Em conclusão, a análise da velocidade de desenvolvimento em relação às plataformas *low-code*, conforme abordada no estudo [6], destaca a importância de considerar o contexto específico em que essas ferramentas são utilizadas. Conclui-se que as limitações do *low-code* podem variar significativamente, dependendo das necessidades e dos requisitos de segurança de um projeto.

Em alguns casos, onde a velocidade e a agilidade são fatores críticos, as plataformas *low-code* oferecem uma vantagem considerável, permitindo o desenvolvimento rápido e eficiente de aplicações. No entanto, é importante reconhecer que essas vantagens podem estar acompanhadas de desafios relacionados à segurança.

A implementação de projetos que dependem de um alto nível de segurança pode não ser a escolha ideal para o desenvolvimento *low-code*, uma vez que alguns riscos de segurança podem estar associados a essas plataformas. Portanto, a decisão de adotar o *low-code* deve ser cuidadosamente ponderada, considerando as necessidades específicas de cada projeto e os compromissos em relação à segurança. É fundamental encontrar um equilíbrio entre a

velocidade de desenvolvimento oferecida pelo *low-code* e os requisitos rigorosos de segurança para garantir o sucesso do projeto.

2.2 ORIGEM DAS PLATAFORMAS DE DESENVOLVIMENTO *LOW-CODE*

A proporção do *low-code* deu-se pela agência de pesquisa de mercado *Forrester Research* em 2014, onde definiu as plataformas como sendo: "Plataformas que permitem a entrega rápida de aplicativos de negócios com um mínimo de codificação manual e investimento inicial mínimo em configuração, treinamento e implantação", o que tornou o *low-code* uma alternativa para as organizações que visam aumentar a rapidez no desenvolvimento e diminuir os prazos de entrega.

As plataformas de desenvolvimento *low-code* (*low-code development platforms*, LCDPs) são disponibilizados na *cloud* a partir do modelo plataforma como serviço (*Platform-as-a-Service,* PaaS), o qual permite a programação e o download de aplicações através de passos que requerem pouco ou nenhum código [6].

As plataformas *low-code* são ambientes que possibilitam desenvolver aplicações a partir interfaces, permitindo que pessoas com pouco conhecimento tecnológico criem aplicações.

Esse recurso permite que utilizadores finais sem conhecimentos prévios de programação, contribuam para o desenvolvimento de aplicações, devido aos recursos simplificados presentes nas plataformas *low-code*. Para além disso, a pouca implementação de código permite que os programadores se concentrem na apreciação das funcionalidades e na lógica necessária, reduzindo o tempo de adaptação à linguagem de programação e erros de código verificados no desenvolvimento tradicional.

Por outro lado, embora as plataformas de *low-code* ofereçam muitas vantagens para pessoas com pouco conhecimento tecnológico e acelerem o processo de desenvolvimento de aplicações, também apresentam algumas limitações e desafios:

- Limitações de personalização;
- Escalabilidade;
- Custos;
- Segurança.

Em resumo, as plataformas de *low-code* são uma ferramenta poderosa para acelerar o desenvolvimento de aplicações, permitindo que utilizadores finais sem conhecimentos prévios de programação contribuam de forma significativa. No entanto, é importante estar ciente das limitações e desafios associados a essas plataformas e avaliar se elas são apropriadas para atender às necessidades específicas de um projeto de desenvolvimento de software.

Essas plataformas *low-code* disponibilizam um ambiente para que seus utilizadores possam criar aplicações, através da interação com interfaces gráficas, diagramas e linguagens declarativas, no lugar dos tradicionais ambientes de programação. No âmbito das LCPD são considerados os princípios da engenharia orientada por modelo (*model-driven engineering*, MDE), a qual tem sido adotada em diversos contextos, baseando-se na automação, análise e abstração de modelos e meta modelos.

O MDE, é uma conceção da realidade que possui capacidade de antecipar o comportamento de um processo de desenvolvimento, estando presente ao longo de todo o processo, sendo o principal foco o desenvolvimento de software com base nos modelos MDE e *model-driven development* (MDD) [2], [7]. Este modelo é subconjunto do *model-bases engineering* (MBE) [8], onde os modelos não conduzem o processo MDD (ver Figura 2), o qual corresponde a uma referência de desenvolvimento que serve a modelos como elemento principal em processo de desenvolvimento. O *model-driven architecture* (MDA) [9] corresponde a um subconjunto de MDD, sendo uma abordagem mais específica proposta pelo *Object Management Group* (OMG).

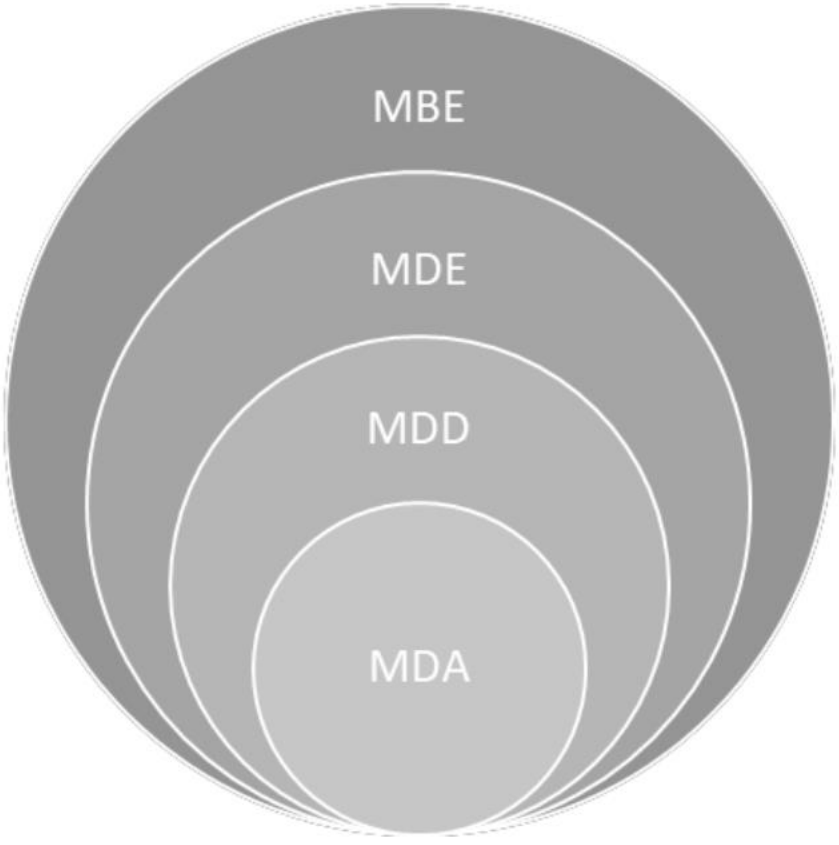


Figura 2. Estrutura *model-driven* (Fonte: [10])

Neste contexto, as soluções e outros mecanismos são gerados de forma automática, a partir dos seus modelos. O MDE aborda os seguintes aspetos:

- ***Domain-specific modeling languages (DSML* ou *DSL)***: correspondem a linguagens de modelagem específicas, as quais são responsáveis por formalizar a estrutura (ver Figura 2), o comportamento e os requisitos do aplicativo em domínios específicos. Essa abordagem segue as abstrações e a semântica do domínio, permitindo que os modeladores percebam que estão trabalhando diretamente com os conceitos do mesmo, através de 3 aspetos: os conceitos e regras do domínio (sintaxe abstrata); a notação usada para representar esses conceitos – seja textual ou gráfica (sintaxe concreta); e a semântica da linguagem (ver Figura 3).
- ***Transformation engines and generators***: esses recursos permitem analisar e sintetizar certos aspetos do modelo, como código-fonte, entradas de simulação e representações de modelos alternativos. O processo de transformação baseia-se em correção por construção (*correct-by-constructor*), contrariamente aos demais processos de desenvolvimento tradicionais que se baseiam em construção por correção (*construction-by-correction*). Esses modelos de *model-driven* são utilizados para traduzir modelos de origem (*source models*) para um ou mais modelos alvo (*target models*), tendo em conta determinadas regras. Essas transformações podem ser modelo para modelo (*Model-To-Model*) ou modelo para texto (*Model-To-Text*) (ver Figura 4).

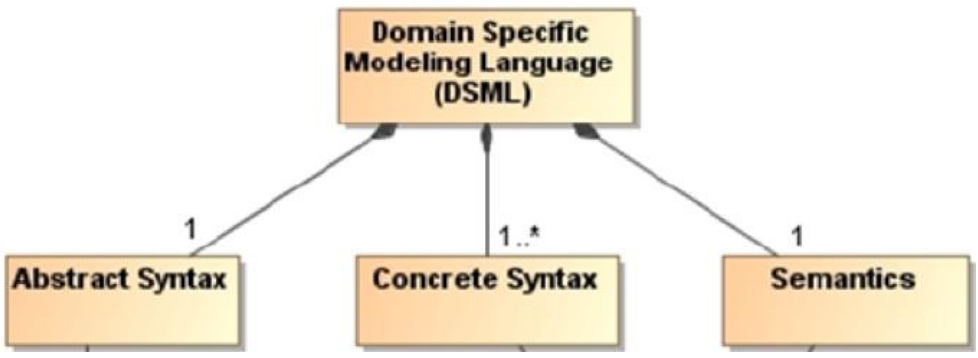


Figura 3. Diagrama DSML (Fonte: [11])

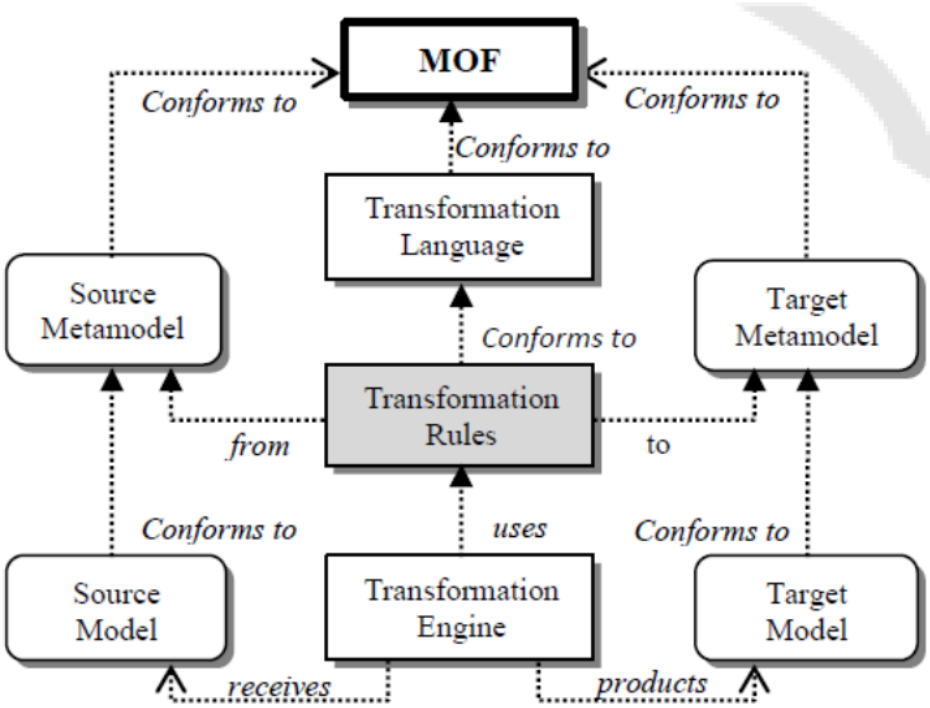


Figura 4. Diagrama de transformação (Fonte: [12])

2.3 ESCOLHA DA PLATAFORMA *LOW-CODE*

Atualmente, prevalece o entendimento de que o modelo de desenvolvimento *low-code* apresenta maneiras mais práticas, dinâmicas e produtivas, sendo uma execução das estratégias de inovação e de transformações digitais. A prática do uso do desenvolvimento *low-code* representa uma rutura em relação à abordagem da programação tradicional, porém o desenvolvimento *low-code* exige uma mudança de cultura.

Uma pesquisa realizada pela plataforma de desenvolvimento Mendix, os profissionais estão cientes da dimensão e do crescimento dessa nova abordagem no desenvolvimento. Quase dois terços (64%) dos colaboradores da área de TI entrevistados concordam que *low-code* tem sido uma solução de desenvolvimento alternativa de trabalho. Os números apontam também que 59% dos projetos que usam o *low-code* são fruto de um trabalho colaborativo entre grupos de negócios e de TI [13].

Em 2021 foi realizado um estudo conduzido pela Mendix, o qual resultou num relatório "*State of Low-code 2021*" [14] tendo sido um dos estudos mais abrangente do setor sobre a adoção do método. Foram ouvidos em torno de 2.025 profissionais de TI dentre eles líderes e desenvolvedores profissionais de TI em 5 países (Estados Unidos da América, China, Bélgica,

Reino Unido, Holanda). Os resultados apontam tendências importantes no contexto atual de desenvolvimento de software:

- 77% das empresas dos cinco países no estudo já adotaram o *low-code*;
- 75% dos líderes de empresas de TI dizem ser uma tendência que não se pode deixar passar;
- Um terço dos entrevistados disse que o *low-code* está acelerando o uso de inteligência artificial (IA), internet das coisas (IoT) e *Big Data*.

Esta pesquisa feita apresenta perceções interessantes sobre o comportamento das empresas, tendo em conta que, dentre as adeptas ao desenvolvimento *low-code*, mais da metade (56%) já utilizam aplicações desenvolvidas em plataformas *low-code* e que, de acordo com o estudo, 51% dos desenvolvedores de software dizem que metade de seu trabalho de desenvolvimento diário poderia vir a ser feito numa plataforma de desenvolvimento de *low-code*. Com isso nota-se que o uso deste tipo de ferramenta representa de facto um ganho de produtividade imenso. Em especial, no contexto mundial pós-pandemia e com os novos desafios digitais surgindo no contexto dos negócios, ter as ferramentas certas para corresponder a tamanha demanda de projetos de desenvolvimento de software com rapidez e eficiência, torna-se fundamental. Isso porque, de acordo com o mesmo estudo, 58% dos colaboradores de TI estão entusiasmados com a aceleração da tecnologia. Em suma, o estudo mostra que empresas de todos os setores já entendem que o desenvolvimento de software atual não suporta a demanda crescente e os resultados da pesquisa ilustram tendência:

- 76% dos profissionais de TI entrevistados disseram que a demanda por desenvolvedores atingiu um nível febril;
- O custo do desenvolvimento de software está aumentando, de acordo com 61% das organizações;
- 2/3 dos projetos de software estão sendo entregues com atraso;
- Apesar de trabalhar incansavelmente, 62% das organizações estão percebendo que suas pendências estão aumentando.

Dado esse contexto abordado, migrar para uma abordagem de plataformas de desenvolvimento *low-code* é uma das estratégias mais inteligentes e tende a trazer resultados positivos a curto, médio e longo prazo.

2.4 PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE EM *LOW-CODE*

O processo de desenvolvimento de software através das plataformas *low-code* é um conjunto de atividades necessárias para desenvolver um produto, sendo dividido em algumas etapas que fazem parte da sua estruturação, tipicamente são [15]:

- **Estabelecer requisitos:** primeira etapa no processo de desenvolvimento, cujo objetivo é o levantamento e definição dos requisitos para a solução;
- **Estabelecer as especificações da aplicação:** esta etapa é implementada através de:
    - **Modelagem dos dados:** feita através de interface gráfica, onde é definido o esquema de dados necessários para a aplicação, através da criação das entidades, relações e dependências;
    - **Caracterização do *layout*:** definição das páginas e suas respetivas estruturas, define o nível de acesso de cada tipo de usuário a nível de páginas, entidades, atributos e seus devidos papeis dentro da aplicação;
    - **Determinação de regras e fluxos de trabalho:** o usuário pode gerenciar fluxos de trabalho entre vários formulários ou páginas que exigem operações diferentes nos componentes da interface. Tais operações podem ser implementadas em termos de fluxos de trabalho baseados em recursos visuais e, para esse fim, notações semelhantes às do modelo e notação de processos de negócios (*Business process model and notation*, BPMN) podem ser empregadas.
- **Integração com serviços externos através de API de terceiros:** nesta etapa é estabelecida a conexão com serviços externos, se necessário for, em casos em que a plataforma forneça meios de estabelecer a comunicação;
- **Adaptação da aplicação:** nesta etapa são feitas as personalizações dentro da aplicação, se necessário for, através de código no *front-end* com folha de estilo em cascata (*Cascading Style Sheet*, CSS) ou JavaScript, através da criação de classes;
- **Testes e aceitação**: fase em que são feitos os testes no lado do utilizador para validar se os requisitos pré-estabelecidos foram implementados;
- **Implementação da aplicação:** nesta etapa é feita a implementação da aplicação na web de forma rápida e onde é gerado o localizador uniforme de recursos (*Uniform Resource Locator*, URL) de acesso da aplicação desenvolvida para o usuário final aceder na web.

2.5 ARQUITETURA DE PLATAFORMAS DE DESENVOLVIMENTO *LOW-CODE*

As plataformas de desenvolvimento *low-code* possuem uma arquitetura que consiste em quatro camadas principais [16], como esquematizado na Figura 5.

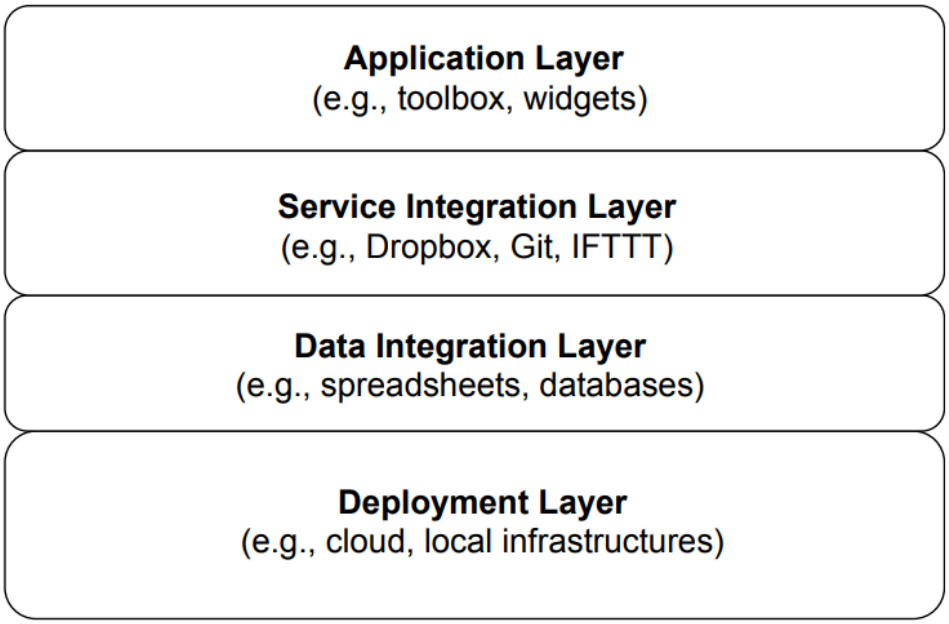


Figura 5. Arquitetura em camadas de plataformas desenvolvimento *low-code* (Fonte: [17])

Na primeira camada temos a apresentação, a qual é a camada que representa o ambiente gráfico que permite aos utilizadores contruírem as interfaces gráficas, de acordo com o objetivo desejado através da construção de modelos. Na segunda camada temos a integração de serviços, responsável por estabelecer a comunicação através das APIs e os mecanismos de autenticação. Na terceira camada temos a integração dos dados, o que permite a manipulação dos dados e integração com outras fontes. Na quarta camada temos o *Deployment Layer*, onde é feita a implementação da aplicação para a *cloud* ou em ambiente (*on-premise*), sendo também responsável por tratar das adaptações da aplicação e trabalhando em conjunto com integração de dados, contribuindo com a camada da integração de serviços.

Considerando esta arquitetura anterior, Sahay [16] detalhou os principais componentes das plataformas de desenvolvimento em *low-code*, onde estão divididos em três níveis, tal como visto na Figura 6.

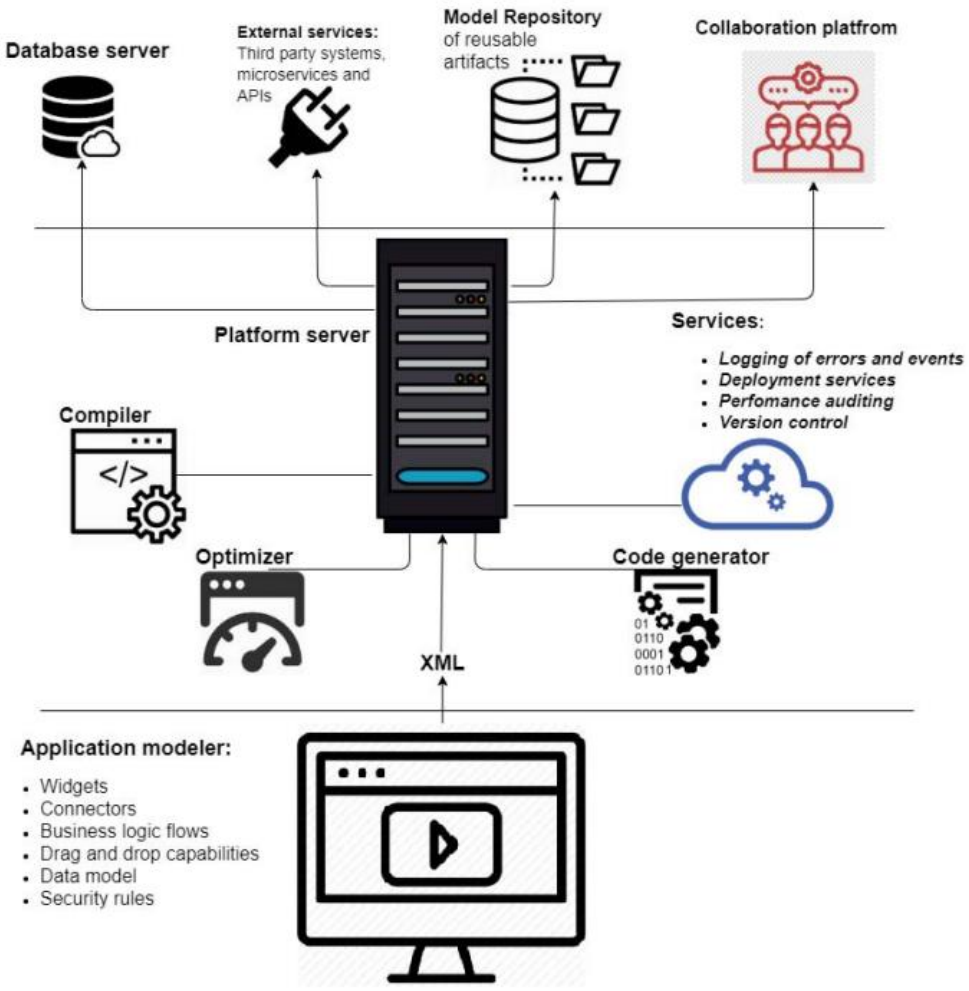


Figura 6. Componentes das plataformas de desenvolvimento *low-code* (Fonte: [17])

Ao nível de desenvolvimento da aplicação, tal como dito anteriormente, a camada do ambiente gráfico (*Application modeler*) representa o ambiente onde os desenvolvedores especificam e constroem os modelos e as abstrações, a partir das necessidades dos clientes, com a possibilidade de correr a aplicação localmente, antes da entrega final. No segundo nível é tratado o que foi feito no modelo anterior, dando início ao processo de geração de código, otimizações e preparação para conexão com outros serviços envolvidos no processo como base de dados, conexão com APIs, micro serviços e modelos de repositórios reutilizáveis. No terceiro nível do servidor de base de dados, os utilizadores e desenvolvedores não possuem controle, considerando que não haja preocupação com o tipo de base dados ou com a integridade dos dados. Já no último nível são apresentados os serviços do segundo nível. Quanto aos micro serviços são configurados no *back-end*, sem qualquer intervenção dos desenvolvedores da aplicação [16].

Essas plataformas possuem repositórios que permitem armazenar estruturas lógicas para serem reutilizados noutras aplicações. As plataformas também facilitam a integração de diferentes metodologias de desenvolvimento, o que possibilita visualizar e acompanhar determinados processos de desenvolvimento e as respetivas soluções [16].

Na Figura 7 é apresentada a arquitetura da plataforma Mendix, atualmente uma das plataformas referência no mercado. Por exemplo, foi eleita pela *Forrester wave*, em 2021, como a líder no seguimento das plataformas *low-code* [18].

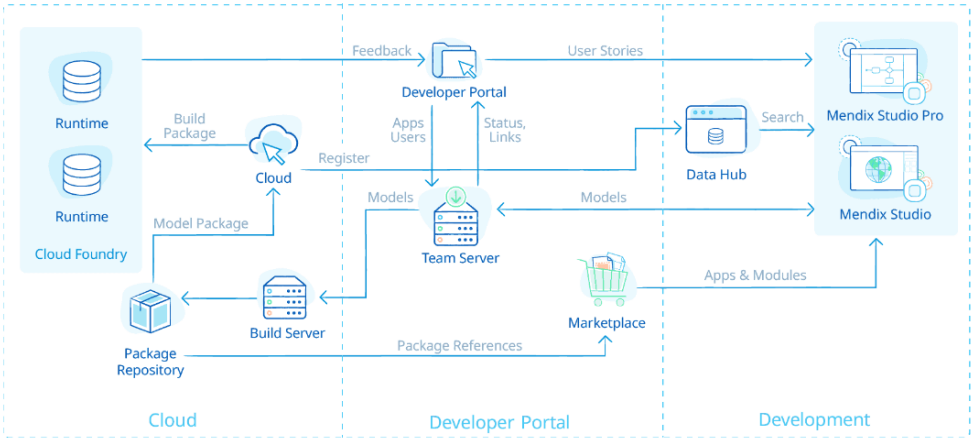


Figura 7. Arquitetura Mendix (Fonte: [19])

Na Figura 7 é possível observar a representação dos componentes citados anteriormente, para os três níveis: primeiro teremos o portal do desenvolvedor o qual apresenta o *team server*, onde é feita a descrição dos modelos de dados e os requisitos de desenvolvimento da aplicação; segundo teremos a *cloud* que apresenta o servidor de construção (*Build Server*) e o repositório de pacotes (*Package Repository*), recebendo os dados do team server e gerando os componentes necessários para aplicação; terceiro teremos o ambiente de desenvolvimento (*Development*), onde é feita a integração dos serviços através da central de dados (*Data Hub*) e o Mendix Studio Pro. A plataforma em questão incorpora e automatiza as operações DevOps, o que permite simplificar o ciclo de vida de desenvolvimento das aplicações [20].

DevOps é um termo que deriva da junção de duas palavras na língua inglesa, desenvolvimento (*Development*) e operações (*Operations*). O conceito DevOps representa uma mudança na cultura do IT, que tem como principal objetivo uma rápida prestação de serviços IT, através de práticas ágeis no contexto de uma abordagem direcionada para o sistema. Possui também um direcionamento em melhorar a colaboração entre operações e as equipas de desenvolvimento, sua implementação usufrui de ferramentas de automação para alavancar estruturas cada vez mais programáveis e dinâmicas, em uma perspetiva de ciclo de vida [19].

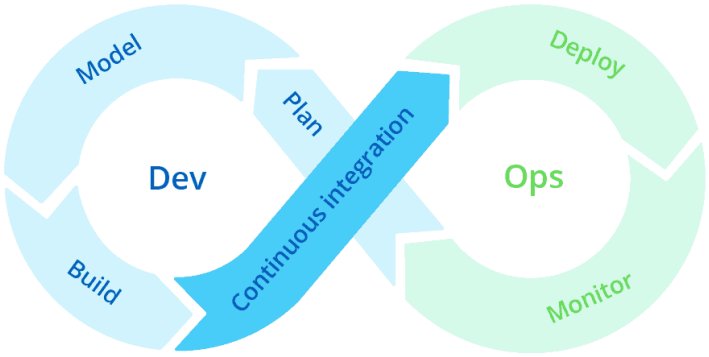


Figura 8. Modelo DevOps em Mendix (Fonte: [20])

Dado o contexto das plataformas *low-code*, a plataforma Mendix apresenta como base na sua estrutura a prática DevOps para que as equipas de desenvolvimento possam adotá-la. A plataforma também fornece ferramentas DevOps *out-of-the-box* para uma integração e entrega contínua (*Continuous Integration*, CI / *Continuous Delivery*, CD), automação de testes e monitorização das APIs da plataforma para integrar com *pipelines* existentes [20].

Isto faz com que seja possível criar, testar e manter as aplicações em ambiente de produção de forma autónoma, através das operações (modelagem, construção, integração continua, implantação e monitoramento) introduzidas pelo DevOps [21] (ver Figura 8).

2.6 PLATAFORMAS DE DESENVOLVIMENTO *LOW-CODE*

As plataformas de desenvolvimento *low-code* foram desenvolvidas, geridas e comercializadas por organizações que tinham como objetivo a criação de aplicações de negócios com entrega contínua e rápida ao cliente, com o mínimo de programação possível. Isso se fez através de interfaces gráficas e componentes direcionados, onde seria necessário apenas arrastar e soltar (*drag and drop*), tendo sido a estratégia adequada para que as plataformas tivessem impacto no mercado e cumprissem com o referido objetivo.

Anualmente a empresa de consultoria Gartner publica um estudo análise de mercado das plataformas *low-code* [22]. Este estudo tem como objetivo avaliar as organizações detentoras das plataformas segundo critérios de negócio e estratégias definidos. A avaliação é baseada em alguns pontos principais, seja no produto, na estratégia de venda e no marketing, nos modelos de negócios e no seu entendimento de mercado. Como resultado obtiveram-se os pontos fortes e fracos de cada organização, sendo possível extrair o diagrama apresentado na

Figura 9, onde as plataformas de desenvolvimento rápido foram divididas entre: líderes, visionárias, desafiadoras e contidas ou de nicho. No referido contexto temos, por exemplo, Mendix e Outsystems como líderes e Appian ou PegaSystems como desafiadoras.

Figura 9. Quadrante mágico para plataformas de desenvolvimento *low-code* (Fonte: [22])

Nas próximas secções iremos detalhar um pouco mais das referidas plataformas, nomeadamente: OutSystems, PegaSystems e Mendix.

2.6.1 OUTSYSTEMS

OutSystems é a líder neste quadrante mágico de acordo com Gartner [22]. Essa liderança é suportada na sua diferenciação no mercado, a qual é baseada na sua capacidade de aumentar a produtividade do desenvolvedor através de uma programação visual e orientada a modelos de aplicação, que podem ser executadas na *cloud* ou em ambiente local. A plataforma também permite fazer integração com IA e DevOps, bem como uma extensão de componentes feitos através de código tradicional.

A plataforma OutSystems permite através de uma interface gráfica, chamada *Framework OutSystems UI*, de desenvolvimento rápido de softwares web e/ou mobile, suportando-se em *templates* existentes ou páginas vazias [15,16]. Esses tipos de interfaces são desenvolvidos a partir do *drag and drop*, de blocos lógicos e ferramentas que compõem as páginas. Nesta plataforma a programação é feita no *Service Studio*, sendo o ambiente determinado para o desenvolvimento visual e de baixo código da plataforma [15,16]. Entretanto existem outros ambientes que permitem a criação de interfaces gráficas ou fluxos de trabalho (*Workflows*), chamados construtor de experiência (*Experience Builder*) ou construtor de fluxo de trabalho (*Worflow Builder*), respetivamente.

A plataforma apresenta um ambiente que permite integração direta com serviços externos o *Integration Studio*, responsável por gerir extensões correspondentes às estruturas, entidades e ações da plataforma [15,16].

OutSystems é capaz de dar suporte ao ciclo de vida de desenvolvimento de soluções implementadas através de:

- **Implantação:** através de um *click* oferece uma implantação automatizada e também gerencia dependências em portfólios de aplicativos inteiros, acelerando a entrega e minimizando riscos;
- **Governança automatizada:** monitora quando os desenvolvedores fazem alterações, verificando o impacto, *bugs* e o desempenho em tempo real;
- **Gerenciamento de configurações:** monitora as aplicações em produção de forma a simplificar a implantação das aplicações e garantir uma entrega contínua.

Esta plataforma tem sido amplamente utilizada para automatizar processos empresariais devido à sua capacidade de aumentar a produtividade e auxiliar os desenvolvedores na criação de aplicativos corporativos modernos. Além disso, ela oferece segurança robusta, experiência de desenvolvimento multidisciplinar e recursos aprimorados pela IA, permitindo um desenvolvimento de aplicativos mais ágil. A Figura 10 representa o visual da plataforma de desenvolvimento OutSystems ao iniciarmos um projeto dentro da ferramenta *Service Studio*.

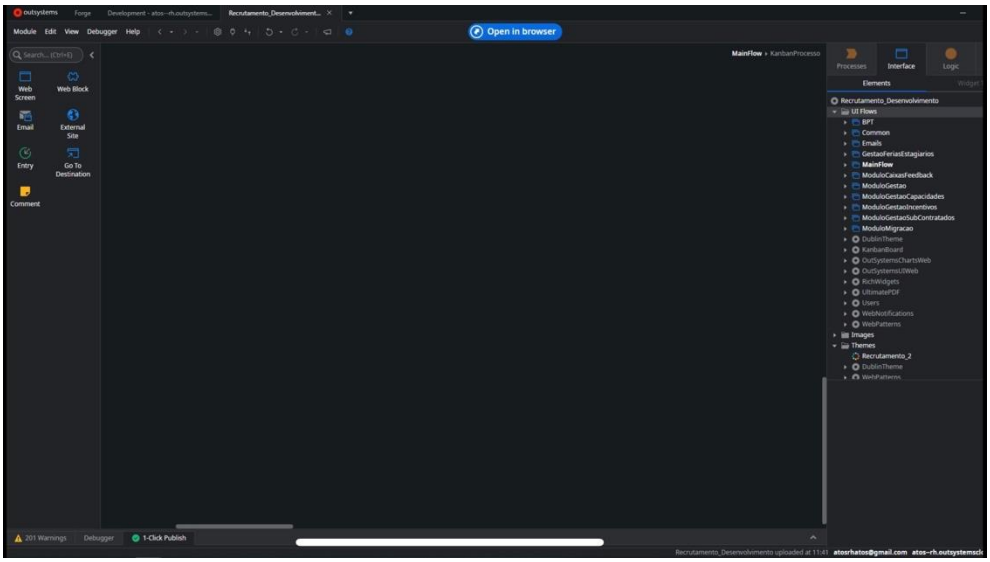


Figura 10. Janela inicial do *Service Studio OutSystems*

2.6.2 PEGASYSTEMS

PegaSystems é uma poderosa plataforma de *low-code* que permite desenvolver aplicações de forma ágil para as principais organizações do mundo, fazendo com que estas possam se adaptar às mudanças. As decisões tomadas dentro da plataforma, são baseadas em AI e automação de fluxos de trabalho para resolução de desafios de negócios, tendo uma arquitetura escalável e flexível para ajudar os desenvolvedores a se focarem no que é mais importante: satisfazer as necessidades das empresas.

PegaSystems, de acordo com estudo feito Gartner, *critial capabilities for sales force autoamation* [22], e em outro estudo realizado pela *The Forrester Wave* [25], deram destaque para a plataforma devido a sua capacidade de automação da força de vendas e também devido ao gerenciamento de interação em tempo real (*Real Time Interaction Management*, RTIM) proporcionado pela plataforma.

A automação disponível nesta plataforma faz com que tenha destaque em oferecer suporte para todos os setores, embora alguns recursos sejam especialmente relevantes para clientes nos setores como seguros, saúde, serviços financeiros e assistência médica. Esse destaque é devido ao gerenciamento de interação em tempo real acoplado a automação dos processos digitais presentes na plataforma, fazendo com que o portefólio de serviços ao cliente seja capaz de otimizar a plataforma através da experiência do utilizador (*Customer Experience*, CX) em todo o ciclo de vida do software.

Além disso, PegaSystems possui uma abordagem nomeada de Pega Express, onde são usadas as melhores práticas para priorizar a colaboração, implementar soluções e criar valor por meio de inovação rápida. Pega Express é dividida em 4 fases, sendo elas [26]:

- **Descobrir:** onde é feito o desenho do roteiro, determina a prontidão do projeto e a definição do resultado;
- **Preparar:** executar as sessões principais, criar lista de pendências e concluir os planos;
- **Construir:** criar a solução, iniciar a configuração e testar os recursos;
- **Adote:** testa a prontidão para entrada em produção, transmissão ao vivo e inovar para o que vem a seguir.

Essa plataforma tem sido amplamente usada em processos desafiadores, como citado anteriormente os setores em que mais se destaca; É possível perceber que são setores que demandam uma iteração e tomada de decisão rápida, dada a necessidade final do cliente. Seus recursos são amplamente explorados por AI e automação [27].

A Figura 11 mostra uma janela do ambiente de desenvolvimento Pega Studio, que oferece uma interface gráfica, ferramentas e recursos que podem ser utilizadas pelos desenvolvedores e profissionais para criar aplicações e configurar processos de negócios.

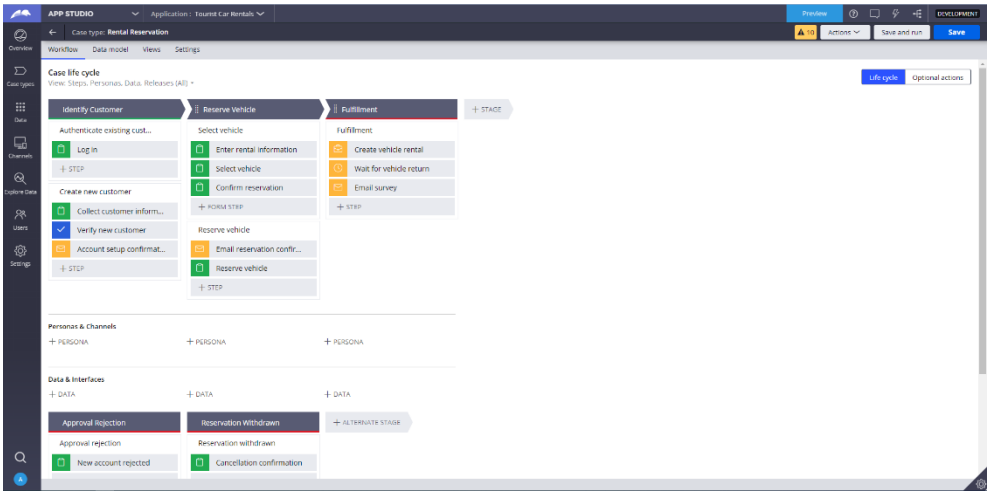


Figura 11. Janela do *PegaSystems Studio*

2.6.3 MENDIX

O Mendix é uma plataforma de desenvolvimento *no-code* e *low-code*, sendo orientada a modelos e com arquitetura baseada em *cloud-native*. A plataforma do Mendix é capaz de suportar o desenvolvimento através de integração, *workflows*, processamento de eventos e utilização de IA. A plataforma fornece recursos empresariais avançados em *low-code* que

beneficiam novos casos de uso, como a IoT. Além disso inclui o *Data Hub* para serviços de dados e eventos, e possui uma ferramenta chamada *MendixAssist Performance Bot*, que deteta proactivamente anti padrões de modelagem e arquitetura que podem afetar o desempenho do programa Gartner [22], [28].

A plataforma possui dois ambientes de desenvolvimento:

- **Mendix Studio:** ambiente intuitivo, visual e sem código (*no-code*), projetado para trazer usuários de negócios não técnicos para o processo de desenvolvimento, incluindo aqueles que nunca desenvolveram um aplicativo antes. Esses desenvolvedores são conhecidos como desenvolvedores cidadãos (*citizen developers*) que possuem um entendimento mais claro dos impulsionadores, objetivos e processos de negócios, e através de um conjunto de modelos para orçamento, controlo de tarefas e tempo, portais e muito mais tornam-se capazes de desenvolver aplicações e softwares. A plataforma ainda conta com o Mendix Assist que fornece recomendações e sugestões dos passos a serem seguidos, acelerando o processo de aprendizado e garantindo a qualidade da aplicação.
- **Mendix Studio Pro:** poderoso ambiente em modelo visual de *low-code*, que oferece aos desenvolvedores profissionais todos os recursos necessários para criar aplicações robustas, complexas e de missão crítica. O Studio Pro, como é chamado, cuida do "*busy work*" que consome muito tempo aos desenvolvedores, permitindo que o mesmo, através de engenharia de software e lógica de negócios, possa construir softwares. O Studio Pro também permite controlo e flexibilidade aos desenvolvedores para personalizar o código. E, tal como no Mendix Studio, também conta com a presença do Mendix Assist, o qual atua como um segundo programador, antecipando os próximos passos, verificando os fluxos lógicos e identificando erros. Esta ferramenta pode oferecer recomendações com 95% de precisão, ajudando a aumentar a produtividade e a eficiência e diminuir em 100 vezes os impactos de custos e tempo com possíveis falhas no software.

A plataforma também possui um repositório de metadados (*metadata*) que permite aos desenvolvedores gerir dados dentro do ecossistema de dados conectados, apropriadamente designado Data Hub [12,13].

Mendix é uma plataforma líder de acordo com quadrante mágico citado anteriormente, diferenciando-se no mercado pelo seu suporte para equipas de fusão, suporte para implementação *multicloud* e oferta locais de serviços nativos de nuvem para desenvolvimento multi-experiência, de acordo com a Gartner [22].

A Figura 12 apresenta a interface do Mendix Studio Pro, projetada para simplificar o processo de desenvolvimento de aplicações de forma visual e eficiente. À esquerda da tela, temos os módulos do projeto, fornecendo uma visão organizada da estrutura do aplicativo em desenvolvimento. No centro da interface, encontra-se o ambiente de desenvolvimento principal, onde os utilizadores podem criar e personalizar elementos de suas aplicações. Na parte inferior da tela, há uma janela que exibe informações cruciais, incluindo as alterações feitas no projeto, erros identificados, dados em uso e outras informações relevantes. À direita, os utilizadores têm acesso a uma paleta de blocos e componentes que podem ser arrastados e soltos no ambiente de desenvolvimento para a construção de páginas ou lógica de aplicativo de forma intuitiva. Essa interface rica em recursos torna o Mendix Studio Pro uma ferramenta poderosa para criar aplicações personalizadas com facilidade.

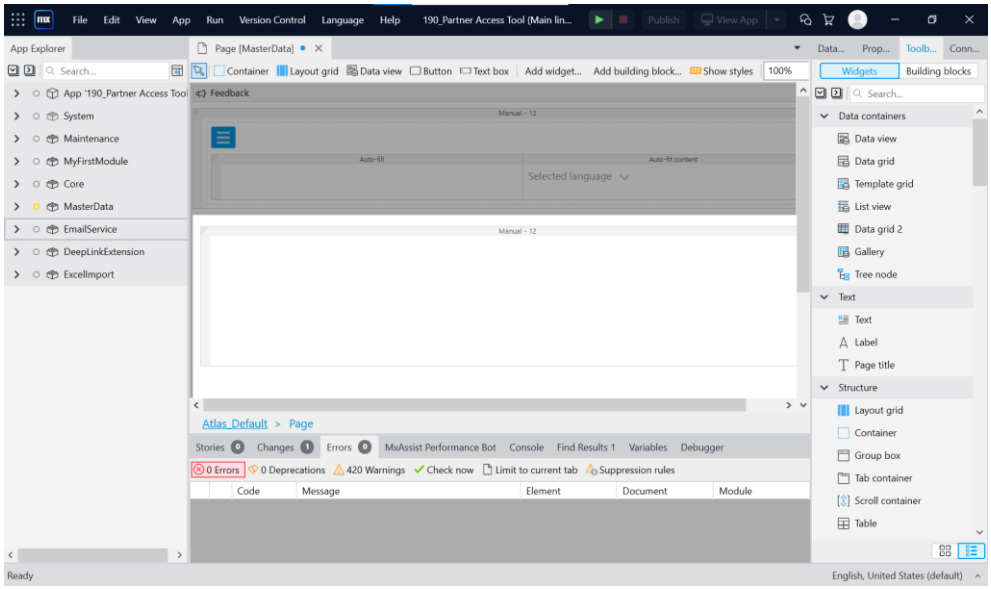


Figura 12. Ambiente de desenvolvimento Mendix Studio Pro

2.6.4 COMPARAÇÃO ENTRE AS PLATAFORMAS

Uma comparação das plataformas descritas anteriormente é feita nas Tabela 1 e Tabela 2, de acordo com o SourceForge [29], [30]. Esta sistematização permite ao usuário escolher uma plataforma de acordo com as funcionalidades presentes em cada uma delas.

A partir da tabela é possível perceber que entre as três plataformas descritas, duas são muito semelhantes nas suas abordagens e muito mais abrangentes num contexto geral de mercado. Como descrito anteriormente, PegaSystems é uma plataforma para mineração de processos, não tendo tanta facilidade de se adaptar às necessidades de alguns projetos. Por outro lado, Outsystems e Mendix possuem abordagens muito parecidas e adequam-se melhor às

adversidades dos projetos, dando maior amplitude e escalabilidade para o desenvolvimento de aplicações por parte dos desenvolvedores e usuários.

Tabela 1 Comparação de funcionalidades entre as Plataformas – Categorias e Recursos de desenvolvimento de aplicativo (adaptado de [29], [30])

Funcionalidades	Mendix	Outsystems	PegaSystems
Categorias			
App Makers	✓	✓	×
Business Process Management	×	×	✓
Application Development	✓	✓	✓
Digital Experience Platforms (DXP)	×	✓	×
Low-code Development	✓	✓	✓
No-Code Development	✓	✓	×
Rapid Application Development (RAD)	✓	✓	×
Web App Development	✓	✓	×
Recursos de desenvolvimento de aplicativo			
Access Controls/Permissions	✓	✓	✓
Code Assistance	✓	✓	×
Code Refactoring	✓	✓	✓
Collaboration Tools	✓	✓	✓
Compatibility Testing	✓	✓	✓
Data Modeling	✓	✓	✓
Debugging	✓	✓	×
Deployment Management	✓	✓	✓
Graphical User Interface	✓	✓	×
Mobile Development	✓	✓	×
No-Code	✓	×	✓
Reporting/Analytics	✓	✓	✓
Software Development	✓	✓	×
Source Control	✓	✓	✓
Testing Management	✓	✓	×
Version Control	✓	✓	×
Web App Development	✓	✓	×

Tabela 2 Comparação de funcionalidades entre as Plataformas – Recursos de desenvolvimento de *low-code* e Integrações (adaptado de [29], [30])

Funcionalidades	Mendix	Outsystems	PegaSystems
Recursos de desenvolvimento de *low-code*			
AI-Assisted Development	✓	×	×
Business Process Automation	✓	✓	×
Collaborative Development	✓	✓	×
Data Aggregation and Publishing	✓	✓	×
Deployment Management	✓	✓	×
Drag & Drop	✓	✓	×
Integrations Management	✓	×	×
Iteration Management	✓	✓	×
Performance Monitoring	✓	✓	×
Requirements Management	✓	✓	×
Templates	✓	✓	×
Visual Modeling	✓	✓	×
Web / Mobile App Development	✓	✓	×
Workflow Management	✓	✓	×
Integrações			
Amazon DynamoDB	✓	×	×
Azure SQL Database	✓	×	×
Docmosis	✓	×	×
Facebook	×	✓	×
Google	✓	×	×
Google Cloud Platform	✓	×	×
LinkedIn	×	✓	×
Meeting360	×	✓	×
Microsoft 365	✓	×	×
Microsoft Word	✓	×	×
MuleSoft Anypoint Platform	✓	×	×
NetSuite	×	✓	×
Redshift	✓	×	×
SAP Cloud Platform	✓	×	×
Salesforce	✓	✓	×
Shift	×	✓	×
Sitecore	✓	×	×
Slack	✓	×	×
Tableau	✓	×	×
Workato	✓	×	×

A partir da Tabela 1 e Tabela 2 é possível perceber algumas diferenças entre as funcionalidades diferenciadas por cada plataforma, tais como:

- PegaSystems não permite recursos de desenvolvimento *low-code* dentro da plataforma e das aplicações e não possui integração com recursos externos que ajudem na parte de partilha e armazenamento.
- Outystems possui a plataforma de experiência digital (*Digital Experience Plataforms*, DXP) que é um conjunto integrado de tecnologias e processos que dão suporte à composição, gestão, entrega e otimização das experiências do cliente, facilitando iteração da aplicação com o cliente final.
- Mendix possui uma ferramenta IA dando assistência de desenvolvimento a plataforma, na qual ainda é capaz de verificar possíveis erros no decorrer da criação de uma lógica e até mesmo indicar onde se encontra o erro, bem como também permite uma gestão de integração com plataformas externas, aumentando as possibilidades de desenvolvimento e implantações de possíveis softwares.

A comparação de funcionalidades permite identificar as plataformas que cada uma oferece aos utilizadores finais. No entanto, a escolha da plataforma de desenvolvimento ao nível de uma organização é também influenciada por outros fatores, como a dimensão da empresa, as soluções possíveis que cada plataforma apresenta para o contexto requerido, os custos das plataformas e a dimensão do projeto. Sendo assim, os fatores citados necessitam de uma análise antes de se tomar a decisão de usar ou não deste tipo de plataforma, sendo ainda um dos principais pontos que pesam na hora de escolher as plataformas *low-code*.

No próximo capítulo iremos abordar as metodologias de gestão do desenvolvimento de software, uma parte fundamental para entender como os projetos de software são planejados, executados e entregues com sucesso. Desta forma, são oferecidas estruturas e abordagens para gerenciar o ciclo de vida de um projeto de software, desde a conceção até a entrega. Algumas metodologias mais comuns incluem modelo em cascata (*Waterfall*), modelo espiral, desenvolvimento rápido de aplicações (*Rapid Application Development*, RAD), Kanban e Scrum.

Cada metodologia tem seus próprios princípios, práticas e técnicas específicas que adequam-se a diferentes tipos de projetos, equipas e requisitos. Elas também podem influenciar a forma como as equipas colaboram, como as tarefas são planejadas e executadas, bem como os resultados são entregues aos clientes ou utilizadores finais.

3

METODOLOGIAS DE GESTÃO DO DESENVOLVIMENTO DE SOFTWARE

Neste capítulo serão apresentadas breves descrições e enquadramentos históricos de algumas das principais metodologias de gestão de desenvolvimento de software. O enquadramento tem como objetivo apresentar não só vantagens de cada um dos modelos, mas também as suas principais limitações que incitaram o surgimento de novos modelos e perspetivas.

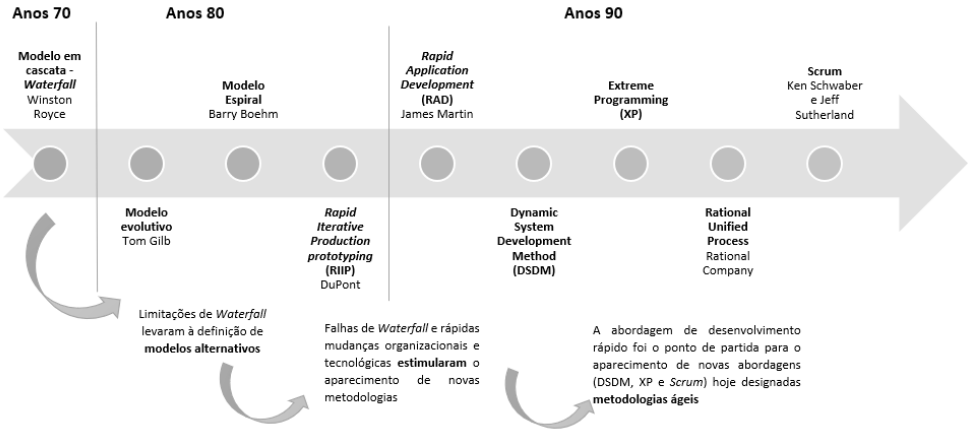


Figura 13. Evolução das metodologias de desenvolvimento de software

A Figura 13 apresenta uma evolução das metodologias que iremos detalhar no restante capítulo. O enquadramento inicia-se com a descrição do modelo em cascata, sendo esse modelo considerado uma metodologia tradicional de gestão de desenvolvimento de software. Em seguida temos a descrição do modelo espiral que surgiu como alternativa para solucionar as falhas do modelo em cascata. Em seguida, descreve-se a metodologia RAD. O aparecimento desta abordagem contribuiu para a evolução de novos métodos conhecidos, atualmente, como

metodologias ágeis. Desta maneira, pretende-se realizar uma breve comparação entre as metodologias que surgiram depois do RAD.

3.1 MODELO EM CASCATA

O modelo conhecido popularmente por cascata (do inglês *Waterfall*), foi proposto na década de 1970 por Winston Royce no artigo "*Managin the development of large software systems*" [31]. Esse modelo é considerado como um ponto de vista clássico do ciclo de vida de um desenvolvimento de software, sendo estruturado, sequencial e direcionado para compor relatórios de todas as decisões tomadas e atividades realizadas ao longo do processo de desenvolvimento.

Neste modelo, o desenvolvimento de um software inicia-se a partir de uma definição detalhada dos requisitos de sistema, tendo em conta as necessidades do cliente e as funcionalidades que o software pode realizar. Com isso, cada etapa do ciclo de vida de desenvolvimento gera resultados e só se inicia o passo seguinte se o atual estiver concluído.

A Figura 14 apresenta o modelo em cascata, esquematizando o fluxo e as interações entre as etapas, nomeadamente.

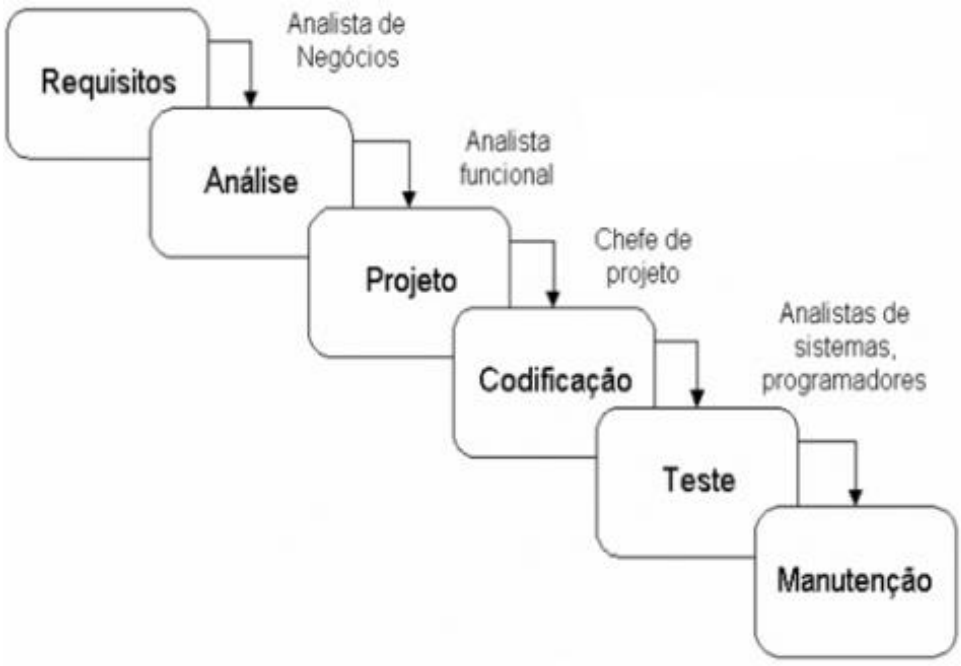


Figura 14. Modelo em Cascata (Adaptado de [31])

- **Requisitos:** Etapa onde são definidos os componentes para o desenvolvimento do projeto, tais como ferramentas de software e requisitos de hardware. Também

define os requisitos e as interações necessárias com outras aplicações e base de dados;

- **Análise:** Esta etapa é feita a transposição das informações levantadas na etapa anterior para uma modelagem do ambiente do usuário, através de diagramas de fluxo de dados, entidade-relacionamento, transições de estados, classes e outras ferramentas;
- **Projeto:** Nesta etapa só é iniciada quando as anteriores terminam, sendo direcionado a 4 atributos distintos do programa: estrutura de dados, arquitetura do software, procedimentos e caracterização da interface;
- **Codificação:** A etapa da codificação está diretamente ligada a programação e envolve a geração de linhas de código para as funcionalidades;
- **Testes:** Nesta etapa é feita a verificação de eventuais erros no código, assim como a validação do cumprimento das especificações definidas na solução;
- **Manutenção:** Etapa ocorre depois da entrega da solução e tem como objetivo o tratamento de problemas e pedidos de melhoria na solução apresentada.

Na prática, atualmente, essa abordagem raramente é seguida por programadores por ser extremamente rígido, e pela dificuldade de entender ainda no início os requisitos do projeto, em alguns casos desconhecidos pelos próprios clientes.

Sendo assim, esse modelo tende a ser mais utilizado em projetos cujos requisitos do software são bem conhecidos pelos clientes e envolvendo processos "maduros".

3.1.1 CARACTERÍSTICAS DO MODELO

É importante observar que o modelo em cascata possui como principal característica a sua fluidez e a correlação bem definida entre as etapas de desenvolvimento. Cada uma das etapas desse método funciona somente após a conclusão da anterior, fazendo com que o processo de desenvolvimento seja mais demorado, devido a interligação entre as etapas.

Outra característica do método cascata é a sua rigidez em cada uma das etapas, ou seja, é quase impossível que erros ocorram durante o processo de desenvolvimento de software, devido a ideia de que uma etapa necessita estar concluída para iniciar outra. Nesta abordagem, erros são facilmente detetados. Caso ocorra algum erro, é feito um processo de retorno das etapas até o princípio, não sendo possível resolver os problemas "por partes".

Além disso, é caracterizado ainda por ser um método em que o cliente e o desenvolvedor se completam de forma mútua, dado que o progresso do desenvolvimento depende de ambas as partes para que sejam concluídas as etapas do processo. Por outro lado, o cliente é responsável

por desenvolver a ideia, expondo as suas necessidades e expectativas para o projeto, e o desenvolvedor criará o software. Um orienta e o outro executa. Com isso, as partes se tornam interdependentes, e se uma das partes deixar de considerar um elemento do projeto, o projeto recomeça do absoluto zero.

A metodologia cascata é fluida, sendo expressa em etapas, com durações mais longas e com correlação em seus projetos, tanto no relacionamento entre suas etapas quanto no desenvolvimento entre cliente e os desenvolvedores.

3.2 MODELO ESPIRAL

As limitações apresentadas na Secção 3.1 para o Modelo em Cascata levam à criação de modelos alternativos, como por exemplo o Modelo em Espiral. Criado por Barry Boehm em 1988 [23,24], o Modelo em Espiral é uma evolução do Modelo em Cascata e de outros modelos alternativos criados anteriormente. O nome do modelo em questão deriva da sua representação (ver Figura 15), onde cada volta de uma espiral se tende a percorrer todas as fases do processo de desenvolvimento de software.

Esse modelo apresenta um comportamento orientado para o risco – *risk-driven* – tendo como principal objetivo a análise das alternativas e a identificação de objetivos. O ciclo de vida deste modelo apresenta 4 fases, sendo que as voltas necessitam ser repetidas quantas vezes forem necessárias para que o software chegue na etapa de poder ser completamente entregue. Deste modo, o processo deste modelo é evolucionário, sendo ideal para softwares que necessitam passar por diversos melhoramentos ao longo do seu desenvolvimento.

A Figura 15 apresenta um esquema do Modelo em Espiral, contendo as já referidas 4 etapas, nomeadamente:

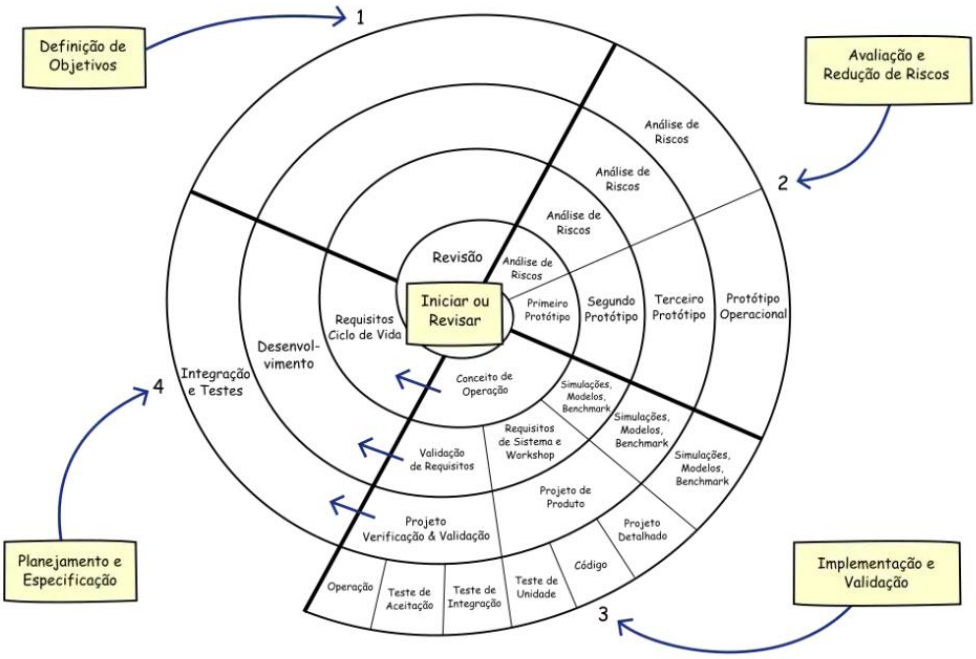


Figura 15. Modelo em Espiral com o processo completo (Fonte: [33])

- **Definição de objetivos:** Etapa onde são definidos os objetivos do produto a ser implementado, meios de implementação e as suas respetivas limitações, em relação ao orçamento definido;
- **Avaliação e redução de riscos:** Nesta etapa identificam-se os riscos e para cada um destes riscos é feita uma "análise de risco" detalhada, a fim de traçar estratégias para reduzi-lo ou evitá-lo. Em casos em que exista alguma dificuldade em especificar ou detalhar claramente um requisito, isso significa que existe um "risco de requisitos inadequados" e, como solução, é feito um protótipo para apresentar ao cliente como sugestão e contribuir para refinar os requisitos;
- **Implementação e validação:** Nesta etapa é feita a escolha do modelo de desenvolvimento, podendo até ser usado o modelo em cascata citado anteriormente ou algum outro que se adeque melhor às necessidades. Também é possível usar diferentes modelos em cada iteração da implementação, de acordo com as necessidades;
- **Planejamento e especificação:** Por fim, na última etapa, é feita uma análise do projeto como um todo, de modo a avaliar o trabalho realizado e para que possa ser feito um planejamento de onde será o próximo passo para iniciar uma nova espiral ou completar por fim o sistema.

Entre as principais vantagens, o modelo apresenta uma documentação sólida, minimizando riscos e eliminando erros. Tem como foco para a reutilização de software e estimula o estudo de novos aprimoramentos, ao que tende a ser um modelo mais realista a partir das estimativas de cada uma das suas etapas, reduzindo o tempo da implementação do software, e não faz distinção entre desenvolvimento e manutenção.

Por outro lado, este modelo possui limitações como sejam: os custos com especialistas em análises de risco e dependência da análise desses especialistas. Dado a dependência desta análise, fatalmente erros podem ocorrer, caso um risco não seja identificado e gerido de forma correta. Assim, o erro na avaliação impacta diretamente o projeto.

3.2.1 CARACTERÍSTICAS DO MODELO

O que diferencia este modelo dos demais modelos de processamento de software, é a sua capacidade de reconhecer explicitamente os riscos, o que pode reduzir consideravelmente a possibilidade de falhas em grandes projetos, pois faz a avaliação dos riscos repetidamente e a verificação do produto em desenvolvimento em cada volta da espiral.

Na realidade o modelo em espiral é um aglomerado de componentes dos demais modelos de ciclo de vida de software, como por exemplo o modelo em cascata e dos demais modelos que surgiram antes do modelo em espiral.

As suas características permitem-lhe lidar com qualquer tipo de adversidade que possa ocorrer ao longo do processo de construção de software, melhorando os demais modelos anteriores. No entanto, por possuir tantos elementos, faz com que esse modelo seja muito mais complexo do que os seus modelos antecessores de desenvolvimento de software.

3.3 DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES

A expressão RAD aplica-se em sua grande maioria a projetos que possuem prazos curtos, e geralmente envolvem o uso de protótipos e ferramentas de desenvolvimento de alto desempenho.

Essa metodologia é definida como “uma abordagem para construir sistemas de computador que combina ferramentas e técnicas CASE, protótipos dirigidos ao usuário e prazos curtos de forma eficiente, testada e confiável para alta qualidade e produtividade” [34].

A metodologia RAD contribui para que as organizações desenvolvam mais rapidamente softwares a custos reduzidos no desenvolvimento e mantendo a qualidade. Essa redução de custos e qualidade devem-se à uma série de técnicas de desenvolvimento de aplicações comprovadas, agregada a uma metodologia estrategicamente definida. As falhas nos modelos

anteriores, juntamente com as rápidas mudanças organizacionais e tecnológicas, foram estímulos para a definição de RAD.

Figura 16. Relação dos principais objetivos do método RAD (Fonte: [35])

A Figura 16 ilustra a relação entre os objetivos utilizados durante o processo de implementação do método. Tendo em conta os objetivos e as relações entre os mesmos, é possível observar que se torna uma relação extremamente vantajosa para ambas as partes, devido ao resultado produzido desde a conceção do projeto até o software final, considerando os custos e a qualidade para as empresas e a utilização de ferramentas automatizadas com maior resposta às suas necessidades. O sistema desenvolvido juntamente com essa metodologia consegue satisfazer melhor as necessidades dos utilizadores e com menores custos de manutenção.

Para um desenvolvimento rápido, é necessária a participação de pessoas qualificadas e bem orientadas, juntamente com uma gestão adequada para evitar que o processo burocrático atrase o desenvolvimento do sistema. A metodologia RAD condensa o desenvolvimento incremental dos métodos tradicionais para um processo bem mais colaborativo entre os donos do projeto

e desenvolvedores. Com isso, a sua abordagem envolve desenvolver e otimizar os modelos de dados, processos e protótipos em paralelo através de um processo iterativo. Desta forma, os requisitos do usuário são detalhados, para a solução é feito um protótipo, o protótipo é alterado, é concedido novamente o acesso ao usuário e o processo começa novamente [34].

a Figura 17 esquematiza o desenvolvimento tradicional de software, frequentemente associado ao modelo em cascata, como apresentado na Secção 3.1, o qual as etapas do projeto são executadas em uma ordem fixa e linear. Isso significa que uma fase deve ser concluída antes que a próxima comece, e as mudanças significativas geralmente são difíceis e caras de serem implementadas após o início do desenvolvimento.

Por outro lado, o RAD é uma metodologia que segue uma abordagem mais iterativa e colaborativa. O foco está na entrega rápida de um protótipo funcional e iterativo do software, o que permite a rápida identificação de requisitos e mudanças, resultando em uma maior flexibilidade para acomodar ajustes no decorrer do projeto.

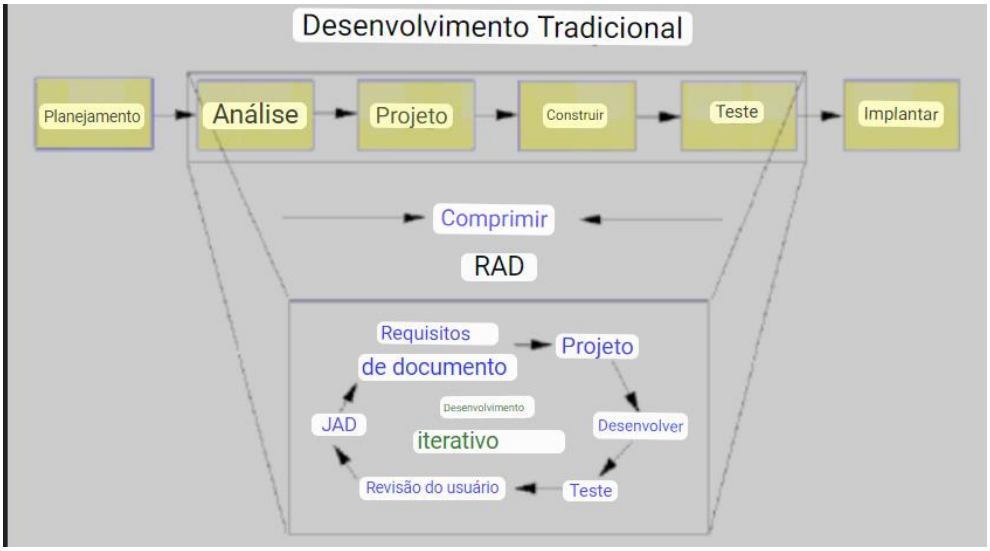


Figura 17. Otimização modelo de dados, processos e protótipos RAD (Adaptado de [36])

3.3.1 CARACTERÍSTICAS DO MODELO

Devido ao seu processo de desenvolvimento iterativo e flexível, a metodologia RAD possui algumas características dentro das equipas que a utilizam (e.g., reutilização de módulos), além da melhora no tempo de desenvolvimento, consegue reduzir as chances de deteção de *bugs* na aplicação, devido aos testes previamente realizados.

Alguns outros pontos característicos desse modelo são:

- Garantia de *feedback* rápido e constante;
- Maior flexibilidade no tocante a mudanças de necessidades;

- Reduz a codificação;
- O progresso é facilmente mensurado;
- Equipas que desenvolvem mais em menos tempo;
- Garantia de um ciclo de desenvolvimento curto, variando entre 60 e 90 dias.

Um ponto desse modelo é que a aplicação deve possuir requisitos muito bem estruturados e o sistema possuir a possibilidade de ser modularizado para o melhor funcionamento do RAD. Quando é feita uma análise do projeto a ser desenvolvido, e o mesmo apresenta a possibilidade de ser dividido em componentes e em poder reutilizar componentes prontos, sempre visando o curto prazo, é passível de ser escolhido o RAD como uma das melhores opções.

Esse método não é normalmente sugerido quando se possui um risco técnico alto ou quando não se tem as equipas apropriadas para suprir a agilidade que o método requer.

Com isso, é possível perceber que a metodologia RAD é extremamente interessante para quem busca a criação de aplicações de forma rápida, mas sem perder qualidade final do projeto [37].

3.4 KANBAN

A origem deste método deu-se pela Toyota como forma de melhorar o processo de fabricação de veículos [38], mas com o passar do tempo começou a ocupar outras áreas, dentre elas a de desenvolvimento de software Ágil.

Kaban é uma metodologia popular de gestão de fluxo de trabalho para definir, gerenciar e melhorar os serviços fornecidos. Utiliza um método mais visual para ilustrar as etapas do trabalho, maximizar e melhorar de forma continua. É ilustrado através de uma estrutura de quadro, onde cada uma das suas colunas representa uma etapa de elaboração. A isso deu-se o nome de quadro Kanban e, através desse quadro, é possível otimizar a entrega dos trabalhos em várias vertentes em simultâneo. Além disso, também é capaz de lidar com projetos mais complexos num único ambiente [38].

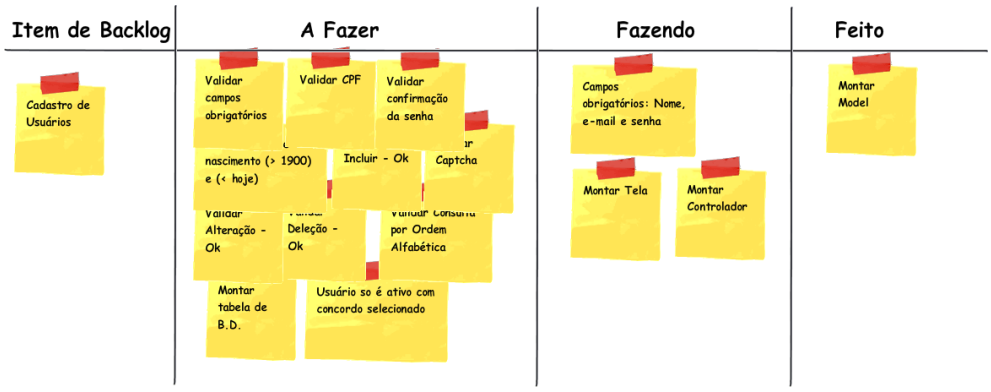


Figura 18. Quadro Kanban (Fonte: [39])

A Figura 18 representa a orientação das tarefas do quadro Kanban. Na coluna história de usuário (*Item do Backlog*) é subdividida em várias tarefas menores, que são adicionadas à coluna "A fazer". Um membro da equipa seleciona uma dessas tarefas, move-a para a coluna "Fazendo". Após a implementação e a execução dos testes de integração apropriados, quando a tarefa é considerada concluída, ela é então movida para a coluna "Feito". Posteriormente, o membro da equipa seleciona outra tarefa na coluna "A fazer" e inicia o ciclo novamente. Esse processo se repete até que a equipa tenha movido todas as tarefas para a coluna "Concluído".
A metodologia Kanban acabou por se tornar parte do grupo de metodologias ágeis, sendo que esta forma de trabalho ajuda as equipas a perceberem o que devem fazer e, assim, atingirem suas metas com maior facilidade e agilidade. Por outras palavras, as equipas possuem alta integração, trocando dados e solucionando problemas de maneira colaborativa.
A metodologia Kanban aplica-se, por exemplo, em processos de correção de *bugs*, atualização e falhas de segurança de um software. Tendo a necessidade de uma equipa responsável pela manutenção do código-fonte, solucionando os problemas de forma ágil e de acordo com a prioridade determinada para cada falha que possa impactar no resultado para o usuário. Os desenvolvedores responsáveis pelo processo de correção de *bugs* possuem um ambiente integrado com o intuito de solucionar dúvidas, identificar a origem do problema e traçar a melhor estratégia para corrigir falhas.

3.4.1 CARACTERÍSTICAS DO MODELO

Este modelo tem como objetivo otimizar a rotina de produção de aplicações, tendo como resultado agilizar a entrega do produto final. A maneira como esta metodologia é estruturada facilita a sua integração na rotina de cada equipa, tornando-se mais simples e prática de ser utilizada no processo de desenvolvimento de software [40].
Para fazer o uso do Kanban, é necessário seguir um conjunto de princípios básicos, sendo:

- **Priorizar rotinas:** A priorização é uma das bases da metodologia Kanban, ajudando a garantir a orientação de qual tarefa deve ser feita e a sua prioridade, ajudando a obter melhores resultados. Deste modo torna-se mais fácil definir o que deve ser executado primeiro e quais são os objetivos subsequentes.
- **Entender a necessidade de evoluir:** Cada etapa é necessária uma análise por parte do time para entender os pontos que necessitam de melhorias e que possam ser replicados noutras áreas. Desta maneira, o nível de qualidade no ambiente de trabalho tende a ser o melhor possível.

- **Respeitar a hierarquia interna:** O papel de cada membro da equipa deve ser respeitado, garantindo que cada um saiba a sua contribuição no desenvolvimento e metas no projeto. É importante que cada profissional saiba a quem deve se dirigir em caso de problemas ou dúvidas, para garantir maior qualidade ao projeto.
- **Incentivar a liderança:** A liderança é essencial para incentivar os profissionais a terem melhores resultados, pois é responsável por definir as prioridades das tarefas, incentivar as equipas e garantir a comunicação clara entre todos os envolvidos no desenvolvimento. Com isso, é importante que os profissionais sejam grandes líderes, que os gestores mantenham uma comunicação única, clara e objetiva, pois assim as equipas estarão engajadas, unidas e direcionadas para o mesmo objetivo.
- **Incentivar a autonomia dos profissionais:** A empresa deve garantir que seus profissionais têm a liberdade e a capacidade de executar as tarefas da melhor forma possível e tomar decisões quando for necessário. Com essa abordagem torna-se mais rápida a resolução de um problema e uma tomada de decisão, sem necessitar de aprovação em processos burocráticos em alguns casos.

A metodologia Kanban tende a ser visto como uma forma de trabalho contínuo e, com isso, tem a capacidade de ser aprimorada para reduzir ao máximo o número de erros, fazendo com que as atividades sejam mais fluidas. Com isso, as empresas podem alcançar os resultados esperados e atuar de maneira estratégica com soluções para clientes que possuam ferramentas inteligentes, inovadoras e desafiadoras [38].

3.5 SCRUM

A metodologia Scrum surgiu nos meados dos anos 90, sendo desenvolvida por Ken Schwaber e Jeff Sutherland [41]. Scrum é uma metodologia de desenvolvimento ágil utilizada em desenvolvimento de software, tendo como princípio processos iterativos e incrementais. Esta metodologia é uma estrutura ágil de fácil adaptação, rápida, flexível e eficaz, projetada para agregar valor ao cliente durante todo o desenvolvimento do projeto.

Scrum tem como principal objetivo a satisfação do cliente através de um ambiente de transparência na comunicação, responsabilidade coletiva e progresso contínuo. O princípio do desenvolvimento parte de uma ideia geral da necessidade do cliente, sendo feita uma lista de características em ordem de prioridade (*Product backlog*) que o cliente deseja alcançar como objetivo final [41].

Essa abordagem surgiu como uma necessidade de uma nova alternativa perante as metodologias tradicionais, como modelo em cascata, que conforme descrito anteriormente,

aplica-se bem em projetos muito bem definidos, que possuem previsibilidade e com pouca possibilidade de alteração ao longo do processo. Entretanto, no atual contexto, a maior parte dos projetos não são previsíveis, principalmente na sua fase inicial e também pelo facto de que as metodologias tradicionais não suportam bem as possíveis mudanças ao longo do desenvolvimento.

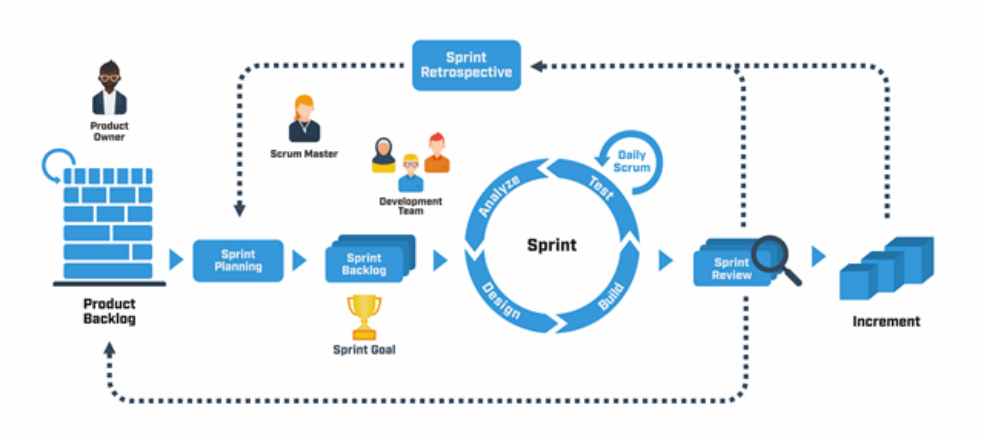


Figura 19. Ilustração gráfica da metodologia Scrum (Fonte: [42])

A Figura 19 representa a estrutura da metodologia Scrum, suas etapas e devidas orientações sequenciais das mesmas [41], nomeadamente.

- ***Scrum Team*:** é uma pequena equipa composta por um *Scrum Master*, equipa de desenvolvedores e uma equipa de *Product Owner*. O *Scrum Master* é o responsável por organizar e fazer com que sejam cumpridos os princípios do Scrum e contribui para que possíveis impedimentos no desenvolvimento sejam devidamente esclarecidos e resolvidos. Os desenvolvedores são os responsáveis pelo cumprimento das tarefas que foram atribuídas e também por fazer o *Sprint Backlog* onde são atribuídas as tarefas definidas para aquele sprint. O *Product Owner* é o responsável pela gestão do *Backlog*, pela criação das histórias de usuários (*User Stories*) e por dar *feedback* em relação ao progresso no desenvolvimento baseado nas descrições e objetivo final do projeto;
- ***Scrum Events*:** são os eventos realizados durante o desenvolvimento do projeto, desde o princípio até o fim, sendo:
  - ***Sprint*:** representa o evento de duração fixa determinado para cada tipo de projeto, onde são realizadas diversas atividades que como: *Sprint Planning*, *Daily Scrum*, *Sprint Review*,*Backlog Refinement* e *Sprint Retrospective*;

- ***Sprint Planning*:** evento feito para determinar o início do *Sprint*, ou seja, o início do desenvolvimento e também determina quais são as tarefas que devem ser feitas naquela semana;
- ***Daily Scrum*:** evento que ocorre diariamente no mesmo horário durante o *Sprint*, geralmente com duração de 15 a 30 minutos. O objetivo deste evento é o acompanhamento do desenvolvimento e do cumprimento dos requisitos. Em alguns casos até são feitos os esclarecimentos de dúvidas;
- ***Sprint Review*:** evento que ocorre sempre no final de um *Sprint*, com o objetivo de rever os resultados alcançados de acordo com as tarefas determinadas e onde se determina o que será feito em seguida;
- ***Backlog Refinement*:** evento que ocorre para que todos possam discutir as dúvidas e esclarecer determinados pontos dos objetivos determinados para o *Sprint*;
- ***Sprint Retrospective*:** evento que tem como objetivo realizar uma análise do último *Sprint* e para planear abordagens distintas de forma a aumentar a qualidade e eficácia do desenvolvimento e do projeto;

- ***Product Backlog*:** representa uma lista em ordem de prioridade de tarefas a serem realizadas, podendo ser requisitos funcionais ou não funcionais ou *bugs*;
- ***Sprint Backlog*:** é composto pelo objetivo, pelo conjunto de tarefas a serem feitas e pelo plano que será seguido no *Sprint*.

Atualmente essa metodologia ágil é a mais usada pelos desenvolvedores. Esta preferência ocorre devido à capacidade de responder rápido às necessidades do mercado, apresentando soluções em ciclos de desenvolvimento mais curtos, o que permite uma diminuição nos custos [41].

3.5.1 CARACTERÍSTICAS DO MODELO

Esta metodologia ágil tem como foco a dinâmica entre desenvolvedores e os clientes, através de uma intermediação e atividades diárias. O Scrum consegue alcançar resultados de forma rápida e com maior exatidão devido a sua ideologia e suas características voltadas para um acompanhamento constante do desenvolvimento, o que também permite uma maior flexibilidade no projeto para adaptações que possam surgir.

Algumas das características desta metodologia são:

- o método permite que os membros da equipa trabalhem e desenvolvam o projeto em conjunto, compartilhando as ideias e soluções descobertas;

- os desenvolvedores são capazes de gerir suas atividades e todas se enquadram dentro de um tempo determinado;
- os clientes recebem versões contínuas do projeto a cada nova etapa concluída, onde já se é possível testar o que foi feito;
- a equipa se concentra sempre em dar respostas rápidas e eficientes ao desenvolvimento do software, tanto os donos do projeto quanto os desenvolvedores;
- o tamanho da equipa também é um fator decisivo, sendo um tamanho ideal de 6 a 10 pessoas.

Para além disso, o Scrum tem implementação incremental e interativa, facilitando a adaptação às diversidades, tendo em conta a participação dos clientes durante todo o processo, o que gera uma facilidade no lado dos clientes em perceber se a ideia inicial faz sentido ou se necessita de alteração, aumentando a satisfação no momento da entrega. Com equipas tecnicamente experientes torna-se extremamente atrativo o uso do modelo nos projetos de desenvolvimento de software, obtêm-se melhores resultados e devido a agilidade, faz com que seja possível fazer mais em menos tempo e avançar para novos projetos [41].

3.6 COMPARAÇÃO ENTRE METODOLOGIAS

A necessidade de uma metodologia eficiente para projetos cada vez mais detalhados e elaborados, vem da exigência de seguir corretamente determinadas práticas como prazos, custos, qualidade, inovação e satisfação do cliente. Essas normas contribuem para um usuário final com uma experiência única. À medida que as necessidades do mercado aumentaram, o gerenciamento dos projetos se tornou prioridade máxima e, com isso, o uso de uma metodologia adequada tornou-se uma necessidade.

Diante das necessidades, tivemos o surgimento das metodologias tradicional e ágil, conforme citado anteriormente, sendo notórias as diferenças entre elas. Na abordagem da metodologia tradicional, apenas é feita uma entrega e o projeto está totalmente pronto. Porém, no caso da metodologia ágil, a entrega é feita continuamente e por partes, até a entrega final do projeto.

A flexibilidade é uma característica importante a ser observada. A metodologia tradicional tem pouca flexibilidade, seguindo o projeto inicial, enquanto na metodologia ágil é flexível o suficiente para fazer alterações ao longo do desenvolvimento, se necessário. Os métodos ágeis são mais iterativos e práticos, enquanto os métodos tradicionais planejam com antecedência cada fase de um projeto.

No pensamento tradicional, o líder é responsável por todo o processo, além de delegar as atividades entre as partes da equipa. No ágil, existe uma divisão da equipa com o objetivo de que todos possam tomar determinadas decisões de forma autónoma.

Outra diferença entre as abordagens é a definição do orçamento. No tradicional, o orçamento é definido para um longo período, enquanto na abordagem ágil o foco é o gerenciamento dos custos do projeto, que são definidos apenas conforme cada fase de desenvolvimento é concluída.

Os métodos tradicionais documentam o plano e definem a sequência de desenvolvimento, enquanto no método ágil segue a ideia de que quanto mais funcional o software, melhor é, e se adaptam ao projeto para implementar tudo de maneira mais eficiente.

3.6.1 COMO DEFINIR QUAL METODOLOGIA UTILIZAR?

Essa questão não possui uma resposta exata, pois vai dependente de fatores como [43]:

- Tipo de projeto;
- Cultura da empresa.

A própria empresa deve definir o que melhor se adequa a sua necessidade. Existem empresas que preferem as metodologias tradicionais, devido a falta de conhecimento das pessoas envolvidas nos projetos a respeito das metodologias ágeis, mesmo sendo mais indicadas para sua execução.

O indicado sempre é fazer uma pesquisa sobre o projeto, conhecer os requisitos e a tecnologia que será utilizada. A partir desses detalhes, fazer uma análise se é melhor começar com uma abordagem ágil ou tradicional. Em projetos em que os requisitos do cliente podem mudar a qualquer momento, torna-se necessário ter liberdade para fazer alterações no software em desenvolvimento, não só do lado do cliente, mas também do lado técnico do projeto. Projetos com potenciais mudanças exigem flexibilidade no planejamento, com isso, a abordagem mais indicada seria a ágil.

A metodologia tradicional é mais indicada em casos mais específicos, onde tem-se a necessidade de ser planejado e decidido ainda no início. Um projeto com poucas chances de ter mudanças e com menor risco, o ideal é ter um plano bem definido antes mesmo de iniciar.

A escolha da metodologia é importante não só para o sucesso do processo, mas também para a entrega do produto final. Ambos os métodos possuem vantagens particulares, mas possibilitam que sejam usados em conjunto de forma harmoniosa, pois tem como principal objetivo a otimização do projeto. O importante é respeitar os pressupostos de ambas abordagens e saber que cada um pode acrescentar valor aos objetivos de cada projeto.

4

CASOS DESENVOLVIDOS

Este capítulo apresenta um conjunto de casos desenvolvidos no âmbito do presente estágio, bem como a justificação dos aspetos considerados essenciais na sua definição.

No referido âmbito, o desenvolvimento de soluções através de plataformas *low-code* possibilitou a definição de dois casos elaborados de forma a demonstrar o domínio profissional, com a finalidade de englobar os projetos feitos e a utilização da plataforma de desenvolvimento Mendix mostrada no presente relatório. Para além disso, considerou-se o desenvolvimento de uma solução para gestão de atividades de empresa para empresa (*Business to Business*, B2B) e uma solução para atender aos requisitos pedidos pelo cliente. A consideração destes dois tipos de abordagens pretende apresentar um domínio aplicacional mais dinâmico, permitindo simplificar o entendimento das necessidades.

Neste cenário, cada caso será apresentado subcapítulos distintos, sendo a estrutura da apresentação: (i) uma descrição breve do domínio do negócio; (ii) requisitos do negócio considerados relevantes; e (iii) arquitetura da solução em repostas as necessidades e aos requisitos. Referente à definição da arquitetura, em cada um dos casos de estudo, será apresentada uma arquitetura diferente, de forma a atender o caso em questão. A construção das soluções, em comum acordo com o desenho definido, será realizada recorrendo a plataforma de desenvolvimento *low-code* em análise – Mendix.

O conceito destes casos de estudo foi baseado nos principais desafios no desenvolvimento de aplicações, que acabam por ser responsáveis por atrasar a entrega, independente da tecnologia a ser escolhida: (i) dificuldade em atender as necessidades do negócio em tempo útil, (ii) inflexibilidade, (iii) dificuldade em reduzir custos, (iv) necessidade de manter as aplicações seguras, (v) dificuldade em desenvolvimento UX/UI e (vi) coordenação da implantação.

Sendo assim, de forma a ser possível avaliar de maneira crítica as características da referida plataforma de desenvolvimento, os casos de estudo tiveram em consideração as características dessa plataforma em específico.

4.1 CASO DE ESTUDO 1

Nesta secção é apresentada a descrição do campo de negócios deste Caso de Estudo e a descrição dos principais aspetos de sua implementação. No âmbito do presente Caso de Estudo, para além dos propósitos relacionados à definição de arquitetura, pretende-se que seja possível construir e analisar soluções. O seu desenvolvimento deve ser essencialmente focado nos detalhes de modelagem de dados, lógica de negócios e interface do sistema. Portanto, pode-se analisar os recursos de resposta da plataforma em destaque no presente relatório.

4.1.1 CARACTERIZAÇÃO DO NEGÓCIO

A empresa multinacional no ramo da engenharia (que por motivos de confidencialidade não é possível identificar neste relatório, como já referido), é de origem e sede alemã. Atuando em diversos setores da tecnologia, oferece soluções para infraestrutura, energia, automação, digitalização e mobilidade. Conta com ampla gama de produtos e serviços, desde equipamentos de diagnóstico médico até turbinas eólicas, passando por sistemas de automação industrial, software de simulação e controle de processos, sistemas de transporte ferroviário e soluções de energia descentralizada para edifícios e comunidades. A entidade tem uma presença global significativa e trabalha em estreita colaboração com clientes e parceiros para ajudá-los a alcançar seus objetivos de negócios de forma mais eficiente, sustentável e inovadora.

Com isso, a empresa identificou uma necessidade de processos mais rápidos e integrados num dos seus setores. Para satisfazer essa necessidade, foram adotadas diversas estratégias, como a implementação de novas tecnologias e a reorganização dos processos existentes. A partir desta necessidade, surgiu a opção pela adoção do *low-code* como forma de acelerar o processo de desenvolvimento e alcançar o objetivo de forma mais rápida.

Em um dos seus setores destinados aos treinamentos internos, foi pensado um software onde seria possível que os funcionários tivessem acesso aos seus treinos e pudessem solicitar novos treinamentos com instrutores específicos, sendo os instrutores responsáveis por administrar os funcionários associados aos seus treinos e fornecer ou solicitar informações necessárias.

Para melhorar a eficiência do processo de integração dos instrutores, a plataforma deve incluir funcionalidades para gerenciamento de recursos e acessos aos mesmos. Esses recursos devem

ser padronizados para garantir que o processo seja mais rápido e com menos erros. Além disso, a plataforma deve fornecer aos funcionários uma visão geral de todos os instrutores e seus recursos e acessos, para que haja transparência em relação ao processo de integração dos instrutores. Essa plataforma deve incluir as seguintes funcionalidades:

- Desenvolver um sistema de gerenciamento de recursos - O sistema deve permitir que os instrutores e os funcionários responsáveis por eles gerenciem os recursos disponíveis, incluindo equipamentos, ferramentas e materiais de treino. Isso ajudará a garantir que os instrutores tenham os recursos necessários para fornecer treinamento eficaz.
- Padronizar os processos de integração - Os processos de integração devem ser padronizados para garantir que ocorram de maneira consistente e com menos erros. Isso pode incluir a criação de *check-lists* e fluxos de trabalho padrão para cada etapa do processo de integração.
- Criar uma visão geral de todos os instrutores e seus recursos - A plataforma deve fornecer aos funcionários uma visão geral de todos os instrutores e seus recursos. Isso ajudará a garantir que os funcionários possam encontrar rapidamente os instrutores que precisam e os recursos necessários para fornecer treinamento eficaz.
- Fornecer suporte técnico - A plataforma deve fornecer suporte técnico para ajudar os funcionários a resolver problemas técnicos que possam surgir durante o processo de integração.
- Integrar a plataforma com outros sistemas - A plataforma deve ser integrada com outros sistemas internos, como sistemas de gerenciamento de recursos humanos e de treinamento, para garantir que os dados sejam compartilhados entre os sistemas e que os funcionários tenham acesso às informações necessárias.

Ao implementar as referidas sugestões, a empresa pode criar uma plataforma eficiente de integração de instrutores, que ajuda a garantir que os instrutores tenham os recursos necessários para fornecer treino eficaz e que o processo de integração ocorra de maneira consistente e com menos erros.

4.1.2 REQUISITOS FUNCIONAIS

Considerando o contexto exposto, os principais requisitos funcionais para cada um dos perfis de acessos foram os seguintes:

- Administrador

    - Gerenciar permissões da aplicação;
    - Criar perfis solicitantes para usuários externos;
    - Pesquisar usuários de forma mais específica;
    - Visualizar dados mestres.
- Aprovador
    - Visualizar e editar perfis dos instrutores;
    - Aprovar e gerenciar novas solicitações;
    - Disparar notificações por e-mail da ferramenta;
    - Exportar a lista de instrutores.
- Solicitante
    - Solicitar a integração de um instrutor, criando um perfil para o aprovador revisar e aprovar/solicitar mais informações;
    - Visualizar apenas as próprias solicitações e status.
- Usuário "Apenas leitura"
    - Ver e pesquisar todos os pedidos de instrutores;
    - Ler a solicitação, sem poder editá-la ou fazer ações que a alterem de qualquer forma ou formato;
    - Função para abrir o Outlook dentro dos pedidos.
- Desenvolvedor
    - Aceder todas as configurações da ferramenta (modelos de e-mail, links de acessos enviados nos e-mails, de acesso fornecidos pela empresa para ser integrada na plataforma, etc.).

Vale a pena ressaltar que podem existir outros requisitos funcionais implícitos ou explícitos que não foram mencionados e que também serão considerados na fase de construção.

Considerando os perfis de acesso, também foram levados em consideração os critérios de aceitação (CA) específicos para cada um dos perfis, tendo em conta as suas permissões e funções:

> CA1 – Criar uma solicitação: o solicitante abre a ferramenta e visualiza a tela de boas-vindas, contendo uma tabela com as suas solicitações e um botão para criar uma solicitação. O solicitante tem de ser capaz de fornecer informações para registar o instrutor. Nesta nova solicitação, os campos obrigatórios conterão uma estrela indicativa e terá um botão para enviar a nova solicitação. Feito isso o solicitante recebe

uma mensagem de sucesso e pode criar outras solicitações em sequência, se for necessário;

CA2 – Aprovar solicitações: o aprovador abre a ferramenta e visualiza a tela de boas-vindas, contendo uma tabela com todas as solicitações com as informações fornecidas pelo solicitante, e ao abrir uma solicitação possui três botões distintos, sendo: "Aprovar", "Rejeitar" e "Pedir mais informações". Quando o aprovador pressionar o botão para "Pedir mais informações", abre um campo para que ele possa descrever quais informações extras serão necessárias. Para as três opções do aprovador, sempre que tomar uma decisão o mesmo irá receber uma mensagem de sucesso, os *status* serão atualizados para ambas as partes e o solicitante também irá receber um e-mail com a atualização da sua solicitação.

CA3 – Iniciar os fluxos de trabalho: o aprovador é o responsável por tomar as decisões dentro dos fluxos de trabalhos, podendo: "Iniciar o fluxo de trabalho", "Reiniciar o fluxo", "Desativar o fluxo". Algumas destas opções ficam inativas de acordo com o *status*, para o caso de o fluxo já ter iniciado ou já estar completo, só fica possível que o aprovador reinicie ou desative o fluxo. O status "aguardando pré-requisitos", uma vez que estamos neste estado, necessita que sejam cumpridos todos os pré-requisitos para que o fluxo de trabalho seja iniciado de forma totalmente automática.

CA4 – Administrador geral: quando um administrador abre a ferramenta, já visualiza uma página de boas-vindas definida para ele, com todos os recursos presentes na plataforma. Tendo como principal função administrar os perfis e as devidas solicitações de acesso a ferramenta, também é responsável por gerir os *templates* dos e-mails que são enviados de acordo com a situação e de acordo com a função do perfil de usuário que irá receber o e-mail.

CA5 – Visualizar dados: o usuário com o acesso de "apenas leitura" ao abrir a ferramenta visualiza a tela de boas-vindas, contendo uma tabela com as solicitações e suas informações, assim como os fluxos de trabalho, onde também consegue visualizá-los, e os perfis dos instrutores. Porém, apesar de visualizá-los, não possui qualquer permissão para alterar qualquer informação referente as funcionalidades da ferramenta.

CA6 – Usuários registados: quando um usuário que já possui registo tenta aceder à ferramenta, vai para a página de *login*, onde é possível aceder o sistema sem precisar preencher usuário e senha, através de uma ligação com uma plataforma de autenticação gerida pela própria empresa, onde todos os dados do usuário ficam registados e o

mesmo só precisa realizar uma autenticação e o acesso já é liberado para dentro da ferramenta.

CA7 – Usuários não registados: quando um usuário sem registo tentar aceder à ferramenta, terá uma tela de boas-vindas contendo os seguintes textos: “infelizmente você ainda não tem acesso!”, “Por favor, preencha a solicitação de acesso e envie para o administrador, para que o mesmo possa lhe dar acesso”, “Em caso de dúvidas, entrar em contato com a equipa”. Nesta última frase é possível clicar num *link* de e-mail, o qual direciona automaticamente para o Outlook e com o destinatário pré-preenchido com o endereço de e-mail.

No contexto do estudo que será realizado, é importante observar que os requisitos a serem implementados apresentem diferentes níveis de complexidade. Isso ocorre porque um sistema geralmente é composto por um grande número de operações de baixa e média complexidade, juntamente com um conjunto menor de operações mais complexas.

Com isso, os requisitos CA1, CA2 e CA3 são considerados os mais significativos, a nível da definição da lógica da ferramenta. Os requisitos restantes consistem principalmente no desenvolvimento de operações de criar, ler, atualizar e apagar (*Create, Read, Update and Delete*, CRUD) básicas ou apenas na consulta de informações, que são considerados essenciais para o domínio, mas mais simples em termos de lógica de negócio e desenvolvimento e, portanto, mais fáceis de implementar. No entanto, é importante destacar que existem diferenças na complexidade das operações, dependendo das relações existentes entre conceitos. Por exemplo, a operação CRUD de criar as solicitações é mais simples do que a de aprovar, pois este último é composto por variáveis externas de informações, como status de cada solicitação para seguir em frente ou não. Em certos pontos, até de forma automática aplica-se uma lógica mais complexa que sempre avalia todas as informações dentro de cada solicitação. Planeja-se implementar mais de um requisito desse tipo para avaliar a progressão dos processos e tornar a ferramenta mais inteligente e autónoma. Espera-se que a implementação de um requisito semelhante a outro já implementado seja mais rápida, uma vez que já existe conhecimento prévio.

4.1.3 REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais são aspetos importantes a serem considerados no desenvolvimento de sistemas de software. Eles definem as características do sistema que não estão diretamente relacionadas às funcionalidades, mas sim à sua performance, funcionalidade, usabilidade, confiabilidade, segurança, entre outras características. Esses

requisitos podem ser críticos para o sucesso do sistema, uma vez que podem afetar a experiência do usuário, a eficiência do sistema e sua disponibilidade. Portanto, é essencial que os requisitos não funcionais sejam cuidadosamente definidos, testados e validados ao longo do ciclo de vida do software. A falta de atenção aos requisitos não funcionais pode levar a problemas de desempenho, usabilidade e segurança, que podem comprometer a qualidade do sistema e sua aceitação pelos usuários. Os requisitos considerados são listados a seguir.

- **Funcionalidade** - O presente requisito diz respeito aos aspetos operacionais do sistema, anteriormente descritos na Secção 4.1.2. No entanto, é importante destacar que os seguintes aspetos devem ser considerados:
    - Autenticação - garantir que somente os usuários autenticados que pertencem à empresa ou que possua vínculo de cliente tenham acesso ao sistema;
    - Autorização - o sistema a ser desenvolvido terá a participação de vários usuários, cada um com um papel diferente (conforme o descrito na Secção 4.1.2). Portanto, dependendo do seu papel, o usuário terá acesso apenas às funcionalidades para as quais tem permissão.
- **Usabilidade** - Este requisito diz respeito à interação do sistema com o usuário final e, no contexto deste estudo de caso, é pertinente considerar o seguinte aspeto:
    - Padrões de interface do usuário - as interfaces gráficas apresentadas ao usuário final devem seguir o mesmo esquema e disposição semelhante de informação, a fim de facilitar o processo de aprendizagem do usuário no novo sistema.
- **Confiabilidade** - Este requisito diz respeito à disponibilidade ou recuperação de falhas. No contexto deste estudo de caso, deve-se considerar o seguinte aspeto:
    - Disponibilidade - o sistema deve apresentar alta disponibilidade, principalmente porque, como mencionado na Secção 4.1.1 (Descrição de negócios), a indisponibilidade do sistema pode comprometer o processo de trabalho.
- **Desempenho** - Este requisito permite avaliar o desempenho do sistema em termos de tempo de resposta, capacidade e escalabilidade. Para este estudo de caso, os seguintes aspetos devem ser considerados:
    - Escalabilidade - o sistema deve ser capaz de lidar com um possível aumento no número de usuários e, consequentemente, no número de solicitações ao sistema.
    - Tempo de resposta - o sistema deve apresentar tempos de resposta baixos.

- **Suportabilidade** - Este requisito considera características como testabilidade, compatibilidade, manutenibilidade, entre outras. Neste caso, os seguintes aspetos devem ser considerados:
    - Testabilidade - deve ser possível garantir uma cobertura adequada de testes do sistema;
    - Flexibilidade - deve ser garantida a possibilidade de realizar alterações no sistema existente.

4.2 CASO DE ESTUDO 2

Esta seção aborda o campo de atuação do Caso de Estudo 2 e descreve os principais aspetos de sua implementação. No contexto do presente Caso de Estudo, além dos objetivos relacionados à definição da arquitetura, busca-se a capacidade de criar e avaliar soluções. O desenvolvimento concentra-se principalmente nos detalhes da modelagem de dados, dados essenciais, lógica de negócios e interface do sistema. Assim, é possível realizar uma análise detalhada dos recursos de resposta da plataforma destacada na pesquisa.

4.2.1 CARACTERIZAÇÃO DO NEGÓCIO

A empresa de engenharia multinacional é a mesma descrita no Caso de Estudo 1. Com isso, num dos seus setores de fornecimento, viu-se a necessidade de desenvolver uma ferramenta para tornar o processo de venda de configurações personalizadas de equipamentos hospitalares mais eficiente, i.e, a empresa pretende criar uma ferramenta de gerenciamento de vendas que permita que a equipa de vendas acompanhe o progresso das configurações dos equipamentos hospitalares criadas pelos clientes. Essa ferramenta deve incluir as seguintes funcionalidades:

- Painel de gerenciamento de vendas - uma interface de usuário que exibe as configurações criadas pelos clientes e o seu estado atual, desde "Configuração criada" até "Oferta oficial enviada ao cliente". Os representantes de vendas podem aceder a essa interface e verificar o estado de todas as configurações dos seus clientes.
- Fluxo de trabalho automatizado - um sistema de fluxo de trabalho automatizado que encaminha as configurações para a equipa de vendas, que pode rever e verificar as configurações. O sistema pode notificar automaticamente os clientes por e-mail quando uma configuração for analisada e aprovada pela equipa de vendas.
- Comunicação direta com o cliente - um sistema de mensagens integrado que permite que os representantes de vendas comuniquem diretamente com seus

clientes para esclarecer dúvidas, enviar atualizações sobre o estado de suas configurações e enviar a oferta oficial ao cliente.

- Armazenamento de informações: uma base de dados que armazena informações detalhadas sobre as configurações criadas pelos clientes e o histórico de comunicação com cada cliente. Isso ajudará a equipa de vendas a gerenciar as configurações de forma mais eficiente e a manter um registo de todas as interações feitas entre as partes.

Essa ferramenta pode ajudar na comunicação com os clientes de forma mais eficaz. Com isso, o setor da empresa pode reduzir o tempo necessário para concluir o processo de venda, melhorar a experiência do cliente e aumentar as vendas.

4.2.2 REQUISITOS FUNCIONAIS

Alguns exemplos de critérios de aceitação para os diferentes perfis de usuário são:

- Administrador Global
  - Deve ser capaz de criar e gerenciar perfis de usuário para todas as regiões;
  - Deve ser capaz de visualizar e editar todos os fluxos de trabalho;
  - Deve ter acesso a todas as configurações da plataforma.
- Administrador Regional
  - Deve ser capaz de criar e gerenciar perfis de usuário para a sua região;
  - Deve ser capaz de visualizar e editar os fluxos de trabalho da sua região;
  - Deve ter acesso às configurações da sua região na base de dados.
- Usuário externo/consumidor
  - Deve ser capaz de visualizar apenas suas próprias configurações na base de dados;
  - Deve ser capaz de criar e abrir novas configurações;
  - Deve ser capaz de enviar mensagens via *chat* para o representante de vendas.
- *Backoffice* de vendas
  - Deve ser capaz de visualizar todas as configurações na base de dados;
  - Deve ser capaz de gerenciar os fluxos de trabalho das configurações;
  - Deve ser capaz de criar e anexar propostas de orçamento para as configurações solicitadas.
- Representante de vendas
  - Deve ser capaz de visualizar todas as configurações na base de dados;

- Deve ser capaz de gerenciar e aprovar os fluxos de trabalho das configurações;
- Deve ser capaz de enviar e receber mensagens via *chat* com clientes e o *backoffice* de vendas.

- Especialista de produtos
  - Deve ser capaz de visualizar as configurações que lhe estão conectadas na base de dados;
  - Deve ser capaz de visualizar os fluxos de trabalho relacionado com a configuração solicitada;
  - Deve ser capaz de enviar e receber mensagens via *chat* com outros especialistas e o *backoffice* de vendas.
- Desenvolvedor
  - Deve ter acesso completo a todas as configurações e funcionalidades da ferramenta;
  - Deve ser capaz de visualizar e editar todos os fluxos de trabalho;
  - Deve ser capaz de aceder à API entre a ferramenta do configurador e a ferramenta já existente no sistema da empresa, voltada para a área hospitalar.

É importante destacar que podem existir requisitos funcionais implícitos ou explícitos que não foram mencionados e que serão considerados na fase de construção. Além disso, foram levados em consideração os critérios de aceitação específicos para cada perfil de acesso, de acordo com suas permissões e funções.

Para os requisitos funcionais, os critérios de aceitação são:

CA1. Base de dados de todas as configurações - Deve ser possível armazenar todas as informações de configuração num base de dados centralizado e acessível por todos os perfis de usuário autorizados.

CA2. Fluxo de trabalho visual numa configuração - Deve ser possível visualizar o fluxo de trabalho de cada configuração de forma clara e intuitiva para os clientes e vendas, indicando em que estado a configuração está atualmente.

CA3. Possibilidade da força de vendas verificar o que deve fazer em cada etapa do fluxo de trabalho - Deve ser possível definir fluxos de trabalho personalizados e claros para cada configuração, de modo que a equipa de vendas possa saber o que fazer em cada etapa do processo de configuração.

CA4. Função de *chat* para os usuários - Deve ser possível que os usuários comuniquem uns com os outros por meio de um *chat* interno, facilitando a troca de informações e a resolução de problemas.

CA5. Administração de fluxo de trabalho para edição de fluxos dependendo do país do cliente - Deve ser possível definir fluxos de trabalho personalizados para cada região ou país, garantindo que o processo de configuração atenda às necessidades e regulamentos específicos de cada país.

CA6. API entre a ferramenta de configurador e a ferramenta já existente no sistema da empresa, voltada para a área hospitalar.

No contexto do estudo em questão, é relevante destacar que os requisitos a ser implementados possuem distintos graus de complexidade. Tal facto se dá pelo motivo de que um sistema normalmente é composto por uma grande quantidade de operações de baixa e média complexidade, além de um conjunto menor de operações de maior complexidade.

Os requisitos CA1, CA2, CA4 e CA5 são considerados os mais significativos, pois definem a lógica da ferramenta. Os demais requisitos consistem principalmente em operações CRUD básicas ou apenas na consulta de informações, que são essenciais para o domínio, porém mais simples em termos de lógica de negócio e desenvolvimento, portanto, mais fáceis de implementar.

No entanto, é importante ressaltar que a complexidade das operações varia de acordo com as relações entre os conceitos. Por exemplo, a operação de aprovação é mais complexa do que a operação de solicitação, pois envolve informações externas ao sistema, fluxo de trabalho e região, que determinam se o fluxo deve continuar ou não. Para isso, uma lógica mais complexa foi criada para avaliar cada etapa do fluxo de trabalho e todas as informações associadas. Futuramente planeia-se implementar mais requisitos semelhantes para tornar a ferramenta mais inteligente e autônoma. A implementação de requisitos semelhantes deve ser mais rápida, uma vez que já existe conhecimento prévio.

4.2.3 REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais para este Caso de Estudo são semelhantes aos identificados na Secção 4.1.3, mas incluem os seguintes aspetos adicionais:

- Funcionalidade
  - Autenticação - deve-se garantir que somente usuários registados tenham acesso à aplicação.
- Usabilidade

    - Facilidade de uso - interfaces gráficas fornecidas aos usuários devem ser intuitivas e seguir um padrão comum que facilite a memorização das ações realizadas na aplicação. Além disso, deve-se garantir o menor número possível de cliques entre as diferentes ações.
- Confiabilidade
    - Disponibilidade - o sistema deve apresentar alta disponibilidade, especialmente porque, como mencionado na Secção 4.2.1, a aplicação pode ter alta demanda e a indisponibilidade compromete os pedidos dos usuários.

Esta secção ilustra dois casos de estudo envolvendo a referida empresa multinacional no setor de engenharia. Cada caso oferece uma visão aprofundada dos desafios e soluções enfrentados por esta empresa em seu contexto específico de atuação. Essas análises servem como valiosos *insights* para compreender como a empresa aborda questões complexas de engenharia, adapta-se a cenários variados e alcança seus objetivos em um ambiente global e dinâmico.

Na próxima secção, exploraremos em detalhes a implementação de cada Caso de Estudo mencionado anteriormente. Analisaremos como esses projetos foram planeados e executados, as estratégias adotadas e os resultados alcançados. Esta análise aprofundada fornecerá uma visão abrangente das abordagens específicas utilizadas em cada caso, contribuindo para uma compreensão mais completa das práticas e soluções implementadas pela referida empresa.

5

IMPLEMENTAÇÃO

Neste capítulo mergulharemos na fase crucial do processo de desenvolvimento, a implementação. Após um planejamento detalhado e uma análise cuidadosa dos requisitos, é hora de transformar os conceitos e as ideias num sistema funcional.

A implementação é o estágio em que o projeto ganha vida, onde a lógica é programada, as interfaces são construídas e as funcionalidades são integradas. Aqui, focaremos em como traduzir as especificações em código, utilizando o *low-code* como ferramenta de desenvolvimento, através da plataforma Mendix.

Deste modo, este capítulo irá acompanhar os dois casos de estudo apresentados no capítulo anterior, exemplificando cada etapa do processo de implementação. Analisaremos os desafios enfrentados e as soluções adotadas.

5.1 PRINCIPAIS CONCEITOS DA PLATAFORMA MENDIX: VISÃO GERAL

A plataforma Mendix oferece o Mendix Studio Pro, que é um ambiente de desenvolvimento integrado (*integrated development environment*, IDE) que permite criar aplicativos com ações completas, exigindo um conhecimento mais avançado em programação, em comparação com o Mendix Studio (ver Secção 2.6.3).

No Mendix Studio Pro todas as aplicações têm a mesma estrutura, concentrada no App Explorer, que consiste numa pasta para configurações de segurança, navegação e outras configurações relevantes no módulo do aplicativo. Cada módulo do aplicativo possui o seu próprio modelo de domínio, que representa entidades e seus relacionamentos por meio de associações [44]. Cada entidade possui uma propriedade que determina se ela será persistente, ou seja, terá uma tabela específica para que os seus dados sejam armazenados na base de

dados. Caso contrário, a entidade é armazenada somente em memória durante a execução do software.

As entidades são representadas por atributos, caracterizados por um tipo de dado (por exemplo, decimal, enumeração, binário, *AutoNumber*) e podem ter uma ou mais regras de validação ou um *Microflow* que calcula o valor do atributo. Os relacionamentos entre as entidades são definidos por associações, que têm uma multiplicidade indicada por 1 ou *. Essas associações podem ser do tipo Um-para-Muitos (1:∗), Muitos-para-Muitos (∗:∗) ou Um-para-Um (1:1). Além disso, as entidades podem ter generalização (frequentemente chamada de herança), permitindo que se relacionem com entidades do sistema [44].

No Mendix, a lógica de negócios de um aplicativo pode ser implementada por meio de *Microflows* e *Nanoflows*. Os *Microflows*, cuja notação gráfica é baseada no BPMN, permitem realizar várias ações, como seja, criar e atualizar objetos ou exibir páginas. Da mesma forma, os *Nanoflows* permitem expressar a lógica de um aplicativo, mas oferecem vantagens, tais como, funcionar em diretamente no navegador/dispositivo e ser utilizados *offline* [44]. No entanto, não é recomendado realizar ações de base de dados nesse tipo de fluxo. Esses fluxos podem ter parâmetros de entrada preenchidos no ponto em que o fluxo é acionado [44]. Além disso, eles possuem um conjunto de atividades sequenciais que correspondem a ações a ser executadas, podendo incluir condições e ciclos para iterar sobre uma lista ou condição.

A plataforma Mendix permite integração com outros aplicativos, que pode ser feita usando protocolo de dados aberto (*Open data protocol*, OData), Transferência de Estado Representacional (*Representational State Transfer*, REST) ou Protocolo de acesso a objetos simples (*Simple Object Access Protocol*, SOAP/Web) *Services*. A plataforma pode importar ou exportar dados em linguagem de marcação extensível (*Extensible Markup Language*, XML) e JSON. Para isso, são utilizados mapeamentos - *Import Mapping* e *Export Mapping* - que convertem objetos do Mendix para XML ou JSON, respectivamente.

Em relação às interfaces gráficas das aplicações, no Mendix o resultado é sempre um aplicativo de página única (*Single-Page Application*, SPA), o que significa que toda a interação ocorre numa única janela do navegador [44].

5.2 CASO DE ESTUDO 1

No Caso de Estudo 1 pretendia-se adotar uma abordagem de arquitetura monolítica combinada com arquitetura em camadas. Para implementação desta abordagem, foi criada uma aplicação que abrange a modelagem de entidades, lógica de negócio e interfaces gráficas. O projeto foi organizado em módulos para facilitar o desenvolvimento. Nesse caso não foi viável

implementar aplicações finais separadas para cada papel de usuário. Portanto, foi necessário considerar as permissões dos usuários nas páginas da aplicação. Essa consideração foi feita porque a criação de novos módulos exigia a definição de um modelo de domínio.

Com base no requisito analisado, a implementação começou com a definição das entidades que compõem o domínio do problema. Utilizou-se a funcionalidade de arrastar e soltar (*drag and drop*) para criar as entidades e, em seguida, configurou-se os atributos e associações. Para permitir que um cliente autenticado faça encomendas, foi necessário relacionar essa entidade com uma entidade de autenticação. A plataforma oferece a entidade `Account` que representa um representa uma conta de usuário ou um perfil de usuário. Portanto, criou-se uma entidade chamada `AccountExtension` com uma referência à entidade `Account` por meio da propriedade `Generalization`. Além disso, foram criadas as entidades `PartnerRequest`, `RoleRequest` e `DataViewSettings`, com os seus devidos atributos necessários à cada entidade.

A Figura 20 apresenta as entidades `AccountExtension`, `PartnerRequest`, `RoleRequest` e `DataViewSettings`, com os respetivos atributos e a associação entre as mesmas.

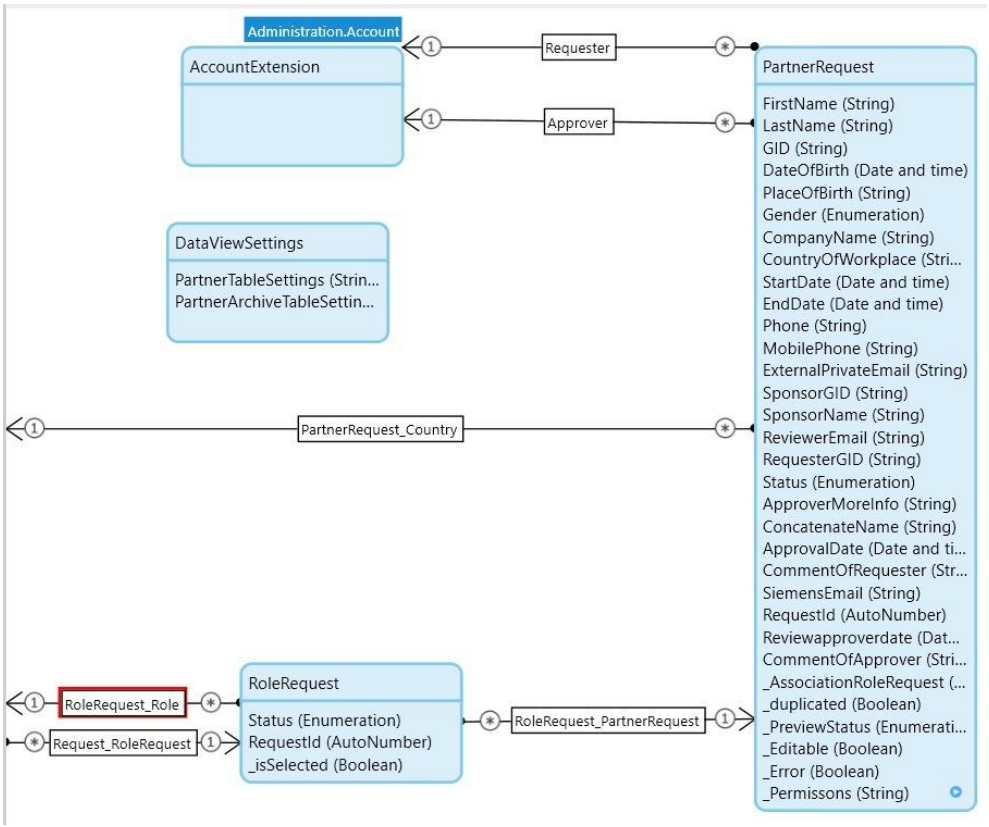


Figura 20. Entidades `AccountExtension`, `PartnerRequest`, `RoleRequest` e `DataViewSettings`

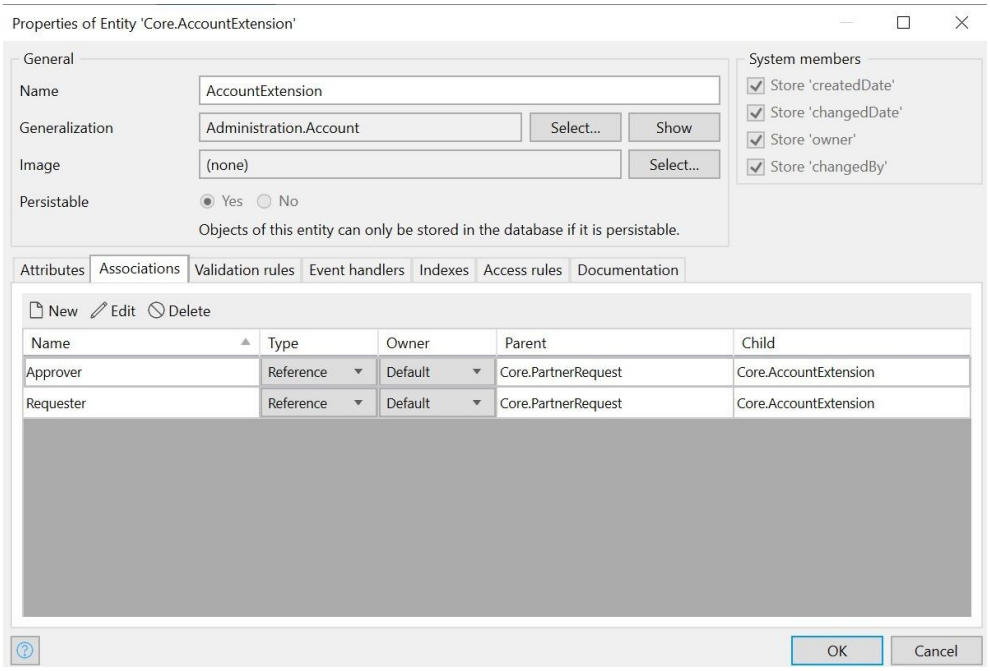


Figura 21. Propriedades da entidade `AccountExtension`

Na definição dos atributos das solicitações, em casos particulares como nas permissões, foi implementado um *Microflow* para permitir gerar as informações deste atributo auxiliar. Assim, ao invés do atributo ser recebido e armazenado, este é definido a partir de um fluxo, que é considerado na criação da entidade. Quando este tipo de fluxo é definido não é necessário realizar qualquer atribuição ou chamada ao longo da implementação da lógica. No entanto, quando esses tipos de atributos calculados são utilizados, devem ser apenas para leitura quando apresentados em algum tipo de formulário. Para além disso, foram consideradas algumas regras de validação para garantir a obrigatoriedade de alguns atributos. As associações entre as entidades são definidas e representadas visualmente, sendo estas identificadas por um nome e por uma cardinalidade. Com isso temos que a entidade `PartnerRequest` é *owner* e apresenta uma relação de muitos para um com a entidade `AccountExtension`. Por outro lado, a entidade `RoleRequest` é *owner* e apresenta uma ligação de muitos para um com a entidade `PartnerReques` (ver Figura 20). Para armazenar informações relevantes e anteriores das solicitações foi criado um atributo auxiliar enumerável, dado que o valor é fixo de outra enumeração existente. Estes enumeráveis são preenchidos com valores pretendidos, sendo mostrado na definição do atributo da entidade `PartnerRequest`. No caso das enumerações é possível apresentar um valor por defeito que, neste caso, não foi necessário.

É possível observar que a plataforma utiliza uma base de dados relacional, acedida através de uma abordagem baseada em mapeamento objeto-relacional (*Object-Relational Mapping*,

ORM). Com essa abordagem, as estruturas da base de dados são criadas com base nas entidades definidas no modelo de domínio. Isso resulta num alto nível de abstração na plataforma, onde as ações na base de dados são executadas por meio de *Microflows*, utilizando ações como, por exemplo, *Commit*, *Change*, *Create*, *Delete Object* ou *Retrieve*. Nessa implementação, a plataforma é responsável por lidar com a abstração dos dados e das ações na base de dados na camada de acesso aos dados.

Na plataforma é possível realizar eventos diretamente nas páginas da aplicação, como *Save Changes*, *Create Object* ou *Delete*, permitindo, por exemplo, a criação direta de objetos associados a um formulário. Essa abordagem pode ser mais rápida e conveniente, especialmente quando não é necessário considerar lógica de negócio adicional. Caso contrário é necessário criar fluxos que descrevam a lógica, tal como ocorre no contexto da criação de uma nova solicitação, tendo sido implementado um *Microflow* para submeter uma nova solicitação de `PartnerRequest` (ver Figura 22).

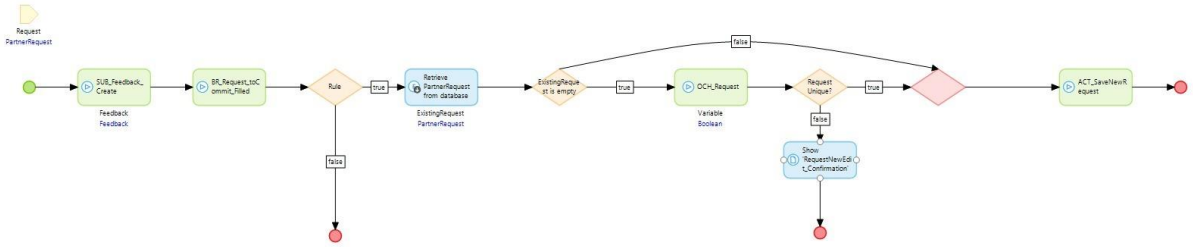
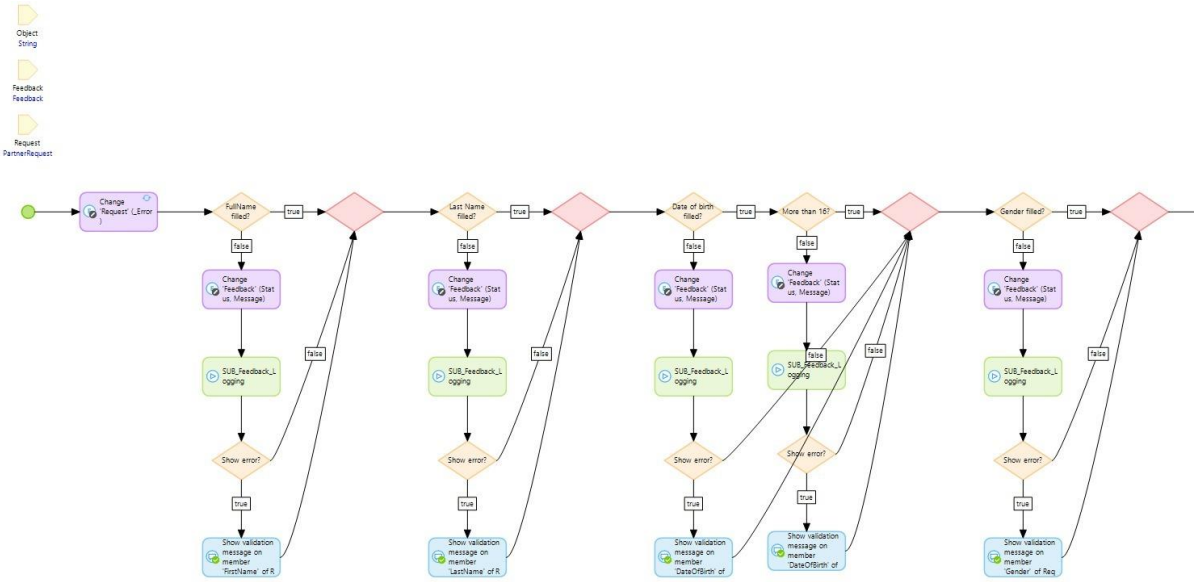


Figura 22. *Microflow* de submissão de um novo *request*



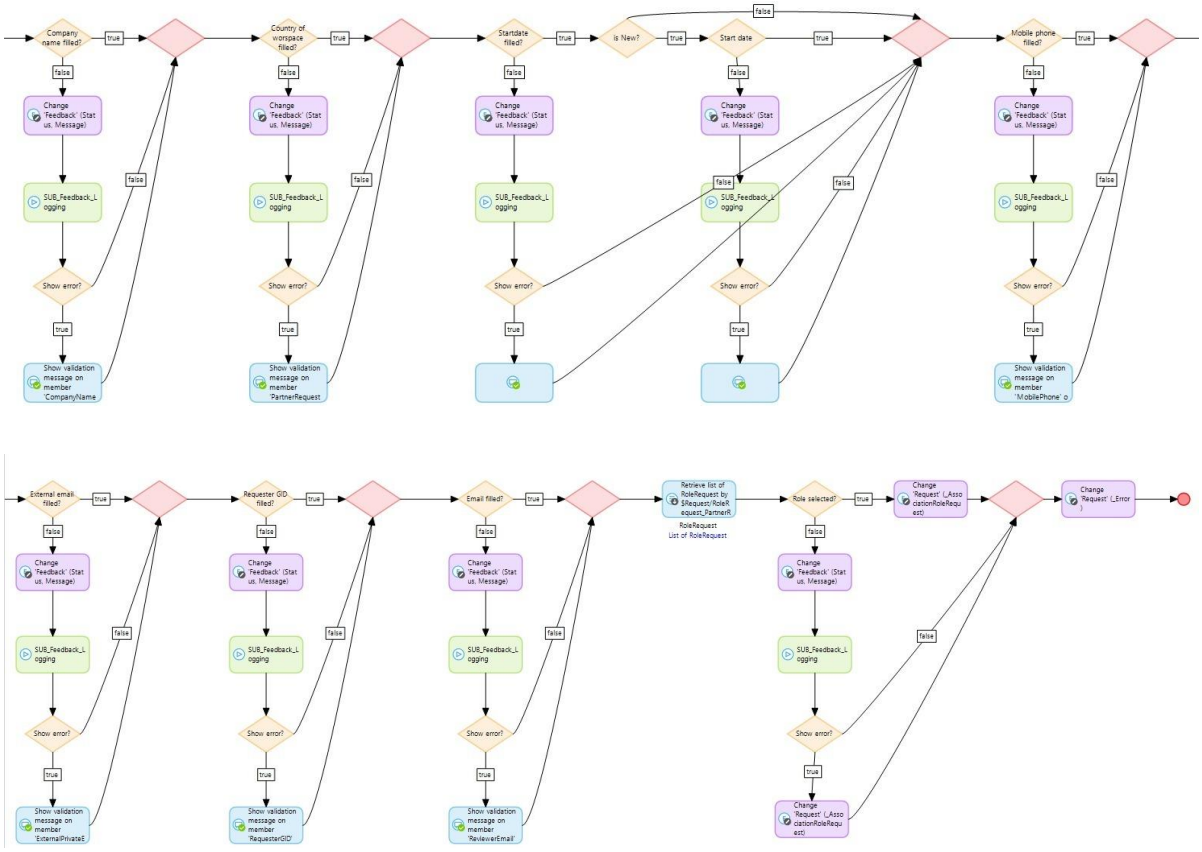


Figura 23. *Sub-Microflow* para verificar os campos submetidos em um `PartnerRequest`

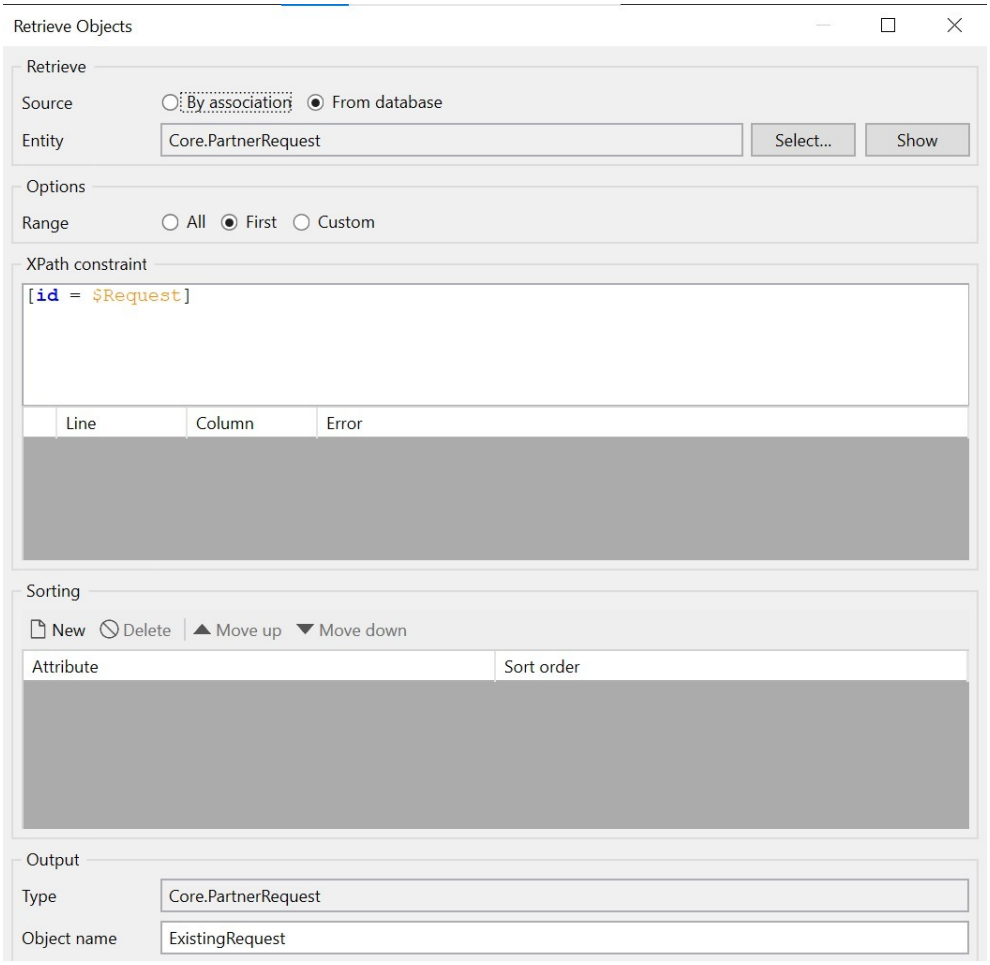


Figura 24. Configuração para identificar se já existe algum *request* igual na base de dados

O fluxo para submeter uma nova solicitação começa por verificar se a solicitação é única e recebe um parâmetro, o objeto `PartnerRequest` e começa por criar um *feedback* (ver Figura 22). Caso ocorra algum tipo de erro, este *feedback* é incrementado e apresentado na tela do usuário. De seguida é feita através de um *Sub-Microflow* a verificação de cada um dos campos presentes no formulário de solicitação (ver Figura 23), onde este *Sub-Microflow* irá retornar um erro caso tenha campos vazios ou com informações inadequadas para aquele campo e será alterado o *feedback*. Os passos seguintes estão focados na verificação de que trata-se de um único pedido com as informações contidas nos campos presente na base de dados. Por fim, é necessário fazer a submissão da solicitação depois de todas as verificações através do fluxo *ACT_SaveNewRequest*, cuja implementação é apresentada na Figura 25.

Este *Sub-Microflow* recebe como parâmetro o objeto do formulário de solicitação e inicia as ações dentro da base de dados com os atributos sendo preenchidos. Segue-se ainda dentro desta submissão do formulário, o envio de notificações via e-mail com alguns dados que foram preenchidos e sendo uma confirmação da criação para o usuário saber que seu pedido foi

enviado e está disponível para que seja avaliado e aprovado ou negado pelo responsável desta área. Dentro deste *Microflow* notam-se duas decisões a serem tomadas, ambas são para o usuário receber um e-mail ou uma mensagem na tela de acordo com o tipo de submissão, pois a verificação tende a resultar em se o formulário preenchido é novo ou se somente está sendo feita uma correção de dados, ou seja, o usuário será notificado por e-mail e no seu ecrã, com uma mensagem específica de acordo com a sua ação.

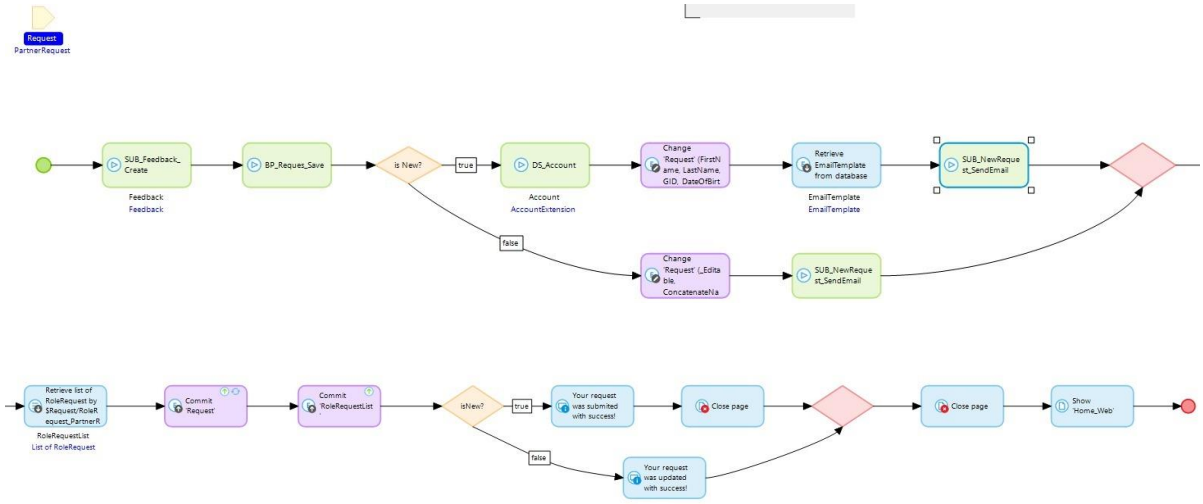


Figura 25. *Microflow* para a criação ou atualização de um *request*

Para a implementação da lógica apresentada na Figura 25 foi necessário entender as necessidades, os dados obrigatórios e como a informação seria tratada ao longo do processo de criação de novo *Request*. Todos os passos apresentados são feitos a partir do momento que o usuário realiza a submissão do formulário, pois sempre que um formulário com os dados preenchidos é submetido e vai para a base de dados é necessário que seja verificado se não existe outro com o mesmo ID, o que poderia gerar erros nos restantes processos e, por fim, mostrar as informações erradas na tela do usuário.

Durante a implementação da interface gráfica para a criação de uma solicitação, optou-se pelo uso de um modelo de página denominado *Form Vertical*, disponibilizado pela plataforma, que é específico para a construção de formulários. O formulário criado, exibido na Figura 26, é baseado numa *data view*, e foi necessário configurar sua fonte de dados para mapear as informações de acordo com a entidade "`PartnerRequest`". Com essa alteração na fonte de dados, o formulário foi mapeado para exibir os atributos correspondentes a entidade e alguns outros atributos associados e necessários para o preenchimento da solicitação. Todos os campos foram apresentados de acordo com tipo de atributo, seguindo a orientação e organização previamente estabelecida pelos *Project Owners*. A estruturação seguiu também

as boas práticas indicadas pela plataforma de desenvolvimento, onde foi criado um *Snippet* em que pudesse ser reaproveitado de forma rápida para outras situações ou até mesmo outras estruturas em que fossem necessários os mesmos campos.

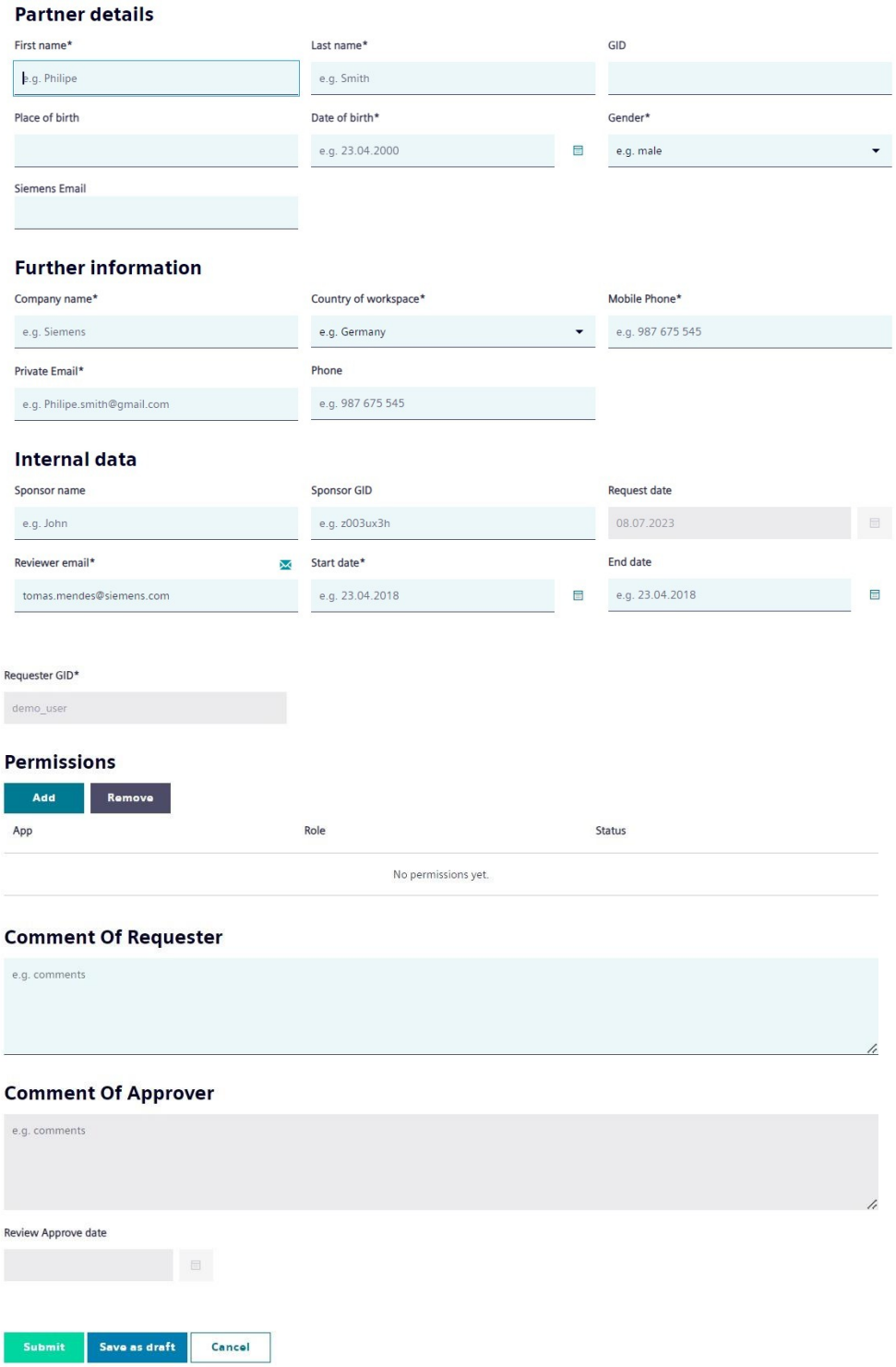


Figura 26. Formulário para a criação de um novo `PartnerRequest`

A *Data view* exibida no formulário não considera automaticamente as permissões que cada solicitação pode necessitar. No entanto, por meio da associação entre as entidades, adicionou-se uma *Data grid 2*, definindo a entidade `RoleRequest` como a fonte de dados, com base na associação *RoleRequest_PartnerRequest*. Essa *Data grid 2* resulta numa tabela e botões correspondentes para adicionar ou remover uma função de usuário. Foi necessário criar e associar uma nova página para a primeira ação. Essa nova página foi criada da mesma forma que a anterior, mas com a configuração da *data view* para utilizar a entidade `PartnerRequest` como fonte de dados e, inseriu-se dentro desta, um *Data grid* através de um *Xpath* para exibir funções disponíveis para serem selecionadas pelo usuário.

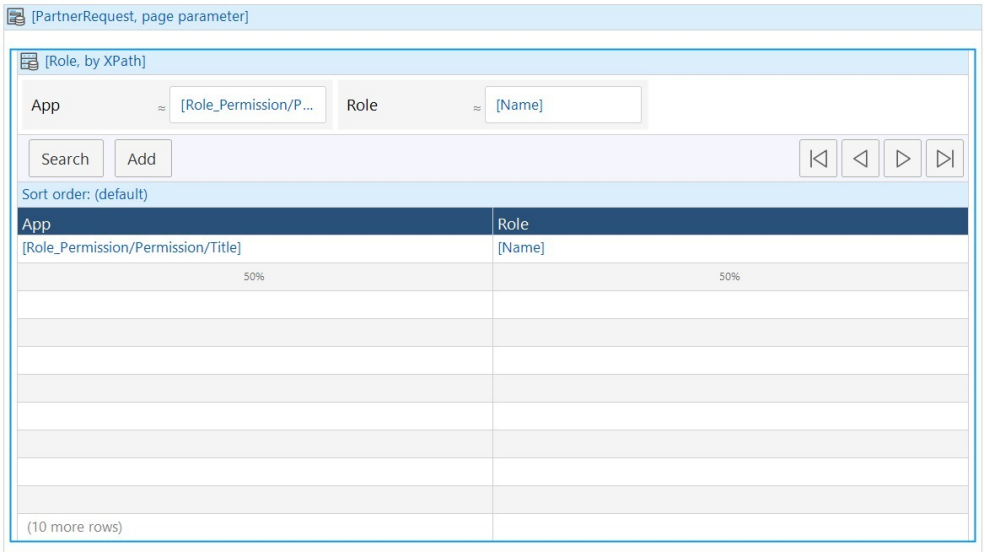


Figura 27. Página para selecionar as funções dentro do formulário `PartnerRequest`

Além disso, no formulário de criação de um `PartnerRequest`, no botão *Submit*, foi definido o evento *OnClick* que chama o *Microflow* mencionado anteriormente (Figura 23). Esse fluxo começa validando a lista com os dados preenchidos e, caso esteja vazia, exibe uma mensagem ao usuário, indicando os campos obrigatórios, sendo marcados com um fundo vermelho e uma legenda indicando com o que deve ser preenchido ou, em caso de mal preenchimento, o que está mal (ver Figura 28). Nesse caso, a validação foi feita no nível do *Microflow*, pois trata-se de informações da *Data grid*.

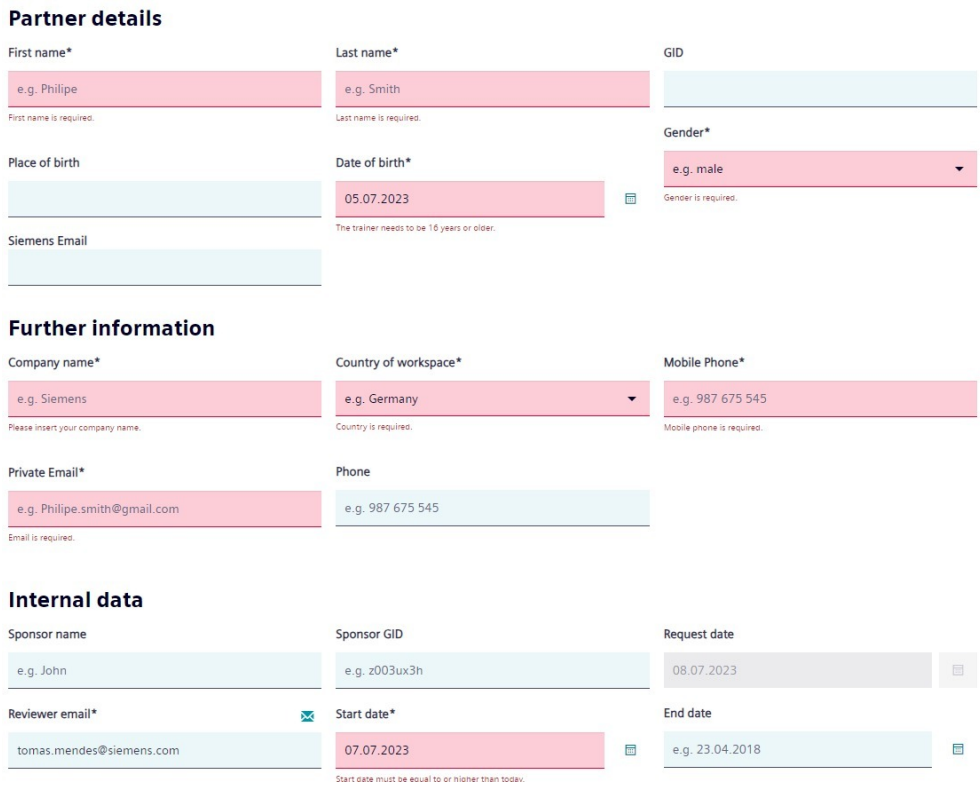


Figura 28. Ilustração de submissão de um `PartnerRequest` com campos em branco e com informações erradas

Adicionalmente, nesta interface, foram configuradas as propriedades de visibilidade de navegação, permitindo que botões e ações tornem-se possíveis consoante a uma determinada função de usuário e até de acordo com o estado de um `PartnerRequest`. Essas funções são consideradas apenas nos níveis de segurança *Prototype/demo* ou *Production*. Nesse nível, também é possível atribuir permissões de acesso a *Microflows* e *Nanoflows.*

Nesta solução, foram implementadas as boas práticas recomendadas pela plataforma, como a criação de um *Snippet* onde estão inseridos os campos de um "`PartnerRequest`", todas as nomenclaturas desenvolvidas para nomear as lógicas de programação desenvolvidas com suas respetivas etapas, as restrições de acesso dentro das entidades através de *Xpath* e também as condições de visibilidade de acordo com as necessidades do projeto, seguindo as orientações da plataforma. Na página principal onde é possível visualizar as suas respetivas solicitações e informações inseridas, foi implementado um *Widget* chamado de *Data grid 2* (ver Figura 29) onde é possível esconder colunas da tabela para que as informações fiquem mais bem organizadas e para que o usuário possa escolher quais informações quer visualizar no seu ecrã (ver Figura 30).

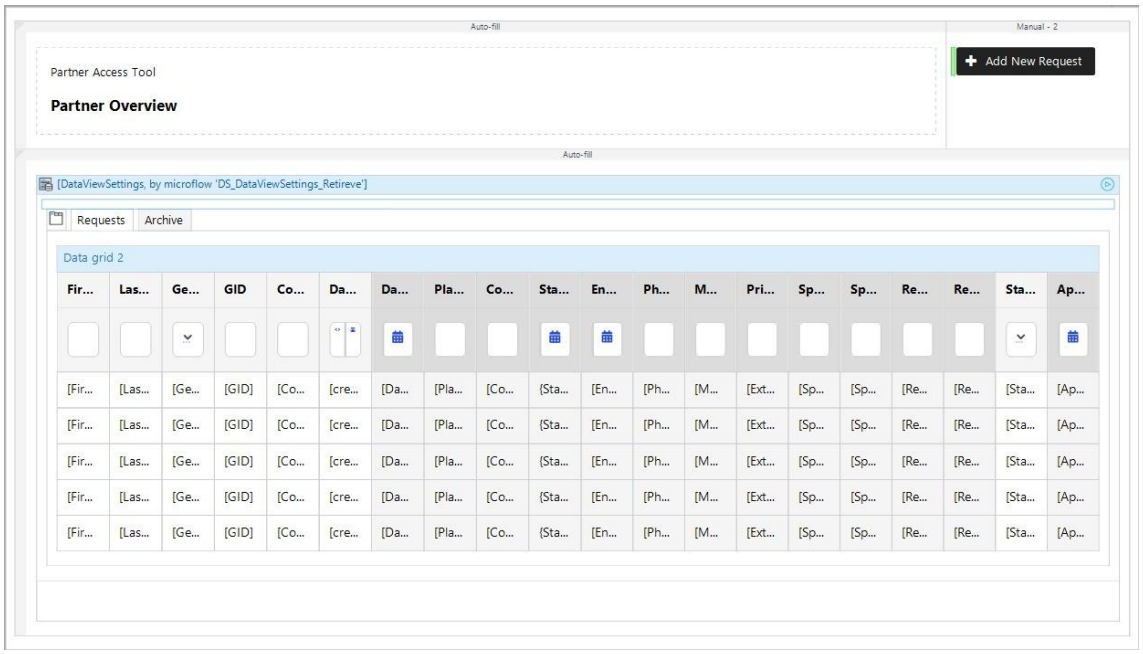


Figura 29. Estrutura da página principal dentro da plataforma Mendix

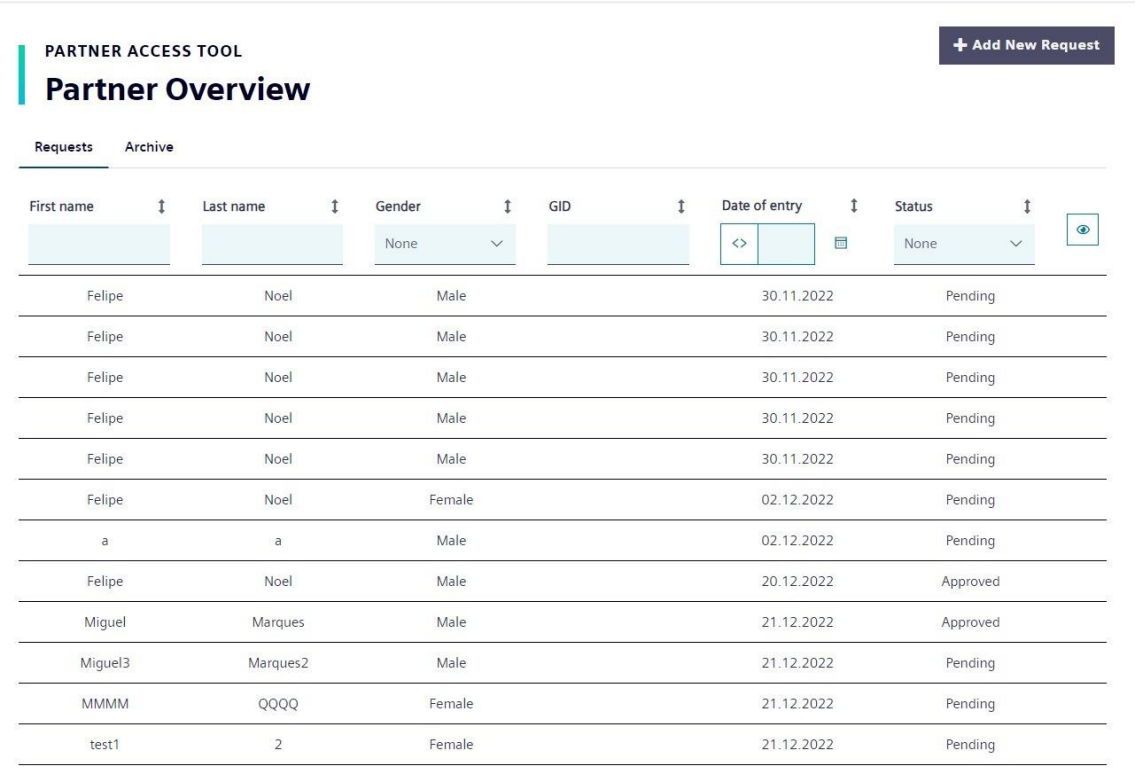


Figura 30. Layout final na tela do usuário

Dentro do contexto dessa plataforma, foi viável implementar a solução seguindo a arquitetura desejada. Apesar de a aplicação desenvolvida inicialmente apresentar apenas o modelo de domínio, foi intuitivo definir a estrutura para seguir as camadas da arquitetura pretendida. Isso

se deve ao facto de a plataforma permitir a organização da solução em pastas e fornecer conceitos como modelo de domínio, *Microflows* e interfaces, os quais puderam ser facilmente associados às camadas da aplicação.

Durante o desenvolvimento, foi possível garantir que cada camada tivesse uma única responsabilidade e se comunicasse com a camada imediatamente inferior. No entanto, observou-se que, ao longo do processo, entidades do domínio foram utilizadas na definição de lógica e interfaces. Embora não tenha sido considerada uma abordagem alternativa, seria possível definir entidades que não são persistentes na base de dados, utilizando-as apenas para transferência de informações.

5.3 CASO DE ESTUDO 2

No Estudo de Caso 2 foi feita a mesma conceção e análise inicial que foi feita para o Caso de Estudo 1, porém para outro projeto e outro contexto.

A plataforma oferece a entidade `Account`, que representa uma conta de usuário ou perfil de usuário. Portanto, foi criada uma entidade chamada `AccountExtension`, com uma referência à entidade `Account` por meio da propriedade `Generalization`, para além da autenticação, utilizando a mesma propriedade foi criada uma entidade `Attachment` para que fossem anexados os documentos dentro de cada configuração de pedidos de produtos. Além disso, foram criadas as entidades `ProductConfiguration`, `Attachment`, `Filter`, `Overview_Helper`, `WorkflowConfig`, `GroupConfig`, `WorkflowInstance`, `GroupInstance`, `TaskConfig`, `FieldConfig`, `TaskInstance`, `FieldInstance`, `EmailNotification`, `Conversation`, `Conversation_Helper` e `Region`, com seus respetivos atributos necessários a cada entidade e suas respetivas associações de acordo com as necessidades de iteração entre as mesmas.

As Figura 31-Figura 34 apresentam as entidades citadas anteriormente, ilustrando os seus atributos, as suas associações e o tipo de entidade, dentro dos seus respetivos módulos.

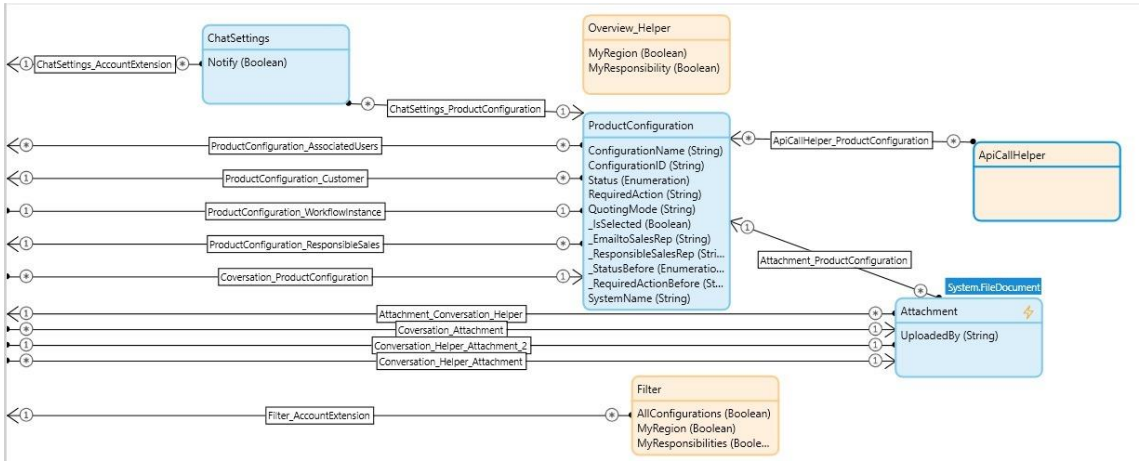


Figura 31. *Domain model* para a configuração do produto

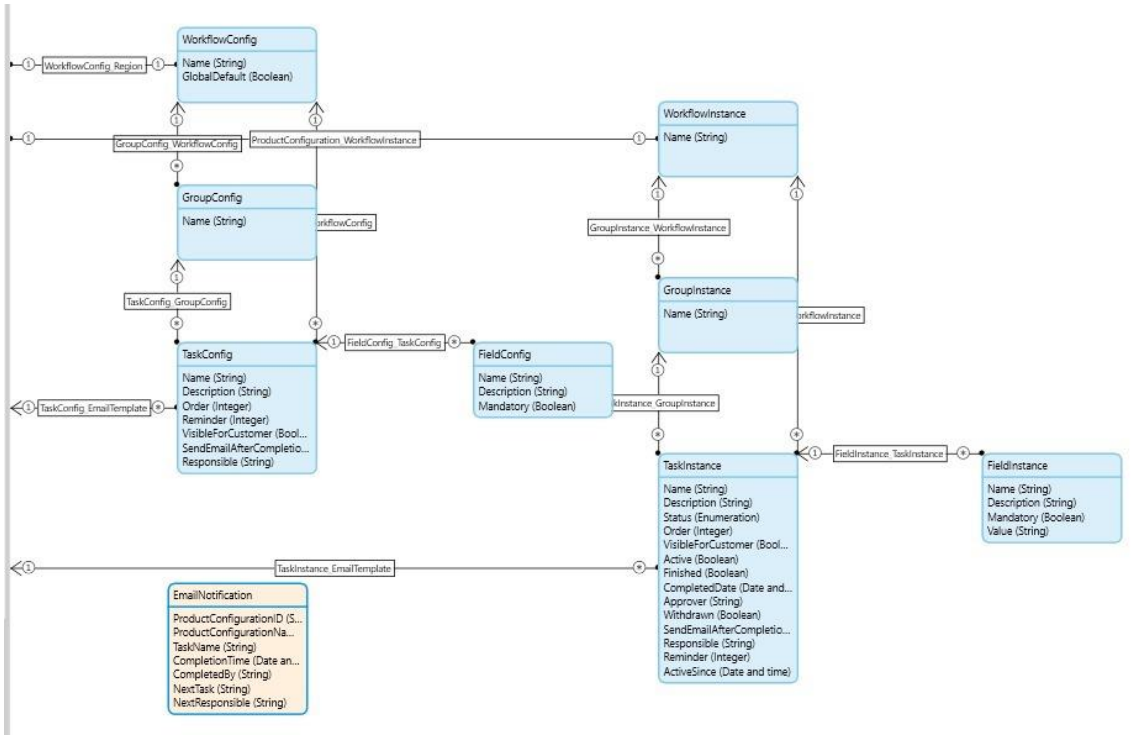


Figura 32. *Domain model* para a configuração dos fluxos de trabalho

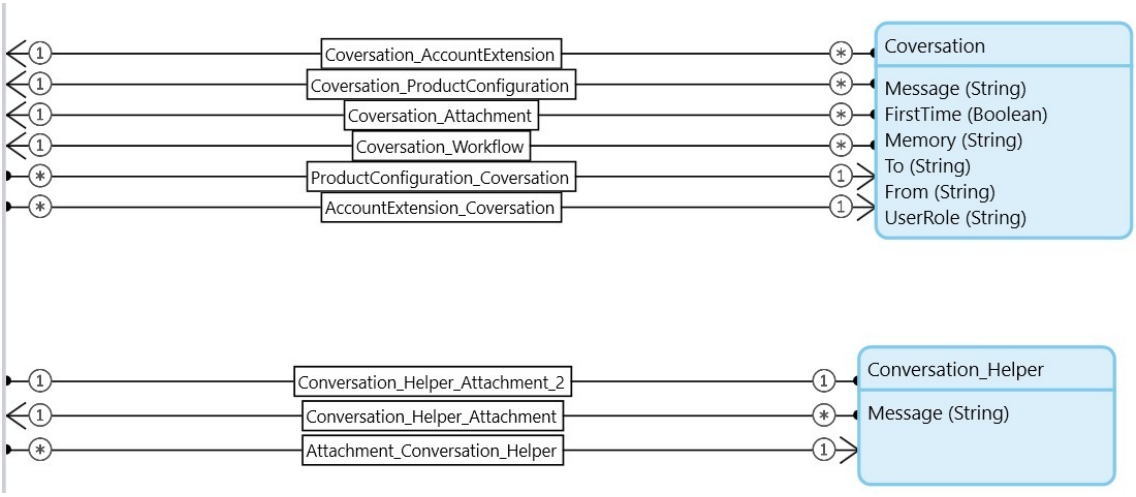


Figura 33. Domain model para configuração do *chat* em tempo real dentro de cada configuração de produto

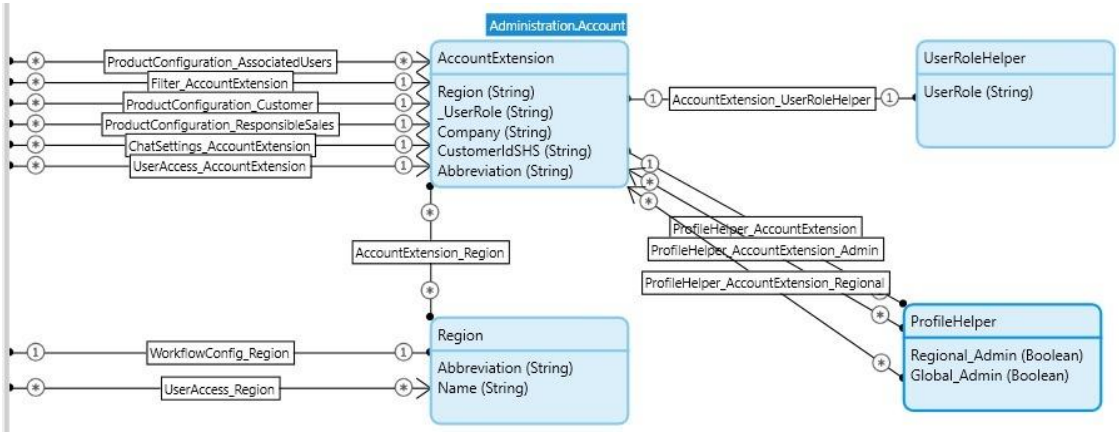


Figura 34. *Domain model* para utilizar o sistema de controle de usuários da plataforma

As associações entre as entidades são definidas e representadas visualmente, sendo identificadas por um nome e uma cardinalidade. Por exemplo, a entidade `ProductConfiguration` é a proprietária (*Owner*) e possui uma relação de "muitos para um" com a entidade `Costumer`. Por sua vez, a entidade `Conversation` também é proprietária e possui uma relação de "muitos para um" com a entidade `ProductConfiguration` (ver Figura 31).

Para armazenar os estados em que se encontra uma configuração e históricos anteriores, foi criado um atributo principal e um auxiliar enumerável. Esses enumeráveis são preenchidos com valores pré-definidos, como mostrado na definição do atributo da entidade `ProductConfiguration`. É possível definir um valor padrão para as enumerações, mas no caso em questão, essa definição é feita na lógica de criação tendo sido orientada pelos

próprios clientes, com a intensão de facilitar o entendimento e de ajudar no acompanhamento pedidos.

A plataforma utilizada neste estudo de caso emprega uma abordagem de ORM para aceder uma base de dados relacional. Essa abordagem oferece uma forma eficiente de criar as estruturas da base de dados com base nas entidades definidas no modelo de domínio. Por meio da base de dados, a plataforma proporciona um alto nível de abstração, permitindo que as operações na base de dados sejam executadas por meio de *Microflows*, que empregam ações como *Commit*, *Change*, *Create*, *Delete Object* ou *Retrieve*. Assim, a plataforma assume a responsabilidade de gerenciar a abstração dos dados e das operações na base de dados na camada de acesso aos dados. Essa abordagem simplifica o desenvolvimento e aumenta a produtividade dos desenvolvedores ao oferecer uma interface intuitiva para interagir com o base de dados relacional.

A plataforma oferece a funcionalidade de realizar eventos diretamente nas páginas da aplicação, como *Save Changes*, *Create Object* ou *Delete*, possibilitando a criação direta de objetos associados a um formulário. Essa abordagem pode ser vantajosa em termos de agilidade e conveniência, especialmente quando não é necessário considerar lógica de negócio adicional. No entanto, caso seja necessário incorporar lógica de negócio mais complexa, é necessário criar fluxos que descrevam essa lógica de maneira adequada. Esses fluxos permitem definir ações específicas a serem executadas, garantindo um controle mais preciso sobre o processo e considerando as regras e restrições necessárias. Portanto, a plataforma oferece flexibilidade para escolher a abordagem mais adequada às necessidades de cada caso, seja através dos eventos diretos nas páginas ou da criação de fluxos para lidar com a lógica de negócio.

Neste contexto foi desenvolvido um *Microflow* para fazer a submissão de uma nova configuração, onde é feita uma verificação se existe algum fluxo de trabalho orientado para a configuração em questão e também cria uma associação entre o pedido em questão e o fluxo, sendo então aberta a possibilidade de ter o *chat* para o mesmo, anexar documentos e realizar o acompanhamento de cada etapa do fluxo de trabalho, além de possuir o histórico do que foi feito, alterado, quem realizou a alteração, quando foi feita a alteração e quem foi associado a configuração.

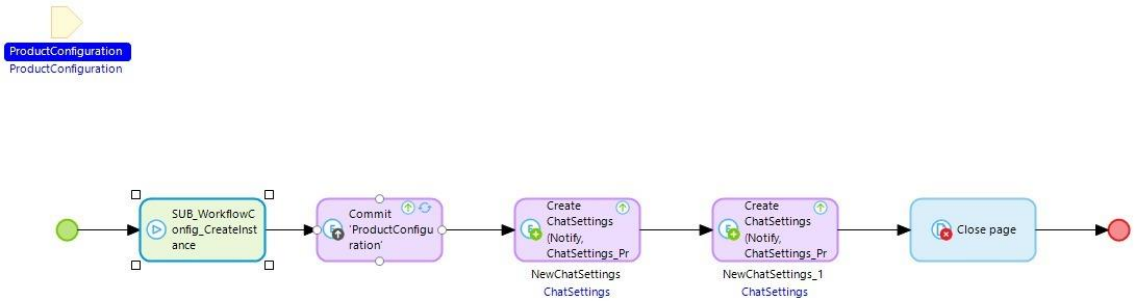


Figura 35. *Microflow* de submissão de uma nova configuração

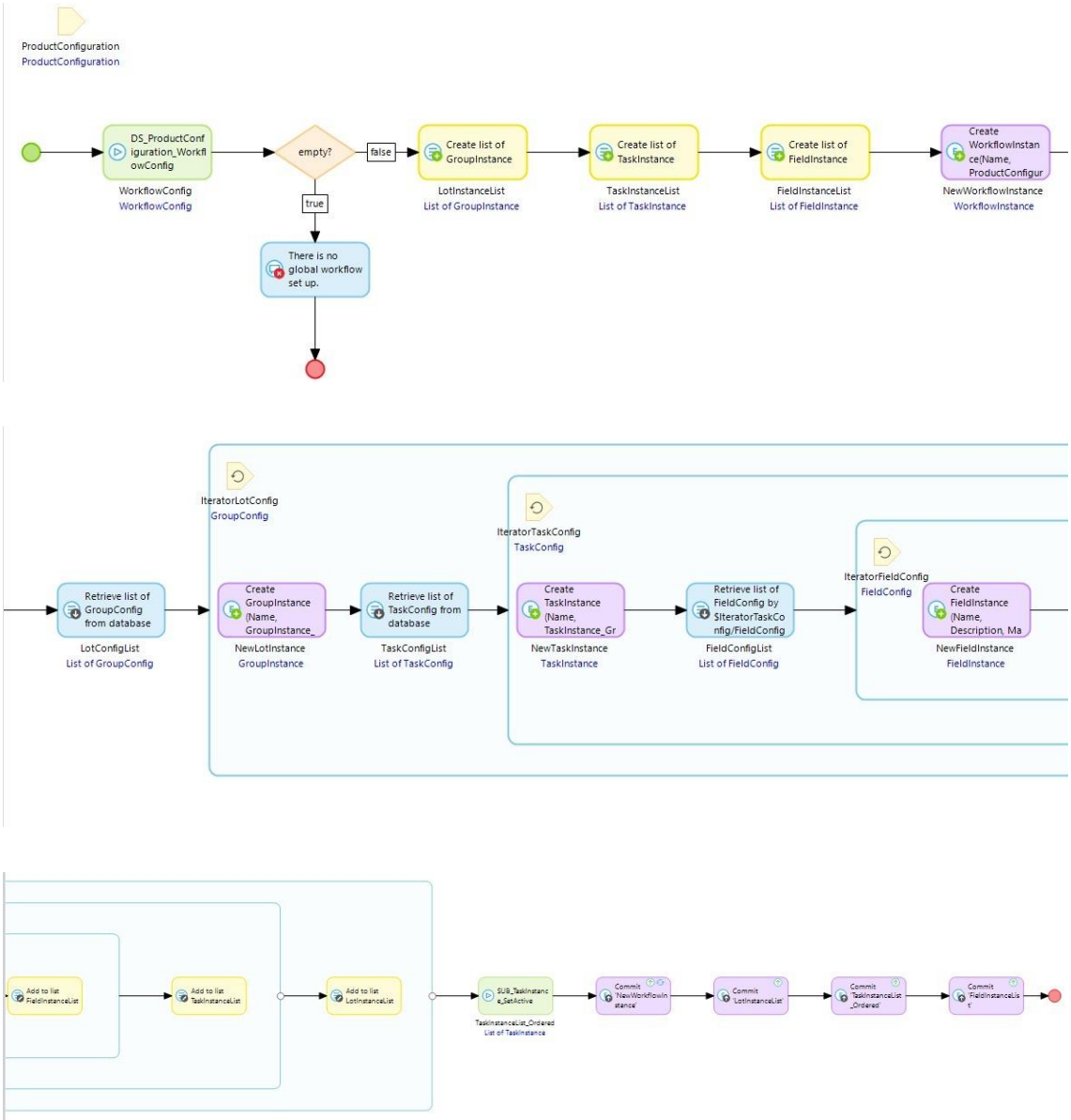


Figura 36. *Sub-Microflow* para atribuir um fluxo de trabalho a nova configuração

O fluxo de submissão de uma nova solicitação inicia-se com a verificação a existência de um fluxo de trabalho para a região do usuário em questão. Dentro desta verificação no início temos

uma outra verificação em um outro *Sub-Microflow*, onde é feita uma varredura na base de dado para ver se a região do usuário já existe. Em caso negativo essa região é criada na base de dados. Já em caso afirmativo, essa região é passada e é feita uma busca por um fluxo de trabalho já pré-definido para a região em que o usuário se encontra. Em seguida, é realizada a verificação de cada campo presente no formulário da configuração. Os próximos passos do *Microflow* consistem em verificações adicionais para garantir que a solicitação seja única e que as informações fornecidas nos campos estejam em conformidade com os dados presentes na base de dados. Por fim, é feita a submissão dos dados inseridos na base de dados e cria-se um espaço de configuração.

Para o seguimento destes passos, foi necessário compreender as necessidades, os dados inseridos e como deveria ser tratada a informação ao longo do processo de criação da configuração. Todos os passos ilustrados são executados a partido do momento em que o usuário faz a submissão de uma nova configuração, pois sempre que é feita uma nova configuração, deve ser verificado qual fluxo de trabalho é o determinado para a região do usuário, sendo que, para cada região, é criado um fluxo de trabalho para associar a uma configuração. Na Figura 37 ilustra-se a página onde é feito o preenchimento dos dados necessários para gerar uma nova configuração.

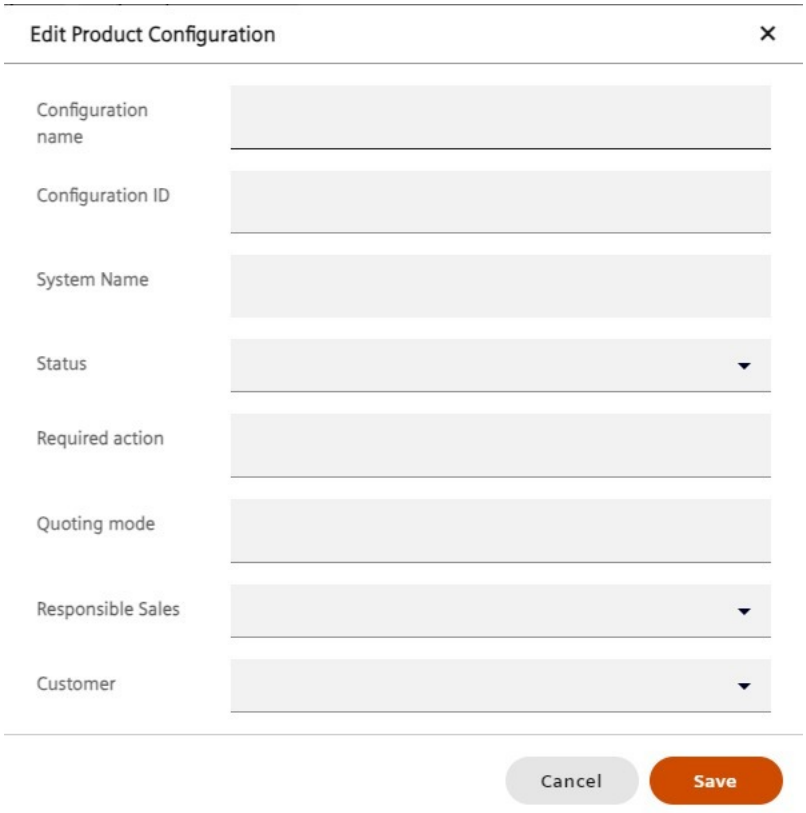


Figura 37. *Layout* do usuário para criação de uma nova configuração para um produto

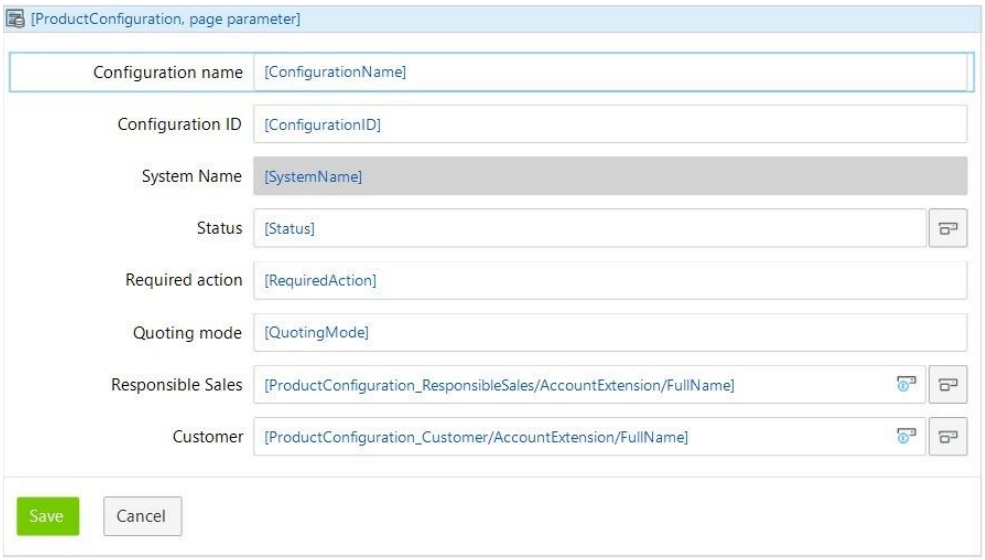


Figura 38. Estrutura página de criação de configuração

Para a página de visualização das solicitações foi utlizado dentro do Mendix um *Data grid 2* como fonte de dados a entidade `ProductConfiguration`, buscando a lista de todos os elementos contidos dentro da entidade, ou seja, todas as solicitações de configurações realizadas. Nesta página é possível visualizar todas configurações da base de dados e filtrar por três possibilidades a tabela, sendo:

- `All configurations` – todas as configurações;
- `My Region` – Sendo um filtro pela região do usuário autenticado;
- `My Responsabilities` – filtro por responsabilidade.

Para esta página, a escolha pelo *Widget* em questão deu-se devido a página de visualização ter sido desenhada por um designer, onde foi feito um esboço de como gostariam que fosse feita a página. Por este motivo, o *Data grid 2* foi aplicado no desenvolvimento da página, devido a sua maior capacidade de executar determinadas ações de forma mais direta, como por exemplo ter a opção de esconder algumas colunas de informações, onde é possível por trás já predefinir quais serão escondidas de forma padrão, com um ícone para o usuário poder visualizar se assim desejar. Além desta opção, o *Data grid 2* não possui os filtros como padrão, sendo assim é necessário fazer cada um dos filtros individualmente. Com essa opção, os filtros acabam por ter uma capacidade maior de filtrar, tendo mais opções como um intervalo de datas, o que também pode ser escolhido pelo usuário final quando for realizar a filtragem.

Conforme citado anteriormente o esboço da página foi feito por um designer, e neste desenho consiste que em cada uma das linhas da tabela, cada uma das configurações teria um botão *Open* que iria para uma outra página, a qual continha os dados específicos de cada

configuração o seu fluxo de trabalho, um *chat* para comunicação e atualização sobre a configuração. Também possui diversas opções de comunicação por e-mail com os responsáveis ou com os clientes, realizar a retirada da configuração e anexar documentos.

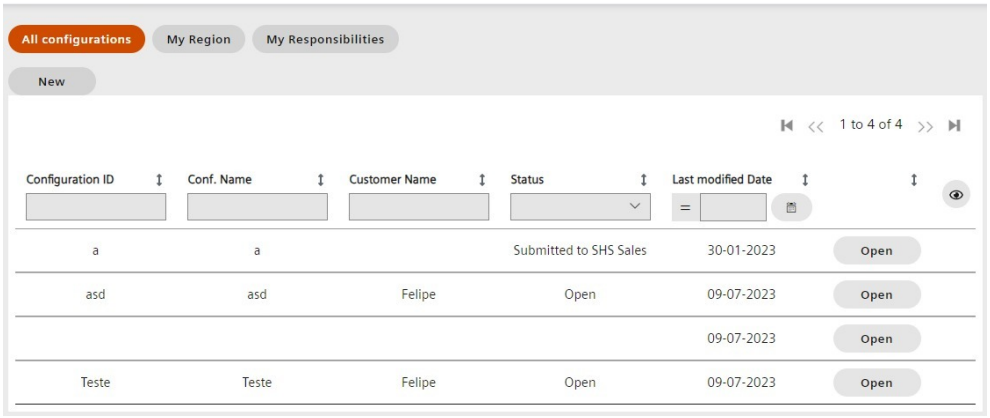


Figura 39. *Layout* do usuário de visualização das configurações

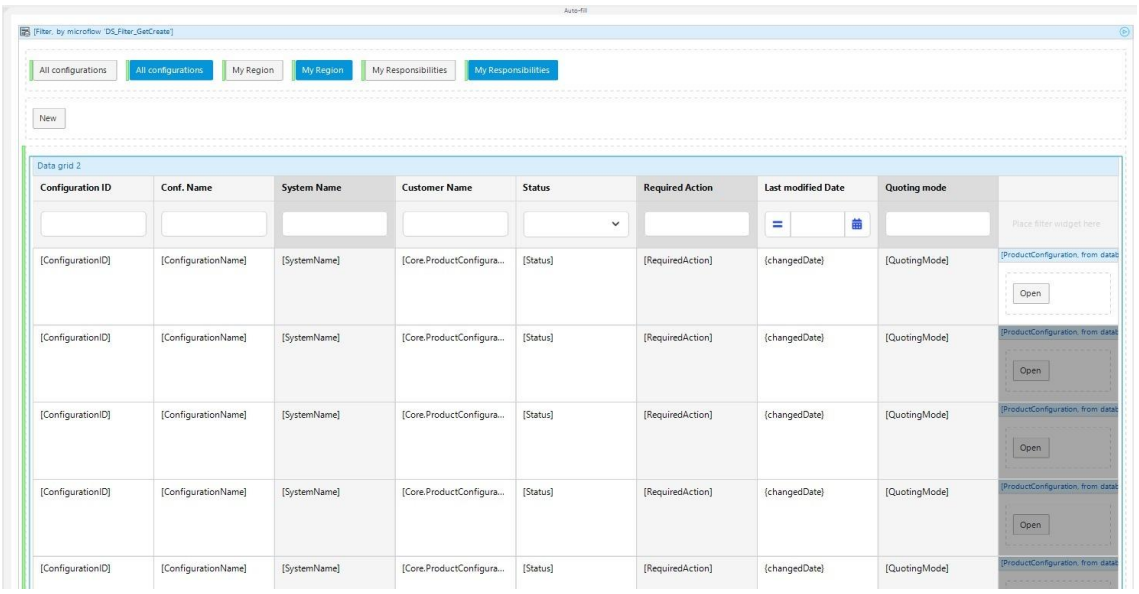


Figura 40. Estrutura de programação da página de visualização das configurações

Dado o *layout* e o esboço do designer, foi possível alcançar o resultado mostrado acima pelas imagens.

Para cada configuração foi introduzido um botão *Open*, onde é recebido o parâmetro com os dados da configuração e é passado para uma outra página destinada para visualizar com uma amplitude maior a configuração do produto. A página em questão possui *layout* também definido pelo designer e com funcionalidades que inicialmente foram pensadas somente para esta página, mas após a implementação foi decidido que poderia ser útil em outras partes da aplicação web. O *chat* foi um dos desafios mais interessantes, juntamente com a possibilidade

de criar um fluxo de trabalho manualmente dentro da aplicação, ambos foram complexos e com muitos detalhes para serem feitos.

Na Figura 41 podemos visualizar como ficou o *layout* final da página, onde temos o *chat* e os passos do fluxo de trabalho desenvolvidos, tornando-se possível que o próprio usuário crie os campos e as etapas do fluxo de acordo com a sua necessidade.

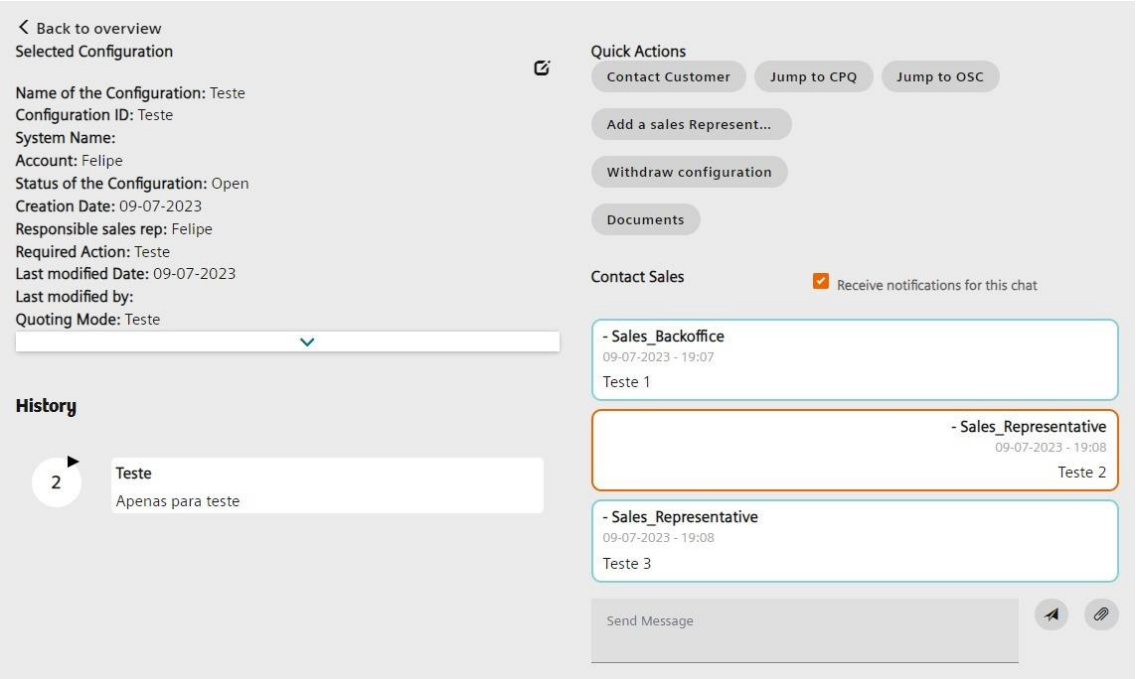


Figura 41. *Layout* da página do usuário para visualização dos dados da configuração

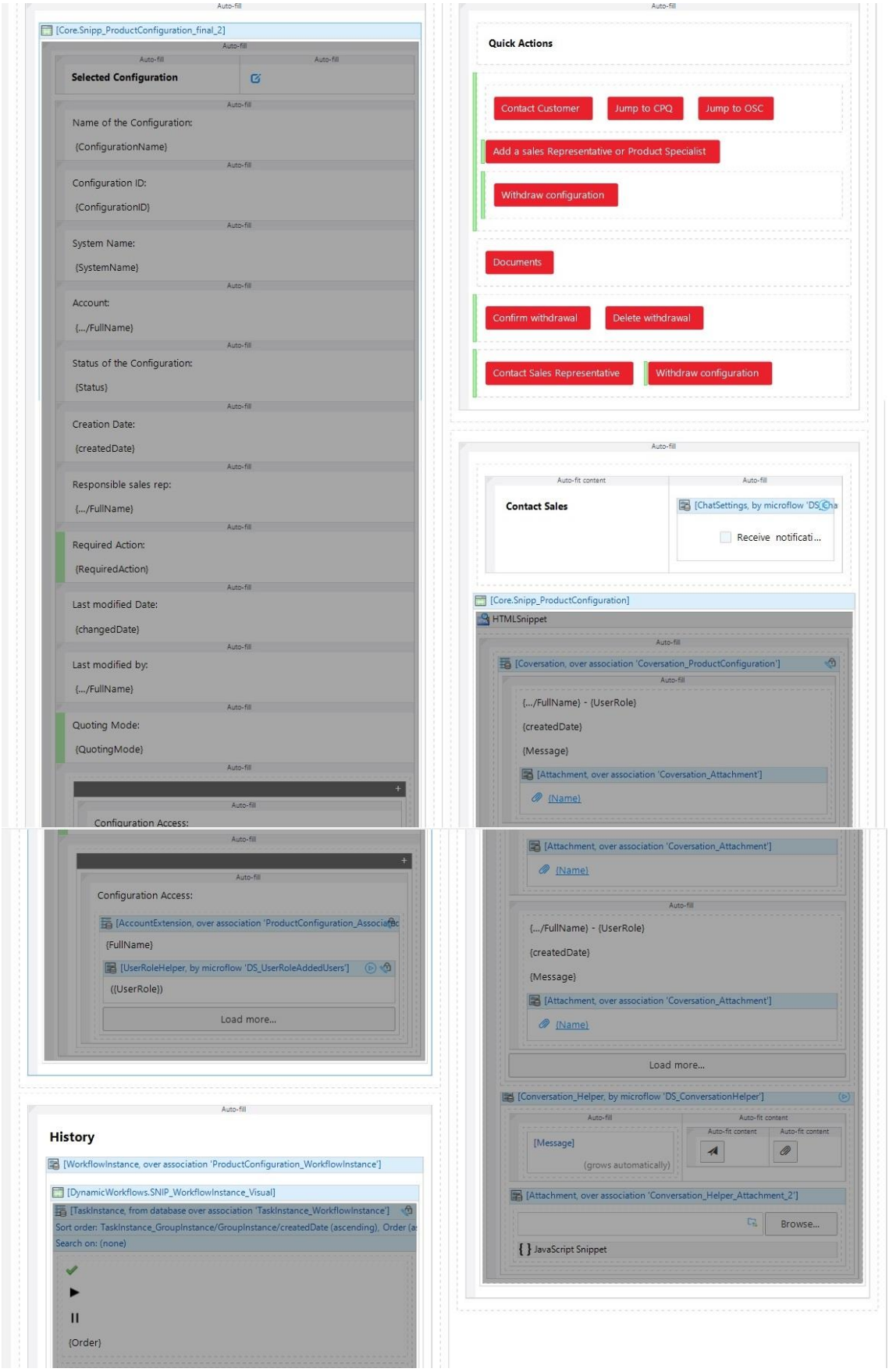


Figura 42. Estrutura de programação da página de visualização das informações da configuração e suas respetivas ações

Através da Figura 42 é possível perceber uma diferença entre o *layout* de um usuário com um perfil de acesso específico e a estrutura por trás de alguns botões que não aparecem. Isso se dá devido as condições de visibilidade que foram aplicadas seguindo as necessidades do cliente, e também através das elaborações dos testes durante o desenvolvimento, onde foram criadas situações para analisar se faria sentido aquele perfil visualizar ou executar determinada atividade ou ação dentro da aplicação.

Na Figura 42 em que é representada a estrutura da página é possível perceber algumas marcações a verde nas laterais de alguns blocos. Essas marcações são indicativas de que existem regras sendo aplicadas dentro daquele bloco, como por exemplo de visualizar ou até mesmo na dependência de que uma variável seja alterada para que a informação ou ação fique aparente na tela final.

Em suma, este Caso de Estudo explorou o potencial da plataforma Mendix para a criação de configurações e a implementação de um *live chat* e de um *Workflow* configurável pelo usuário. Através do uso dos *Microflows*, foi possível desenvolver um sistema flexível e personalizável, permitindo que os usuários definissem suas próprias configurações e adaptassem o fluxo de trabalho às suas necessidades específicas.

A plataforma Mendix proporcionou uma abordagem intuitiva e eficiente para o desenvolvimento do sistema, utilizando sua interface de arrastar e soltar para criar *Microflows* e configurar as funcionalidades de *live chat*. Isso possibilitou uma implementação rápida e simplificada, reduzindo significativamente o tempo de desenvolvimento.

A criação de configurações pelo usuário ofereceu uma experiência personalizada, permitindo que cada usuário ajustasse o sistema de acordo com suas preferências. Isso aumentou a adaptabilidade do sistema e proporcionou uma maior satisfação aos usuários finais.

Além disso, a capacidade de configurar o *Workflow* permitiu que os usuários definissem os passos e a sequência das interações, adaptando-o às necessidades específicas de cada caso de uso. Isso tornou o sistema mais flexível e adequado para diferentes cenários de atendimento ao cliente.

No geral, este Caso de Estudo demonstrou o potencial da plataforma Mendix para a criação de sistemas com configurações personalizáveis e *Workflows* adaptáveis pelo usuário. Essa abordagem oferece uma solução poderosa e flexível para o desenvolvimento de aplicativos, permitindo uma maior agilidade e adaptabilidade às necessidades do usuário final. Com suas capacidades intuitivas e eficientes, a plataforma Mendix se mostra como uma excelente opção para a implementação de soluções personalizadas em diversas áreas de negócio.

5.4 ANÁLISE DA PLATAFORMA UTILIZADA NOS CASOS DE ESTUDO

Os casos de estudo desenvolvidos neste capítulo do presente relatório de estágio ilustram claramente as capacidades abrangentes e as inúmeras possibilidades oferecidas pela plataforma Mendix no desenvolvimento rápido de aplicações web. Ao utilizar o Mendix como ferramenta de desenvolvimento, foram identificados vários pontos positivos que contribuem para um ciclo de desenvolvimento mais ágil e eficiente.

Primeiramente, a plataforma Mendix permite a criação de aplicações web de forma intuitiva e visual, por meio de sua interface de arrastar e soltar. Isso agiliza significativamente o processo de desenvolvimento, uma vez que os desenvolvedores podem criar modelos de domínio, definir fluxos de trabalho automatizados e configurar interfaces gráficas personalizadas sem a necessidade de escrever extensos trechos de código manualmente.

Além disso, a ampla gama de recursos pré-construídos disponíveis no Mendix, como integração de dados, autenticação e visualização de dados, facilita o desenvolvimento de aplicações complexas. Essas funcionalidades prontas para uso eliminam a necessidade de desenvolvimento personalizado e permitem que os desenvolvedores se concentrem nas necessidades específicas da aplicação, acelerando o processo de entrega.

Outro aspeto positivo é a abordagem colaborativa e centrada no usuário oferecida pelo Mendix. A plataforma possibilita a colaboração entre equipas multidisciplinares, incluindo desenvolvedores, designers e usuários finais. Isso promove uma maior compreensão das necessidades do usuário e uma melhor integração entre as expectativas do projeto e sua implementação, resultando em soluções mais eficazes e alinhadas com as demandas do mercado.

Por fim, o Mendix se destaca por sua capacidade de permitir o desenvolvimento rápido de aplicações web de alta qualidade. Ao simplificar o processo de desenvolvimento e fornecer recursos poderosos, a plataforma Mendix ajuda a reduzir o tempo de lançamento no mercado, possibilitando que as empresas se adaptem rapidamente às mudanças e às necessidades dos usuários.

Em resumo, a utilização do Mendix nos casos de estudo demonstrou suas capacidades, possibilidades e pontos positivos no contexto do desenvolvimento rápido de aplicações web. Sua interface intuitiva, recursos pré-construídos, abordagem colaborativa e foco no usuário são elementos-chave que permitem o desenvolvimento eficiente e ágil de soluções web de alta qualidade, impulsionando o sucesso das organizações no ambiente digital.

6

AVALIAÇÃO CRÍTICA AO ESTÁGIO

O estágio realizado na empresa Atos, cujas principais atividades foram o desenvolvimento de aplicações usando a plataforma de *low-code* Mendix e a metodologia Scrum, proporcionou uma experiência enriquecedora no desenvolvimento de software. A combinação dessas abordagens visava acelerar o processo de desenvolvimento e promover uma colaboração eficaz entre a equipa, resultando em benefícios e desafios identificados ao longo do estágio.

Um dos pontos positivos observados foi a facilidade de criação de aplicações proporcionada pela plataforma Mendix. A interface intuitiva e amigável permite que estagiários em início de carreira, independentemente do conhecimento técnico, possam criar aplicações através do recurso de arrastar e soltar componentes. Isso facilita o desenvolvimento e torna as aplicações acessíveis a uma variedade de perfis de desenvolvedores.

Outro ponto positivo é a disponibilidade de recursos poderosos oferecidos pela plataforma Mendix. A capacidade de integração com sistemas externos e a geração de aplicativos nativos para dispositivos móveis ampliam as possibilidades de criação e permitem o desenvolvimento de soluções abrangentes e adaptadas às necessidades específicas do negócio. Esses recursos adicionais fornecem uma vantagem competitiva na entrega de soluções avançadas para as empresas.

No entanto, algumas limitações foram identificadas durante o estágio. A plataforma Mendix pode ter restrições em termos de flexibilidade e personalização. Em determinados casos, as necessidades específicas do projeto podem exigir customizações além das capacidades da plataforma, o que pode levar à busca por soluções alternativas ou ao desenvolvimento de código personalizado, através dos métodos mais tradicionais como Java, JavaScript, CSS e

HTML. Essa necessidade de recorrer a soluções externas pode impactar a agilidade e a velocidade do processo de desenvolvimento.

Além disso, foi observado um desafio em relação ao equilíbrio entre a velocidade de entrega e a qualidade do código produzido. Embora o desenvolvimento em *low-code* permita uma rápida prototipação e iteração, é essencial garantir a aderência às melhores práticas de desenvolvimento e a qualidade do código gerado. Isso é crucial para garantir a manutenibilidade e a escalabilidade das aplicações desenvolvidas, bem como a satisfação do cliente a longo prazo.

Apesar dessas limitações e desafios, o estágio em *low-code* utilizando Mendix e a metodologia Scrum proporcionou uma experiência valiosa no desenvolvimento de software. A combinação dessas abordagens permitiu um processo de desenvolvimento mais rápido, colaborativo e adaptável às necessidades em constante evolução. O estágio também ofereceu a oportunidade de desenvolver habilidades práticas e se familiarizar com abordagens modernas de desenvolvimento de software.

Em suma, o estágio em *low-code* utilizando Mendix e Scrum apresentou pontos positivos e desafios a serem considerados. A plataforma Mendix oferece uma interface amigável e recursos poderosos, enquanto a metodologia Scrum proporciona uma estrutura ágil para a gestão de projetos. No entanto, é importante estar ciente das limitações da plataforma e garantir a aderência às melhores práticas de desenvolvimento para obter sucesso no desenvolvimento de aplicações de alta qualidade. O estágio foi uma oportunidade valiosa para aprimorar habilidades e adquirir experiência em abordagens modernas de desenvolvimento de software.

7

CONCLUSÃO

O presente relatório abordou um conjunto de tópicos essenciais para o desenvolvimento de software no mercado atual, incluindo metodologias ágeis como o Scrum, a abordagem *low-code*, a plataforma Mendix e o desenvolvimento web. Além disso, explorou a aplicação dessas abordagens em projetos de *outsourcing*, onde as necessidades do mercado exigem soluções rápidas, eficientes e de alta qualidade. Isto tudo no contexto do estágio do autor na empresa Atos.

Foi evidente que a utilização do Scrum e das metodologias ágeis trouxe inúmeros benefícios para os projetos de desenvolvimento. A divisão do trabalho em *sprints* e a realização de reuniões diárias garantiram uma comunicação efetiva, permitindo que a equipa se adaptasse rapidamente às mudanças e requisitos emergentes. Essa abordagem colaborativa e ágil foi fundamental para alcançar os objetivos dos projetos de forma eficiente.

A abordagem *low-code* utilizando a plataforma Mendix também desempenhou um papel crucial no desenvolvimento de aplicações web. A facilidade de criação, com recursos como arrastar e soltar componentes, permitiu um desenvolvimento rápido e simplificado. Além disso, a plataforma Mendix ofereceu recursos poderosos, como a integração com sistemas externos e a geração de aplicativos nativos, aumentando a flexibilidade e a adaptabilidade das soluções desenvolvidas.

No contexto dos projetos de *outsourcing*, essas abordagens mostraram-se particularmente eficazes. A redução dos custos, do tempo de desenvolvimento e dos *bugs* nas aplicações foram resultados diretos da utilização das metodologias ágeis, da abordagem *low-code* e da plataforma Mendix. A entrega rápida e eficiente das soluções aos clientes foi um fator chave para atender às demandas do mercado atual.

Além disso, o presente relatório demonstrou a necessidade e a importância do desenvolvimento rápido de aplicações web no cenário atual. A velocidade de entrega e a capacidade de adaptação às mudanças do mercado são essenciais para garantir a competitividade das empresas. A abordagem *low-code* e a plataforma Mendix demonstraram ser ferramentas valiosas nesse sentido, permitindo que as empresas desenvolvam e implantem rapidamente soluções personalizadas e de alta qualidade.

Em suma, o presente relatório demonstrou que a combinação de metodologias ágeis, a abordagem *low-code* e a utilização da plataforma Mendix são fundamentais para o desenvolvimento de software no mercado atual. Essas abordagens proporcionaram uma resposta ágil às necessidades do mercado, resultando em redução de custos, tempo de desenvolvimento e *bugs*, além de possibilitar o desenvolvimento rápido de aplicações web. Essas conclusões reforçam a relevância dessas abordagens e apontam para uma direção promissora no desenvolvimento de software nos próximos anos.

REFERENCIAS

[1]“Company Profile - Atos.” Accessed: Mar. 05, 2023. [Online]. Available: https://atos.net/en/company-profile

[2]D. C. Schmidt, “Model-driven engineering,” *Computer (Long Beach Calif)*, vol. 39, no. 2, pp. 25–31, Feb. 2006, doi: 10.1109/MC.2006.58.

[3]“Classification of Development Frameworks for Enterprise Apps - DZone.” Accessed: Mar. 02, 2023. [Online]. Available: https://dzone.com/articles/classification-of-development-frameworks-for-enter

[4]“API Code & Client Generator | Swagger Codegen.” Accessed: Sep. 12, 2023. [Online]. Available: https://swagger.io/tools/swagger-codegen/

[5]“Feature-rich Java Framework for Business Application Development.” Accessed: Sep. 12, 2023. [Online]. Available: https://www.jmix.io/framework/

[6]“What is PaaS? | Intro to Platform as a Service App Development | Mendix.” Accessed: Sep. 13, 2023. [Online]. Available: https://www.mendix.com/application-platform-as-a-service/which-is-the-right-apaas-for-you/

[7]B. Selic, “The pragmatics of model-driven development,” *IEEE Softw*, vol. 20, no. 5, pp. 19–25, Sep. 2003, doi: 10.1109/MS.2003.1231146.

[8]R. G. Pettit, N. Mezcciani, and J. Fant, “On the needs and challenges of model-based engineering for spaceflight software systems,” *Proceedings - IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, ISORC 2014*, pp. 25–31, Sep. 2014, doi: 10.1109/ISORC.2014.13.

[9]“Transforming software development: an MDA road map | IEEE Journals & Magazine | IEEE Xplore.” Accessed: Jul. 15, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/1510571

[10] M. Brambilla, J. Cabot, and M. Wimmer, “Model-Driven Software Engineering in Practice,” 2017, doi: 10.1007/978-3-031-02549-5.

[11] “Definition of a Domain-Specific Modelling Language - Marbella International University Centre.” Accessed: Sep. 25, 2023. [Online]. Available: https://miuc.org/definition-of-a-domain-specific-modelling-language/

[12] A. Kriouile, N. Addamssiri, and T. Gadi, “An MDA Method for Automatic Transformation of Models from CIM to PIM,” *http://www.sciencepublishinggroup.com*, vol. 4, no. 1, p. 1, Mar. 2015, doi: 10.11648/J.AJSEA.20150401.11.

[13] “Desenvolvimento low-code ganha popularidade entre profissionais e organizações, diz pesquisa - IT Forum.” Accessed: Sep. 13, 2023. [Online]. Available: https://itforum.com.br/noticias/desenvolvimento-low-code-ganha-popularidade-entre-profissionais-e-organizacoes-diz-pesquisa/

[14] “The State of Low-Code 2021: A Look Back, The Light Ahead | Mendix.” Accessed: Jul. 15, 2023. [Online]. Available: https://www.mendix.com/resources/the-state-of-low-code-report/

[15] A. Sahay, A. Indamutsa, D. Di Ruscio, and A. Pierantonio, “Supporting the understanding and comparison of low-code development platforms,” *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*, pp. 171–178, Aug. 2020, doi: 10.1109/SEAA51224.2020.00036.

[16] A. Sahay, A. Indamutsa, D. Di Ruscio, and A. Pierantonio, “Supporting the understanding and comparison of low-code development platforms,” *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*, pp. 171–178, Aug. 2020, doi: 10.1109/SEAA51224.2020.00036.

[17] “Supporting the understanding and comparison of low-code development platforms | IEEE Conference Publication | IEEE Xplore.” Accessed: Sep. 15, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9226356

[18] “Download The Forrester Wave: Low-Code Development Platforms 2021.” Accessed: Mar. 02, 2023. [Online]. Available: https://www.mendix.com/resources/forrester-low-code-platform-wave/

[19] “Enterprise Architecture Platform - Modules & Services | Mendix Evaluation Guide.” Accessed: Mar. 02, 2023. [Online]. Available: https://www.mendix.com/evaluation-guide/enterprise-capabilities/platform-architecture/

[20] “DevOps, Continuous Integration Tools in Mendix | Mendix Evaluation Guide.” Accessed: Mar. 02, 2023. [Online]. Available: https://www.mendix.com/evaluation-guide/app-lifecycle/devops/

[21] "DevOps Tools in Mendix Overview | Mendix Evaluation Guide." Accessed: Mar. 02, 2023. [Online]. Available: https://www.mendix.com/evaluation-guide/app-lifecycle/devops-overview/

[22] "Gartner Reprint." Accessed: Mar. 02, 2023. [Online]. Available: https://www.gartner.com/doc/reprints?id=1-27IIPKYV&ct=210923&st=sb

[23] "Develop Apps Fast | OutSystems Low-Code Platform | OutSystems." Accessed: Mar. 02, 2023. [Online]. Available: https://www.outsystems.com/low-code-platform/

[24] "From Development to Deployment in Minutes | OutSystems Low-Code Platform | OutSystems." Accessed: Mar. 02, 2023. [Online]. Available: https://www.outsystems.com/low-code-platform/development-deployment/

[25] "The Forrester Wave™: Real-Time Interaction Management, Q2 2022." Accessed: Mar. 02, 2023. [Online]. Available: https://reprints2.forrester.com/#/assets/2/54/RES176354/report

[26] "Pega Express: A collaborative approach to delivering solutions | Pega." Accessed: Mar. 02, 2023. [Online]. Available: https://www.pega.com/services/consulting/pega-express

[27] "Intelligent automation capabilities on the Pega Platform | Pega." Accessed: Mar. 02, 2023. [Online]. Available: https://www.pega.com/products/platform/intelligent-automation

[28] "Mendix Announces Studio and Studio Pro; No-Code and Low-Code Visual Development Environments | Mendix." Accessed: Mar. 02, 2023. [Online]. Available: https://www.mendix.com/press/mendix-announces-studio-and-studio-pro-no-code-and-low-code-visual-development-environments/

[29] "Mendix vs OutSystems vs Pega Platform Comparison | SaaSworthy.com." Accessed: Mar. 02, 2023. [Online]. Available: https://www.saasworthy.com/compare/mendix-vs-outsystems-vs-pega-platform?pIds=8924,10134,38206

[30] "Mendix vs. OutSystems vs. Pega Platform Comparison." Accessed: Mar. 02, 2023. [Online]. Available: https://sourceforge.net/software/compare/Mendix-vs-OutSystems-vs-Pega-Platform/

[31] W. W. Rovce, "MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS".

[32] B. W. Boehm, "A Spiral Model of Software Development and Enhancement," *Computer (Long Beach Calif)*, vol. 21, no. 5, pp. 61–72, 1988, doi: 10.1109/2.59.

[33] “O Modelo em Espiral de Boehm. Criado por Barry Boehm em 1988, é uma… | by Ricardo Dias | Contexto Delimitado | Medium.” Accessed: Mar. 02, 2023. [Online]. Available: https://medium.com/contexto-delimitado/o-modelo-em-espiral-de-boehm-ed1d85b7df

[34] S. Vasudevan and D. Wilemon, “Rapid application development: major issues and lessons learned,” p. 484, Nov. 2002, doi: 10.1109/PICMET.1997.653483.

[35] “Rapid application development | WorldCat.org.” Accessed: Sep. 16, 2023. [Online]. Available: https://www.worldcat.org/title/Rapid-application-development/oclc/645818373

[36] “What is Rapid Application Development? i,” 1997.

[37] “RAD: conheça o desenvolvimento rápido de aplicação! | Insights para te ajudar na carreira em tecnologia | Blog da Trybe.” Accessed: Mar. 02, 2023. [Online]. Available: https://blog.betrybe.com/tecnologia/rad/

[38] “Como utilizar a metodologia Kanban no desenvolvimento de softwares?” Accessed: Mar. 02, 2023. [Online]. Available: https://blog.cronapp.io/como-utilizar-a-metodologia-kanban-no-desenvolvimento-de-softwares/

[39] “O Quadro Kanban – Semeru Blog.” Accessed: Sep. 16, 2023. [Online]. Available: https://www.semeru.com.br/blog/o-quadro-kanban/

[40] H. Gong, B. Liu, and D. Shao, “A Simulation Model of Kanban Software Process,” *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, vol. 2017-December, pp. 745–746, Mar. 2018, doi: 10.1109/APSEC.2017.96.

[41] “Scrum Guide | Scrum Guides.” Accessed: Mar. 02, 2023. [Online]. Available: https://scrumguides.org/scrum-guide.html#scrum-team

[42] “Scrum Master Kursus som klassekursus (scrum.org) med elæring som forberedelse.” Accessed: Mar. 02, 2023. [Online]. Available: https://metier.dk/kursus-uddannelse/scrum-master/scrum-master-kursus/

[43] “Metodologias ágil x tradicional: Quais as diferenças? | Blog TreinaWeb.” Accessed: Sep. 16, 2023. [Online]. Available: https://www.treinaweb.com.br/blog/metodologias-agil-x-tradicional-quais-as-diferencas

[44] “Mendix Runtime | Mendix Documentation.” Accessed: May 20, 2023. [Online]. Available: https://docs.mendix.com/refguide/runtime/