INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Migração de soluções Low-code PowerApps para Flutter

JOÃO MANUEL GOMES DOS SANTOS
Outubro de 2022

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Migração de soluções *Low-code* – PowerApps para Flutter

João Manuel Gomes dos Santos

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: Alexandre Manuel Tavares Bragança

Porto, Outubro 2022

Resumo

Atualmente, existe um tipo de desenvolvimento de software denominado de *Low-code*. Este permite desenvolver software sem escrita de código, nem conhecimentos de programação ou mesmo de arquitetura de software. Deste modo, os profissionais sem conhecimentos de TI, especialistas de um domínio de negócio em questão, têm a capacidade de desenvolver as suas próprias aplicações, tendo a vantagem de conhecer melhor o negócio que um desenvolvedor de software.

As aplicações desenvolvidas desta forma possuem desvantagens, tal como, não serem tão personalizáveis relativamente a outro tipo de aplicações. Consequentemente, por vezes, torna-se impossível de responder aos requisitos do cliente. O facto de estas aplicações serem apenas possíveis de ser executadas sobre a plataforma de quem disponibiliza os sistemas de desenvolvimento deste tipo de soluções, como o caso das PowerApps, obriga, por vezes, ao licenciamento das próprias plataformas, tornando, assim, o exposto uma desvantagem.

Desta maneira, com o intuito de responder às solicitações dos clientes, cria-se a necessidade de fazer a migração para uma aplicação nativa. Esta migração pode ser demorada, visto que é desenvolvida uma aplicação do início.

Este projeto tem como objetivo automatizar o processo de migração do desenvolvimento em Powerapps para o desenvolvimento tradicional, compreendendo aquilo que é possível reaproveitar de uma aplicação desenvolvida em Powerapps, e gerar uma aplicação nativa, o mais idêntica possível à anterior. Isto irá permitir que as aplicações em questão deixem de estar dependentes da plataforma Powerapps e possam então ser mantidas e estendidas sem os possíveis constrangimentos dessa plataforma.

Neste documento é descrito o estudo das áreas de *Low-code* e desenvolvimento móvel cross-platform, a segunda por ser uma forma de desenvolver aplicações uma vez e poder executá-las em vários tipos de plataformas como android e ios. Foram ainda estudadas as alternativas para a construção de uma solução capaz de migrar PowerApps para aplicações nativas, de uma forma automatizada. Assim, aproveita-se a rapidez do desenvolvimento *low-code* e a melhor performance de uma aplicação nativa.

Este projeto foi desenvolvido no contexto da empresa Devscope, que desenvolve apps em Powerapps para vários clientes, como por exemplo na área da saúde.

Foi possível implementar uma prova de conceito capaz de migrar Powerapps para aplicações nativas, o que comprova que é possível automatizar este tipo de processos tornando-os muito mais rápidos, aumentando a produtividade dos desenvolvedores de software.

Palavras-chave: *Low-code*, *Cross-platform mobile*, Migração de Software

Abstract

Nowadays there is a type of software development named Low code. This type of development enables to develop software without writing any code neither have knowledge in programming or software architecture, giving the capacity to some business professionals to develop their apps having the advantage of knowing better their business than software developers.

The apps developed this way, however, have some disadvantages, one is that they are not customable like native solutions, and then it makes impossible to answer to the customers' requirements. Another factor it is the fact that they can only be executed in the platform that they were developed, as is the case of PowerApps, and it obliges to licentiate the platform.

In this way, to answer to the customer's requirements, it is necessary to migrate to a native app. This migration can take a lot of time as it is developed an app from zero.

This project has the objective to automate the process of migrating de development of a Powerapp to the traditional development, understanding what is possible to reuse from an app developed in Powerapps, and generate a native app that is equal to the Powerapp. This allows the apps to become independent from the platform and could be maintained and extended without the limitations from the platform.

In this document it is described the study of the areas of low-code development, mobile cross-platform development, the second one it is because this development allows developers to write an app a run in more then one platform. It was studied the alternatives for the construction of a solution capable of migrating apps from Powerapps into native apps in an automated way. Thus, we can use the fast development from low-code and the best performance form native apps.

It was possible to implement a concept proof capable of migrating apps from Powerapps to native apps, what proofs that it is possible to automate this kind of processes making them fasters, increasing the productivity of the software developers.

Keywords: Low-code, mobile cross-platform development, software migration

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos

Lista de Acrónimos

AHP	Analytic Hierarchy Process
API	Application Programming Interface
LCDP	Low-Code Development Platform
PaaS	Platform as a Service

1 Introdução

Neste capítulo, apresenta-se o contexto da dissertação, o problema identificado, os objetivos a atingir para responder ao problema, a abordagem preconizada para cumprir com os objetivos traçados e por último a estrutura do documento.

1.1 Contexto

Atualmente, existe um tipo de desenvolvimento de software denominado de *Low-code*. Este permite desenvolver software sem escrita de código, nem conhecimentos de programação ou mesmo de arquitetura de software (Sahay et al., 2020). Deste modo, os profissionais, de um negócio em questão, têm a capacidade de desenvolver as suas próprias aplicações, tendo a vantagem de conhecer melhor o negócio que um desenvolvedor de software.

Ainda que estes profissionais não se sintam confortáveis em serem eles próprios a desenvolver as aplicações, as plataformas *Low-code* permitem, aos desenvolvedores de software, desenvolver soluções mais rápido do que as formas tradicionais. Sendo o desenvolvimento mais rápido, recebe-se também mais depressa o feedback dos utilizadores, o que permite a aproximação do utilizador com o desenvolvimento de software e assim facilita a perceção do que traz realmente valor ao cliente.

A dissertação foi desenvolvida no contexto de um projeto na empresa DevScope, que utiliza uma plataforma *Low-code* chamada PowerApps (Pearson et al., 2020), e tem já relatos de desenvolvimento de aplicações, para clientes, num curto intervalo de tempo com bastante sucesso.

1.2 Problema

As aplicações desenvolvidas desta forma, possuem desvantagens, tal como, não serem tão personalizáveis relativamente a outro tipo de aplicações. Consequentemente, por vezes,

torna-se difícil de responder aos requisitos do cliente. O facto de estas aplicações serem apenas possíveis de ser executadas sobre a plataforma de quem disponibiliza os sistemas de desenvolvimento deste tipo de aplicações, como o caso das PowerApps, obriga por vezes, ao licenciamento das próprias plataformas, tornando assim, o exposto uma desvantagem.

Por último, as aplicações *Low-code* não têm a mesma performance, nem tão boa usabilidade como as soluções nativas, que são aplicações que são executadas diretamente sobre o sistema operativo (Jobe, 2013). Devido ao facto das aplicações desenvolvidas em PowerApps, terem de estar a correr sobre a plataforma, consomem mais recursos do que estar a correr apenas a aplicação nativa.

Assim, para responder aos pedidos dos clientes, cria-se a necessidade de fazer a migração para uma aplicação nativa. Este tipo de migração pode ser demorado e até mesmo conter algumas falhas. Como este é um processo em que é necessário ser repetido a cada vez que se necessita de passar de uma aplicação *Low-code* para uma tradicional apareceu a conveniência de tornar este processo automatizado. A vantagem seria poupar tempo aos desenvolvedores para realizarem outras tarefas e de apresentar um menor número de falhas neste processo de migração. Para isso foi então necessário migrar os meta-dados que constroem uma Powerapp e também as suas conexões às fontes de dados.

1.3 Objetivos

O objetivo principal desta dissertação é desenvolver uma solução que permita migrar aplicações *Low-code* para aplicações nativas, reduzindo assim o tempo destas migrações e as suas falhas.

Para alcançar o objetivo principal foram apontados os seus objetivos particulares. O primeiro desses objetivos seria sintetizar o conhecimento já existente nas áreas em que o projeto se insere. Sendo as áreas, Plataformas de desenvolvimento de soluções *Low-code*, Ferramentas de geração de código e ainda o Desenvolvimento *Cross Platform Mobile*, este último para termos a vantagem de implementar as aplicações geradas uma vez e podermos executá-las em vários tipos de plataformas diferentes.

Pretende-se, também, desenhar um processo que permita criar um modelo com toda a informação necessária para se poder migrar aplicações desenvolvidas em Powerapps.

Outro objetivo será desenvolver um processo capaz de gerar uma aplicação nativa a partir de modelos de Powerapps, sendo a aplicação gerada o mais idêntica possível à aplicação desenvolvida em Powerapps. Sendo que neste caso o foco do projeto será sobre as aplicações Canvas, que é um tipo de aplicações de Powerapps, sendo o mais usado na empresa em que se desenvolveu este projeto..

Apontou-se também como objetivo realizar os testes necessários à validação da solução desenvolvida, através de testes de integração entre o componente de extração do modelo de

powerapps e o componente de geração de código. Serão também realizados testes de aceitação definidos pelas partes interessadas deste projeto.

Por último será feita a avaliação da solução através dos resultados obtidos, sendo que o resultado esperado é gerar uma aplicação nativa com a componente gráfica igual à aplicação de origem, com o mesmo comportamento e com as mesmas fontes de dados.

1.4 Abordagem

A abordagem preconizada para solucionar este problema será de uma implementação incremental para que as partes interessadas avaliem a solução em tempo útil.

Pretende-se perceber o que a plataforma *Low-Code* disponibiliza relativamente às aplicações que nela são desenvolvidas, e como disponibiliza essa informação. De seguida, construir um meta-modelo que represente esses meta-dados que constroem a aplicação em Powerapps, realizando assim uma transformação modelo-para-modelo, sendo o modelo destino mais facilmente trabalhável e com menos ruído desnecessário.

Será ainda desacoplado a componente de extração dos dados da solução *Low-code* da componente de geração da aplicação nativa, com o propósito de serem mais facilmente reutilizadas noutros processos que não este.

Para fazer essa geração de código, será realizada uma transformação modelo-para-código, e então será utilizada uma ferramenta que permita escrever *templates* de ficheiros para poder gerar o código referente à aplicação nativa.

1.5 Estrutura do Documento

Este documento inicia-se com a **Introdução**, onde é apresentado o essencial à compreensão do mesmo.

No capítulo 2, **Estado da Arte,** é exposta a síntese do conhecimento científico e as tecnologias das áreas em que o projeto se insere.

Na **Análise de Valor**, terceiro capítulo, é exibido o valor deste trabalho.

No quarto capítulo, **Análise de Negócio**, são expostos os requisitos da solução.

No capítulo **Desenho da Solução** são apresentadas diferentes alternativas para a atingir a solução desejada.

No capítulo **Construção da Solução** é apresentada a solução implementada.

Na **Avaliação e Experimentação** é descrita a forma como a solução foi avaliada.

Por último, nas **Conclusões** são apresentados os objetivos alcançados e as limitações do projeto.

2 Estado da Arte

Neste capítulo, é apresentado a síntese do conhecimento científico e das tecnologias e ferramentas nas áreas de *Low-code*, geração de código e desenvolvimento *cross-platform* móvel.

2.1 Plataformas de desenvolvimento Low-Code

Plataformas de desenvolvimento *Low-code* são plataformas de software que assentam na *cloud* e permitem que desenvolvedores de diferentes áreas de conhecimento desenvolvam aplicações prontas para produção (Richardson & Rymer, 2016).

Estas aplicações são desenvolvidas sobre princípios de engenharia orientada a modelos e são implantadas em infraestruturas na *cloud*. A geração do código é automática implementando assim aplicações totalmente funcionais (Paul Vincent et al., 2019).

Estas plataformas capitalizam sobre modelos de desenvolvimento na *cloud*, tal como, por exemplo, as *Platforms as a Service* (PaaS), utilizando padrões de desenho e arquiteturais garantindo a eficiência do desenvolvimento, da implantação e ainda da sua manutenção (Richard Paige et al., 2014).

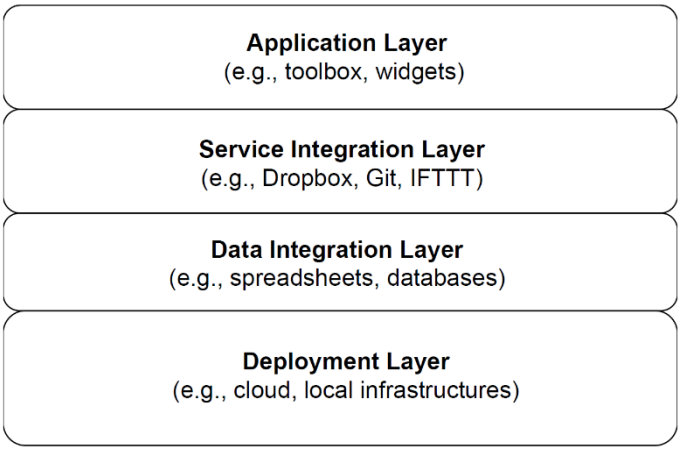


Figura 1 - Arquitetura das LCDP (Sahay et al., 2020)

Ao nível arquitetural as LCDP podem ser divididas em 4 camadas (Sahay et al., 2020). A primeira camada, Camada da aplicação, representa a componente gráfica onde os utilizadores interagem com o sistema e onde especificam as aplicações a desenvolver. É possível encontrar as ferramentas e os *widgets*, utilizados para construir a interface do utilizador da aplicação, tal como a especificação dos mecanismos de autenticação e autorização. A camada da aplicação indica, ainda, o comportamento da aplicação, as fontes de obtenção de dados, como exemplo tabelas, calendários, sensores, ficheiros ou serviços na *cloud* e a forma como estes dados podem ser manipulados.

A camada seguinte, Camada de integração de serviços, é responsável por conectar a aplicação a diferentes serviços através das suas apis e mecanismos de autenticação.

É ainda dedicada uma camada à integração com os dados, Camada de integração de dados, que permite tratar de forma homogénea os dados mesmo que provenientes de fontes heterogéneas.

Dependendo da Plataforma, a aplicação desenvolvida pode ser implantada em infraestruturas dedicadas na *cloud* ou em ambientes *on-premise*, Camada de implantação. Esta camada é ainda responsável pela *containarização* e orquestração das aplicações.

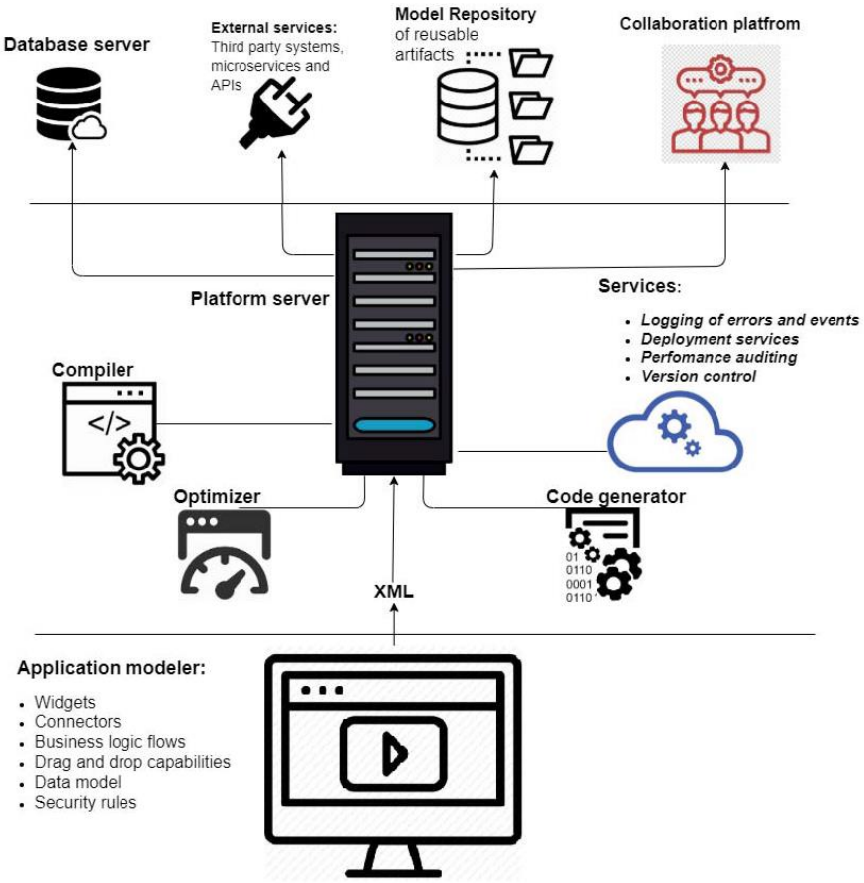


Figura 2 – Componentes de uma LCDP (Sahay et al., 2020)

Os componentes desta arquitetura também podem ser divididos em 3 níveis (Sahay et al., 2020). O primeiro é composto pelo modelador da aplicação, o segundo é responsável pelo lado do servidor e pelas suas funcionalidades e o terceiro é encarregue pelos serviços externos, mas que integram com a plataforma.

As setas presentes na Figura 2 representam as interações entre entidades de diferentes níveis, enquanto as linhas presentes no nível do meio retratam os componentes que dão forma à infraestrutura da plataforma.

Como é visível na figura, a especificação da aplicação a desenvolver é feita mediante um modelador, onde é permitido ao utilizador escolher os *widgets* a usar, bem como a navegação da aplicação e as regras do negócio. É possível ainda correr localmente a aplicação antes da sua implantação para uma perceção do seu funcionamento.
Em seguida, o modelo da aplicação é enviado para a plataforma de *back-end*, onde são realizadas análises e manipulações, consoante os serviços envolvidos, como as bases de dados, micros serviços, conexões com *APIs* e repositórios. É a este nível que ocorre a geração da aplicação completa. Posteriormente a aplicação é testada e pronta para ser implantada na *cloud*.

As bases de dados podem ser tanto SQL como NoSQL, contudo este tipo de decisões ou preocupações de garantir a integridade dos dados e a otimização das *querys* não são entregues aos utilizadores.
No geral, decisões ao nível da arquitetura do sistema não são da responsabilidade dos utilizadores. A plataforma encarrega-se de criar os micros-serviços necessários, orquestrá-los e geri-los sem a intervenção do utilizador.

O utilizador tem a oportunidade de conceber interações com *APIs* externas, mas através de conectores específicos.
Assim, os utilizadores ficam livres da responsabilidade de gerir aspetos como a autenticação, a consistência ao nível da lógica de negócio e a segurança.

As LCDP permitem também o armazenamento de artefactos em repositórios, onde é realizado o controlo de versões.

O desenvolvimento em LCDP pode ser dividido em 5 fases (Sahay et al., 2020), como podemos constatar:

1. Modelação dos dados - Aqui os utilizadores configuram os esquemas dos dados, identificando entidades, as suas relações, restrições e dependências.
2. Definição da interface do utilizador - Nesta fase são desenhadas as páginas e formulários da interface que definem as vistas da aplicação.
3. Especificação da lógica do negócio e *workflows* - Aqui são apontados os fluxos e as regras do negócio.
4. Integração com serviços externos - Nesta fase são especificadas as *APIs*, que interagem com a aplicação.
5. Implantação da aplicação - Por fim, é permitido ao utilizador implantar a sua solução.

No estudo desenvolvido pelo The Forrester Wave (Rymer & Koplowitz, 2019) relativo às plataformas *Low-Code* o Powerapps, Outsystems e Mendix apresentaram-se como das mais relevantes e serão a seguir apresentadas.
Será também apresentado o FlutterFlow visto ser a plataforma mais idêntica a este projeto.
São ainda plataformas importantes na medida em que tem todas um forte componente no desenvolvimento móvel, sendo sobre este que recai o projeto.

2.1.1 Microsoft PowerApps

O Microsoft PowerApps é uma plataforma de desenvolvimento Low-code/no-code que permite criar aplicações de uma forma rápida. Fornece uma coleção de *templates* e apps já desenvolvidas, capazes de serem reutilizáveis e da forma mais apropriada para cada utilizador. Podem ser desenvolvidas aplicações orientadas a modelos, ou com o formato de *canvas*, que é uma tela onde são inseridos os componentes da aplicação. Esta plataforma integra com os vários serviços da Microsoft, como o SharePoint ou o Azure (wibjorn).

2.1.2 OutSystems

O OutSystems é uma plataforma de desenvolvimento *Low-code* que permite desenvolver aplicações de desktop ou moveis, e podem estar implantadas na *cloud* ou em infraestruturas locais. Esta plataforma tem 2 componentes principais, a componente de conexão à base de dados e a componente que permite especificar o comportamento das aplicações a desenvolver. Algumas das aplicações desenvolvidas sobre esta plataforma são CRMs, ERPs, extensões para ERPs já desenvolvidos, e Buisness Inteligence.
O OutSystems disponibiliza ainda mecanismos para publicar as aplicações, conectar com diferentes serviços, desenvolver apps responsivas e mecanismos de segurança (*OutSystems*).

2.1.3 Mendix

O Mendix é uma plataforma de desenvolvimento *Low-code* que não requere qualquer tipo de escrita de código e todas as funcionalidades são bastante intuitivas. Tem uma ferramenta que facilita a reutilização de componentes, e permite aplicar técnicas de *machine learning* e serviços cognitivos. É compatível com Docker e Kubernetes e tem vários *templates* para iniciar os projetos (*Mendix*).

2.1.4 FlutterFlow

O FlutterFlow é uma ferramenta *Low-code* desenvolvida sobre Flutter. Esta ferramenta foi lançada em Maio de 2021, ou seja, no decorrer do desenvolvimento deste projeto. Esta ferramenta permite definir a componente gráfica, o comportamento da aplicação e chamadas *apis*, apesar de o maior foco ser na integração com o Firebase, uma plataforma de bases de dados da Google. Esta plataforma de *low-code* permite fazer o download direto do código em flutter, portanto, combate exatamente o mesmo problema deste projeto. Outro fator positivo é a integração com o github o que facilita o versionamento das aplicações (*FlutterFlow*).

2.2 Desenvolvimento Cross Platform Mobile

Atualmente para o desenvolvimento móvel é necessário desenvolver a mesma aplicação para sistemas operativos diferentes, o que se pode tornar contraprodutivo. Em resposta a este problema apareceram *frameworks* que conseguem com que as aplicações corram em sistemas operativos diferentes tendo sido implementadas numa só plataforma, são estas as *frameworks Cross-Platform*.

Um estudo (Işitan & Koklu, 2020) comparou algumas desta *frameworks* num sentido de primeiro perceber quais as mais populares. Alguns dos dados dos estudos estavam desatualizados, mas tendo por base as mesmas fontes dos dados foram usadas aqui essas mesmas fontes, mas com dados mais recentes.

As *frameworks* comparadas foram as da seguinte tabela.

Tabela 1 – Descrição das ferramentas

	React Native	*Flutter*	*Xamarin*	*Ionic*	*NativeScript*
Open Source	Sim	Sim	Sim, pago	Sim, pago	Sim, pago
Empresa	Facebook	Google	Microsoft	Drifty.co	Telerik
Tecnologias	Javascript	Dart	C#	Typescript	Angular
Plataformas	iOS, Android	iOS, Android	iOS, Android, Windows	iOS, Android	iOS, Android
Data de lançamento	2015	2017	2012	2013	2015

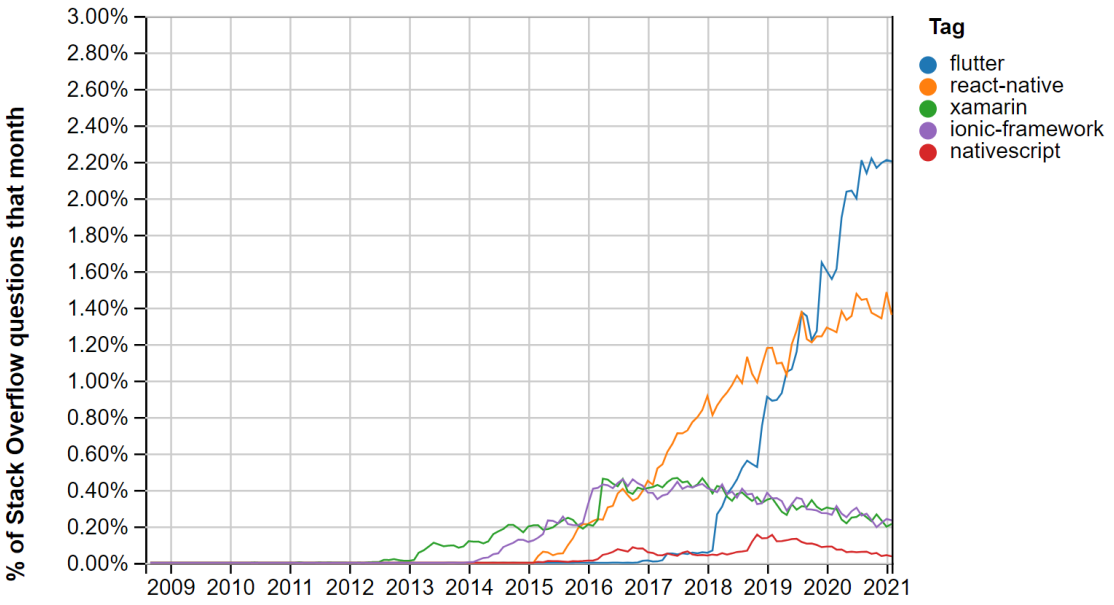


Figura 3 – Tendências de *frameworks* no Stackoverflow (*Stack Overflow Trends*)

O Stackoverflow é uma das comunidades mais populares no desenvolvimento de software, e este gráfico representa a percentagem de pergunta relativas às tecnologias ao longo do tempo. O Flutter aqui a aparecer com mais destaque nos últimos anos.

A seguir temos um gráfico, de um questionário que o Stackoverflow faz todos os anos, relativo as *frameworks* preferidas dos desenvolvedores, e o Flutter aparece à frente da concorrência ao nível do desenvolvimento móvel.

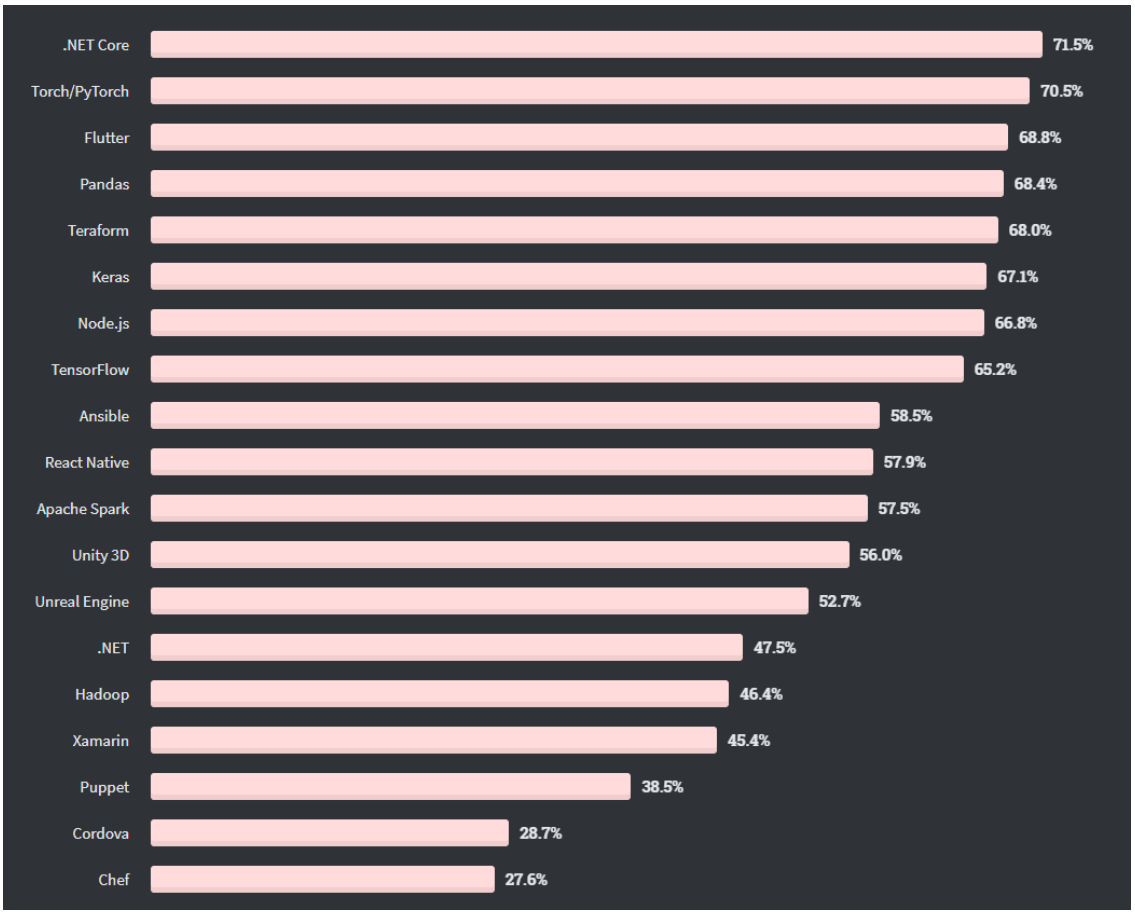
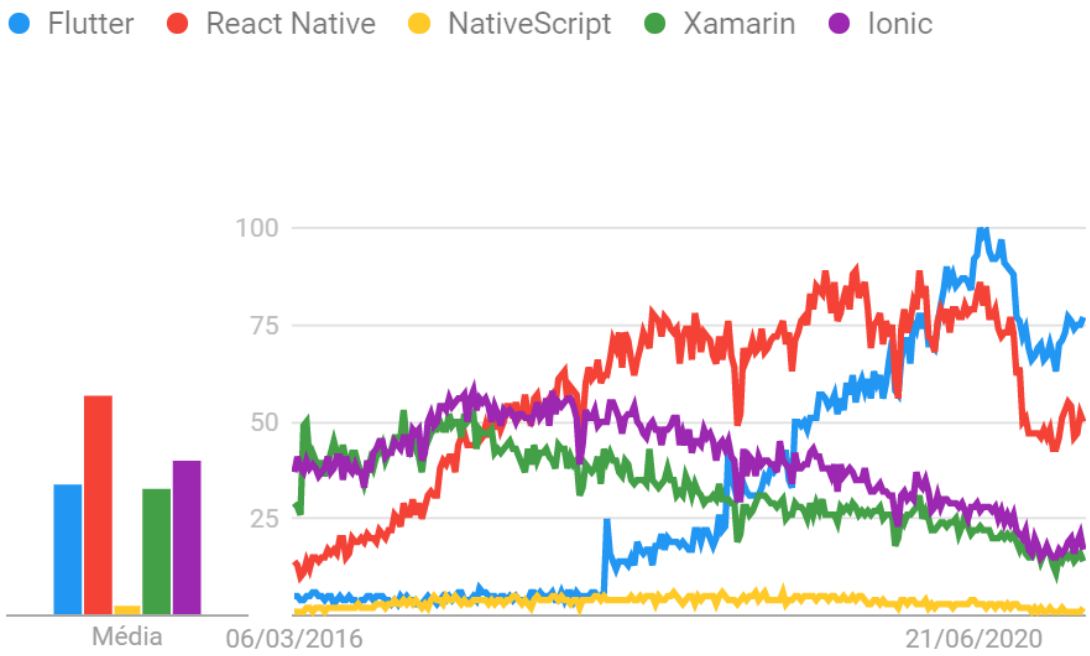


Figura 4 – Lista das *frameworks* mais populares (*Stack Overflow Developer Survey 2020*)



No gráfico acima está presente a comparação entre as diferentes *frameworks* ao nível das tendências da Google, o Flutter com melhores resultados nos últimos anos.

Nas 2 seguintes tabelas, temos na primeira o número de projetos no Github, que é a plataforma mais popular no desenvolvimento de software (Işitan & Koklu, 2020). E em seguida o número de stars no Github, aparecendo novamente o Flutter em primeiro.

Tabela 2 – Número de projetos e estrelas no Github

PLATAFORMA	NÚMERO DE PROJETOS NO GITHUB	NÚMERO DE ESTRELAS NO GITHUB
REACT NATIVE	21,832	90,700
FLUTTER	15,140	105,000
NATIVESCRIPT	687	19,100
XAMARIN	3,129	8,300

O mesmo estudo (Işitan & Koklu, 2020) após comparar a popularidade de cada *framework* realizou testes de performance onde verificou o tempo de renderização, rácio de CPU usado, quanta memória usada e quanta energia consumida. E no fim concluiu que a *framework* com mais popularidade e melhores resultados seria o Flutter.

2.3 Modelação de Aplicações

Os modelos sempre tiverem uma grande importância em diversos contextos científicos, como em áreas de física, química, matemática e até mesmo filosofia (Brambilla et al., 2017).

E os modelos podem ter 2 papeis fundamentais, um de funcionalidade de redução, quando refletem apenas uma parte do original, permitindo assim dirigir o foco para os aspetos mais importantes. Outro papel é o de mapeamento, baseado num elemento individual que é tido como protótipo e tornado abstrato e generalizado.

Os modelos podem ter propósitos diferentes, descritivos, descrevendo a realidade de um sistema ou contexto, prescritivos, determinando a área e os detalhes a estudar de um dado problema, ou para definirem como um sistema deve ser implementado.

De certo modo, é possível concluir que "tudo é um modelo" visto que nada pode ser processado pela mente humana sem ser modelado, ainda que seja apenas uma modelação mental (Olivé, 2004). Assim é possível perceber a importância que os modelos têm nas diferentes áreas da engenharia, incluindo a informática.

2.3.1 Modelação no desenvolvimento de software

Alguns dos motivos pela modelação ser cada vez mais importante no desenvolvimento de software são, em primeiro, os artefactos de software se tornarem cada vez mais complexos e aparecer a necessidade de serem discutidos em diferentes níveis de abstração. Outro motivo é a questão de o software estar cada vez mais presente nas vidas das pessoas e, sendo assim, a expectativa de que terão de ser acrescentadas novas peças aos sistemas já existentes e até mesmo a necessidade de evoluir esses mesmos sistemas. Outro grande motivo é o facto de o este tipo de desenvolvimento requerer a interação com não desenvolvedores o que torna necessário alguma mediação na descrição de termos técnicos (Acerbis et al., 2007).

Segundo a classificação de Martin Fowler (*Martin Fowler*, 2003) os modelos podem ter diferentes utilidades, podem ser usados como sketches, com o propósito de comunicação em que apenas são especificadas algumas vistas do sistema, podem ser usados como *blueprints*, onde é fornecida a especificação completa e detalhada do sistema e por último como programas onde são usados modelos em vez de código para desenvolver sistemas.

2.3.2 Linguagens de modelação

As linguagens de modelação são um dos principais componentes da modelação no desenvolvimento de software. Este tipo de linguagens são uma ferramenta que permite desenhar e especificar modelos de sistemas, tanto a um nível gráfico como textual (Brambilla et al., 2017).

Podemos dividir estas linguagens em duas grandes classes, *Domain-Specific Languages* que são desenhadas para um específico domínio ou contexto e *General-Purpose Modeling*

Languages que são ferramentas que podem ser aplicadas a diferentes domínios, como o caso do UML.

Os modelos descritos por estas linguagens podem descrever 2 dimensões de um sistema, estática ou dinâmica. Os modelos estáticos focam-se, como o nome indica, nos aspetos estáticos de um sistema, como a estrutura dos dados e arquitetura. Os modelos dinâmicos descrevem o comportamento do sistema expondo a execução e sequência de ações e a comunicação entre os diferentes componentes.

2.3.3 Meta-modelação

Tal como já referido os modelos são um componente importante da modelação, então uma necessidade subsequente é definir os próprios modelos através de um modelo mais abstrato, ou seja, da mesma forma que definimos um modelo como uma abstração de um fenómeno da realidade, podemos definir um metamodelo como uma abstração de um modelo. Os metamodelos definem uma linguagem de modelação, disponibilizando uma forma de descrever cada classe dos modelos que podem ser representados pela linguagem. Assim, recursivamente poderemos também definir meta-metamodelos, que seriam a abstração dos metamodelos, e apesar de esta cadeia poder ser infinita, considera-se que este seria o último nível visto que os meta-metamodelos podem se definir a si próprios (Kühne, 2006).

As principais utilidades destes metamodelos são definir novas linguagens de modelação ou programação, definir linguagens para gestão e armazenamento de dados e definir novas propriedades para serem associadas com informação já existente.

2.3.4 Engenharia reversa

Atualmente existe o problema de manter e gerir ou até mesmo de substituir sistemas já existentes, que por vezes foram construídos com tecnologias agora obsoletas ou que não têm documentação suficiente disponível. E uma das formas de fazer evoluir estes sistemas é de perceber a sua arquitetura, funcionalidades, estrutura de dados regras e processos de negócio. A este processo de obter informação útil deste tipo de sistemas é chamado de engenharia reversa (Brambilla et al., 2017).

Podemos definir 3 fases para conceber este processo, sendo a primeira a descoberta do modelo, que tem como objetivo converter a informação heterogénea retirada do sistema em modelos homogéneos. Uma forma de criar estes modelos é representar o sistema completo com um baixo nível de abstração para que não haja perda de informação.

De seguida vem a fase de compreensão, que requer processar os modelos descobertos na fase anterior e obter uma vista com maior abstração para facilitar a análise destes sistemas.

Nesta fase são então feitas algumas transformações aos modelos que eram mais crus tornando-os mais manuseáveis omitindo detalhes não relevantes.

Na última fase, de geração, são utilizados os modelos da fase anterior para gerar os resultados esperados como exemplo código ou uma versão do sistema reformulado.

2.3.5 Transformações entre modelos

As transformações entre modelos permitem a definição do mapeamento entre diferentes modelos e são definidas ao nível do metamodelo e aplicadas ao nível do modelo. Estas transformações são efetuadas a partir de um modelo origem para um modelo destino, e as regras que permitem definir este mapeamento são especificadas através de linguagens apropriadas. Estas linguagens disponibilizam *templates* de transformação onde são aplicadas regras de *matching* aos elementos do modelo (Sendall & Kozaczynski, 2003).

2.4 Ferramentas de geração de código

Nesta secção são apresentadas algumas ferramentas que permitem gerar código.

A primeira ferramenta é o Hygen que é uma ferramenta de geração de código que foi desenvolvida para tornar uma equipa de desenvolvedores mais eficaz (*Generators | Hygen*, sem data). Uma das suas características é que os seus geradores são geralmente incluídos dentro de projetos em desenvolvimento, podendo assim os geradores escalarem com o projeto. No entanto é também possível de criar geradores globais.

Os comandos utilizados para a geração de ficheiros acompanham a estrutura de pastas dos *templates*, e a linguagem utilizada nesses *templates* é *Embedded JavaScript* que é uma linguagem de *templates* muito idêntica a JavaScript.

O Hygen possibilita também adaptar e customizar a interação com o utilizador através da linha de comandos podendo passar e tratar qualquer tipo de parâmetros.

A segunda ferramenta é o Yeoman que tem o objetivo de tornar o desenvolvimento de software mais produtivo. Esta ferramenta é mais orientada à publicação de geradores na comunidade (*The web's scaffolding tool for modern webapps | Yeoman*, sem data).

3 Análise de Valor

A análise de valor é um processo que deve ser metódico, pressupondo assim que haja um planeamento, controlo e coordenação desta atividade. Deve atender às necessidades dos clientes e para tal deve-se entender a finalidade do produto em questão. Sendo compreendido o uso do produto, é necessário ajustar as especificações do mesmo com o intuito de agregar o maior valor para o cliente. Por fim de forma a gerar o maior benefício para a empresa, é essencial aprimorar o processo de produção de forma a reduzir os custos sem sacrificar a qualidade do produto (Rich, Nick and Holweg, Matthias, 2000).

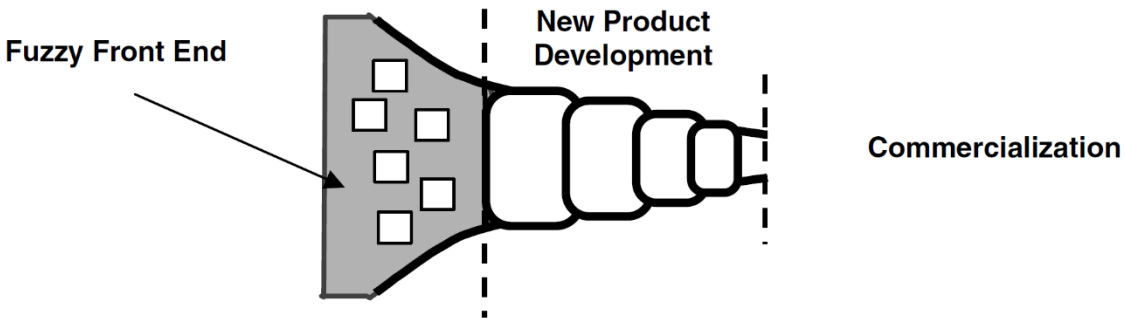


Figura 5 - Fuzzy Front End (Belliveau et al., 2004)

O processo de inovação pode ser dividido em três fases, sendo a primeira a Inovação de Front End, ou Fuzzy Front End, que é definida pelo conjunto de atividades que antecedem a fase formal e bem estruturada chamada Desenvolvimento do Produto Novo (the new product development (NPPD) process), acabando com a última fase que são as etapas de comercialização. (Belliveau et al., 2004)

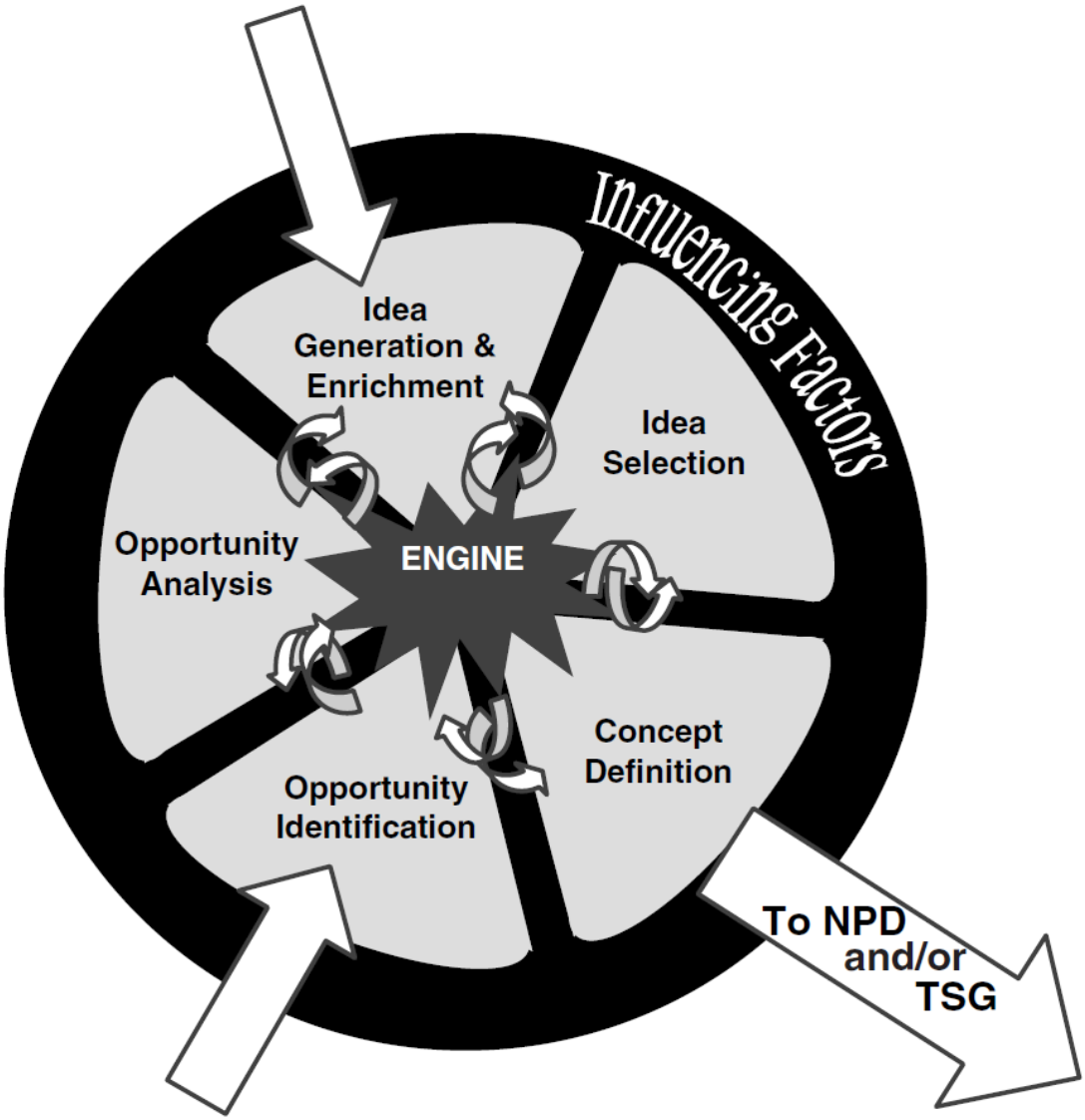


Figura 6 - Modelo de NCP (Belliveau et al., 2004)

NCD:

- **Fatores que influenciam (*Influencing Factors*):** O FFE está contido num ambiente que consiste nas estratégias de negócios da empresa, fatores competitivos, as suas capacidades organizacionais e a maturidade das tecnologias a serem utilizadas. O desenvolvimento sustentado e bem-sucedido do produto só pode ocorrer quando as atividades do FFE podem ser realizadas com os recursos organizacionais da empresa.

- **O motor (*engine*)** é o mecanismo que aciona os cinco elementos de *front-end* e é sustentado pela liderança e pela cultura da organização.

- **Área interna:** define os cinco elementos de atividade controláveis (identificação da oportunidade, análise da oportunidade, geração e enriquecimento de ideias, seleção de ideias e definição de conceito) do FFE.

3.1 Identificação de oportunidade

Como já referido anteriormente as plataformas de *Low-code* apesar de serem uma mais-valia para o desenvolvimento de software, por vezes podem não ser suficientes ou apresentar limitações. É então necessário migrá-las para uma aplicação nativa.

Este tipo de migração pode ser demorado e até mesmo conter algumas falhas, e como é um processo em que é necessário ser repetido a cada vez que se necessita de passar de uma LCS para uma tradicional apareceu a conveniência de tornar este processo automatizado.

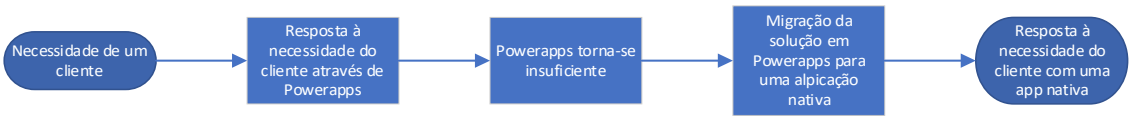


Figura 7 – Diagrama de processos da evolução de uma app

Como é percetível no fluxograma a resposta a uma necessidade de um cliente é feita numa plataforma *Low-code* pelo facto de ser muito mais rápido o seu desenvolvimento e assim o feedback do cliente é também mais rápido. Muitas vezes esta solução torna-se insuficiente por diversos motivos, dos quais, a falta de performance, escalabilidade ou customização da solução. É feita então a migração pelos desenvolvedores para uma app nativa, sendo que a proposta deste trabalho é tornar esse processo automatizado.

3.2 Análise de oportunidade

Um estudo feito pelo Forrester, em que são comparadas as principais plataformas de Low-code, visível nas figuras 8 e 9, aponta a plataforma da Microsoft, PowerApps, como uma das líderes de mercado e é ainda a que obteve melhor pontuação neste estudo. Isto acrescenta valor ao projeto pois não só estamos a fazer a migração de uma qualquer plataforma de *Low-code* mas sim de uma plataforma líder de mercado. (Rymer & Koplowitz, 2019)

A tecnologia decidida para realizar as aplicações nativas foi o Flutter, tecnologia desenvolvida pela google, em que num estudo realizado (Işitan & Koklu, 2020), foi destacada como a tecnologia mais popular, e em comparação com o seu maior concorrente, React Native, apresentou melhores níveis de performance.

Figura 8 – Gráfico de comparação de LCDP (Rymer & Koplowitz, 2019)

	Forrester's weighting	Clear Software	GeneXus	Kony	MatsSoft	Mendix	Microsoft	OutSystems
Current offering	50%	0.84	3.18	4.10	2.54	4.49	3.95	4.58
Tooling for application development	33%	1.54	3.34	3.90	2.82	4.28	4.26	4.14
Tooling for platform and app admin	33%	0.60	3.00	4.60	2.60	5.00	3.80	5.00
App deployment, ops tools, and features	34%	0.40	3.20	3.80	2.20	4.20	3.80	4.60
Strategy	50%	1.75	3.00	3.50	2.25	3.50	4.50	3.50
Vision and strategy	25%	1.00	3.00	5.00	5.00	5.00	5.00	5.00
Training, community, and marketplace	25%	3.00	3.00	5.00	3.00	3.00	5.00	3.00
Partners	25%	0.00	3.00	1.00	0.00	3.00	5.00	3.00
Commercial model	25%	3.00	3.00	3.00	1.00	3.00	3.00	3.00
Market presence	0%	2.32	2.69	2.34	1.67	3.33	4.67	2.67
Revenue from low-code platform sales	33%	1.00	2.00	3.00	1.00	3.00	4.00	3.00
Revenue growth rate	33%	5.00	1.00	1.00	2.00	4.00	5.00	2.00
Number of enterprise customers	34%	1.00	5.00	3.00	2.00	3.00	5.00	3.00

Figura 9 – Tabela de comparação de LCDP (Rymer & Koplowitz, 2019)

3.3 Proposta de valor

Para os desenvolvedores da DevScope que precisam de agilizar o processo de migração de apps, esta solução é uma ferramenta que permite extrair a informação de uma PowerApp e a partir disso construir uma app em flutter, e ao contrário de fazer este processo manualmente através desta solução será muito mais rápido e não será passível do erro humano.

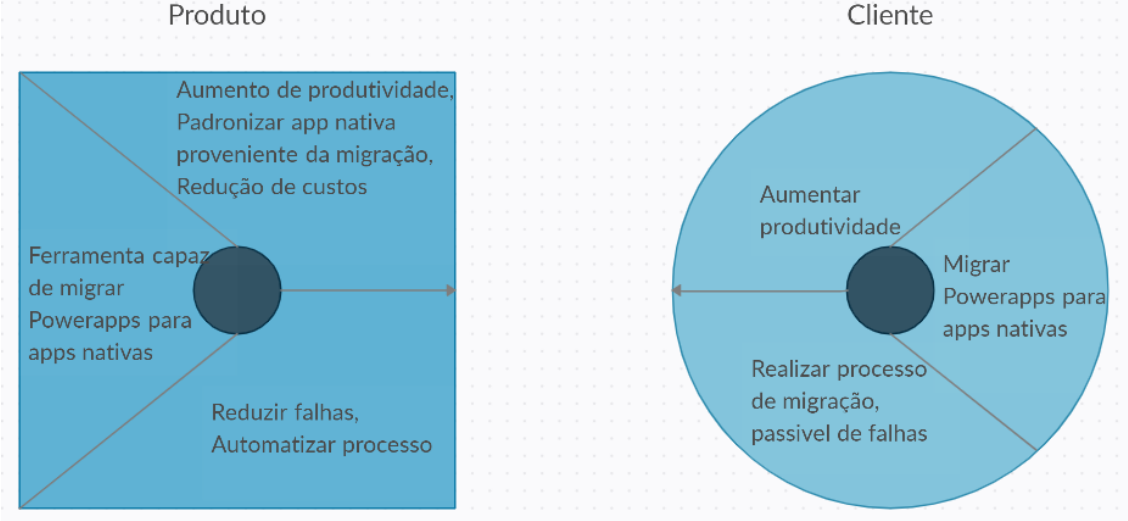


Figura 10 – Canvas da proposta de valor

A Figura 10 representa o *Canvas* da proposta de valor, em que está presente à esquerda o produto e à direita a visão do cliente. O produto está dividido em 3 partes, são estas:

- Criadores de benefícios – Aumento da produtividade, padronização das apps nativas desenvolvidas e a redução de custos deste processo
- Aliviador das dores – Automatizar o processo e reduzir as suas falhas.
- Produto – Ferramenta capaz de migrar PowerApps para aplicações móveis nativas.

A visão do cliente está também dividida em 3 partes, que são:

- Benefícios – Aumentar a produtividade das equipas de desenvolvimento de software
- Sacrifícios – Realizar processo de migração passível de falhas.
- Trabalho a Fazer – Migrar PowerApps para apps móveis nativas.

3.4 Diagrama FAST

Aqui apresentamos um diagrama FAST que é a representação gráfica da lógica de um projeto expondo o seu porquê e como irá chegar à solução. Sendo que lendo o diagrama da esquerda para a direita percebemos como iremos atingir a solução e da direita para a esquerda explicamos o porquê do projeto.

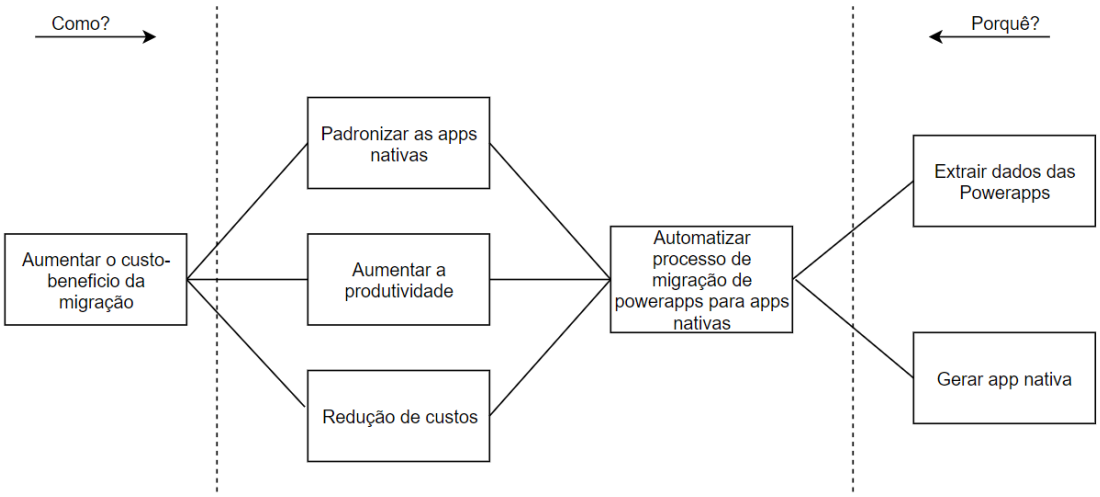


Figura 11 – Diagrama FAST

Tal como está descrito na Figura 11, começando pelo lado direito, está exposto o maior porquê do projeto e é este aumentar o custo-benefício da migração através da padronização das apps migradas, aumentando a produtividade das equipas de desenvolvimento e reduzindo os custos. Para isso é necessário automatizar o processo que é fazer esta migração, e isso é possível extraindo os dados das PowerApps e gerando as apps nativas.

3.5 Método AHP

O método AHP é um método que auxilia na decisão de uma alternativa para um dado problema, avaliando-se as alternativas consoante alguns critérios.

Apontou-se como objetivo escolher a tecnologia para a geração do código Flutter. Sendo qua as alternativas selecionadas foram as tecnologias Hygen, Yeoman e Plop.

Critérios: Orientado ao projeto, Popularidade, Facilidade de utilização.

Sendo que a orientação ao projeto indica se a ferramenta é orientada à publicação dos geradores como é o caso do Yeoman ou orientada a projetos específicos como o Hygen. A popularidade e a facilidade apontam o que o próprio nome indica.

3.5.1 Comparação entre critérios

Nesta fase é feita a comparação dos critérios atribuindo-se um peso a cada um em relação aos outros.

Tabela 3 – Comparação entre elementos

	Facilidade de utilização	*Popularidade*	*Orientado ao projeto*
Facilidade de utilização	1	1/2	1/5
Popularidade	2	1	1/3
Orientado ao projeto	5	3	1

Tabela 4 – Comparação de elementos normalizada

Tabela normalizada	*Facilidade de utilização*	*Popularidade*	*Orientado ao projeto*	*Prioridade*
Facilidade de utilização	1/8	1/9	3/23	0.12
Popularidade	2/8	2/9	5/23	0.23
Orientado ao projeto	5/8	2/3	15/23	0.65

3.5.2 Critério Facilidade de utilização

Tabela 5 – Comparação de Facilidade de utilização

	Hygen	*Yeoman*	*Plop*
Hygen	1	3	2
Yeoman	1/3	1	2
Plop	1/2	1/2	1

Tabela 6 - Comparação de Facilidade de utilização normalizada

	HyGen	*Yeoman*	*Plop*	
HyGen	6/11	2/3	2/5	0.54
Yeoman	2/11	2/9	2/5	0.27
Plop	3/11	1/9	1/5	0.19

3.5.3 Critério Popularidade

Tabela 7 – Comparação de popularidade

	HyGen	*Yeoman*	*Plop*
HyGen	1	1/5	1/2
Yeoman	5	1	3
Plop	2	1/3	1

Tabela 8 – Comparação de popularidade normalizada

	HyGen	*Yeoman*	*Plop*	
HyGen	1/8	3/23	1/9	0.12
Yeoman	5/8	15/23	2/3	0.65
Plop	2/8	5/23	2/9	0.23

3.5.4 Critério orientação ao projeto

Tabela 9 – Comparação de orientação ao projeto

	HyGen	*Yeoman*	*Plop*
HyGen	1	7	6
Yeoman	1/7	1	1/2
Plop	1/6	2	1

Tabela 10 - Comparação de orientação ao projeto normalizada

	HyGen	*Yeoman*	*Plop*	
HyGen	42/55	7/10	4/5	0.76
Yeoman	6/55	1/10	1/15	0.09
Plop	7/55	2/10	2/15	0.15

3.5.5 Resultado

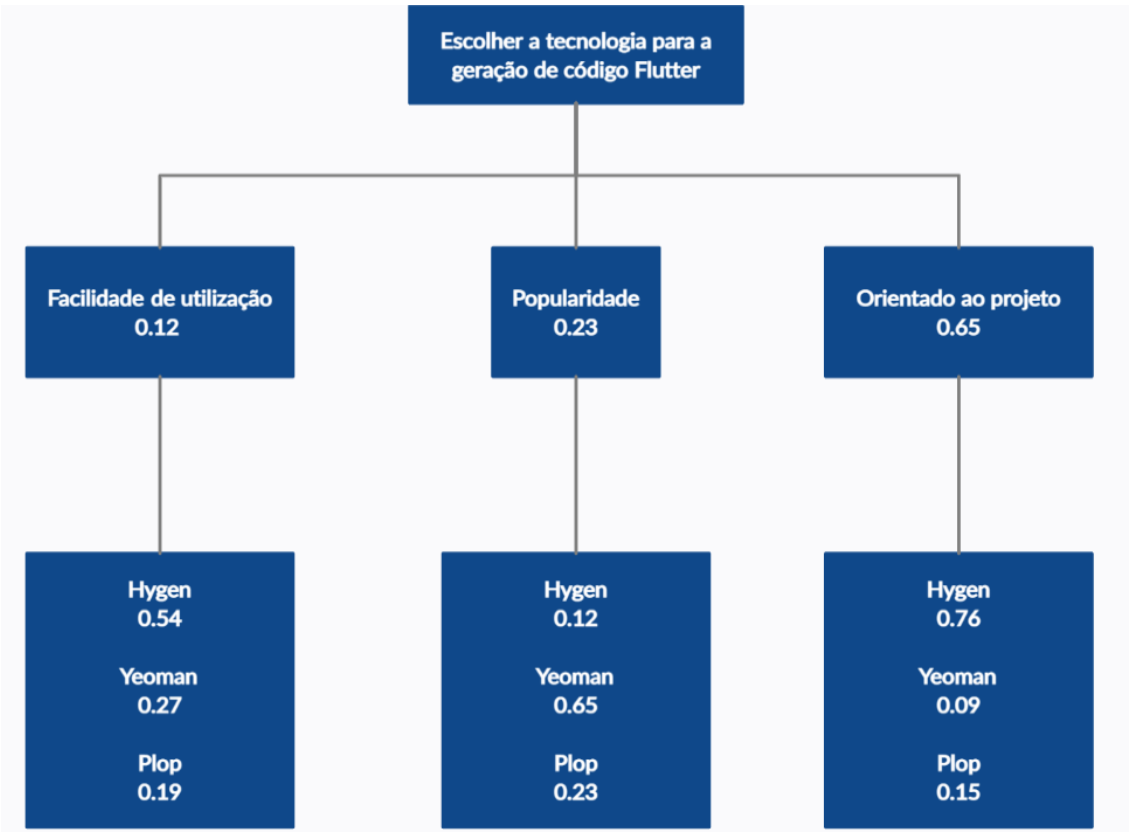


Figura 12 – Árvore hierárquica resultante dos cálculos

Tabela 11 – Matriz prioridade e multiplicação da matriz pelos pesos dos critérios

	Facilidade de utilização	*Popularidade*	*Orientado ao projeto*	*Multiplicação pelos pesos dos critérios*
HyGen	0.54	0.12	0.76	0.59
Yeoman	0.27	0.65	0.09	0.24
Plop	0.19	0.23	0.15	0.17

Consoante os resultados percebemos que a melhor opção seria o Hygen.

4 Engenharia de requisitos

Neste capítulo são documentados os requisitos da solução a desenvolver.

4.1 Visão

Aqui são apresentadas as partes interessadas do projeto a desenvolver e algumas questões feitas à equipa responsável pelo projeto e as suas respostas.

Partes Interessadas:

- Desenvolvedores da DevScope, na medida em que aceleram o processo de migração das aplicações e tornam este processo menos propicio a falhas.
- Os clientes da DevScope, na medida em que a produção dos seus produtos é mais rápida e menos passível de erros.

As questões apontadas tiveram como referência as aulas do modulo de Engenharia e Análise de Requisitos.

Primeiras perguntas:

- Quem está por de trás do pedido deste trabalho?
    - Equipa de portais da DevScope.
- Quem irá usar a solução?
    - Desenvolvedores da DevScope.
- Qual será o benefício económico de uma solução bem-sucedida?
    - Aumento da produtividade no processo de migração das apps.
- Existe outro meio para atingir a solução necessária?
    - Fazer a migração pelos desenvolvedores de uma forma não automatizada.

Perguntas seguintes:

- Como caracteriza um bom output que será gerado pela solução bem-sucedida?
    - Uma app em flutter funcional e idêntica à Powerapp em questão.
- Quais problemas a solução endereça?
    - Processo de migração demorado, repetitivo e passível de falhas.
- Qual o ambiente de negócio em que a solução será usada?
    - Ambiente de desenvolvimento de software da DevScope.
- Existem necessidades de performance ou restrições que afetem a abordagem à solução?

- A ferramenta deve migrar a aplicação mais rápido do que o processo não automatizado.

Perguntas finais:

- As respostas dadas são oficiais?
  - Sim.
- Foram demasiadas perguntas?
  - Não.

Estas respostas permitiram perceber melhor o objetivo desta solução e definem de uma forma oficial os requisitos da solução.

4.2 Elicitação de Requisitos

Nesta secção estão documentadas as medidas tomadas para compreender melhor os requisitos e algumas restrições, e são estas:

- Reuniões – Foram realizadas reuniões com desenvolvedores de PowerApps, Flutter e inclusive desenvolvedores que migraram de uma forma não automatizada de uma para outra.
- Workshops – Foram realizados tutoriais de PowerApps e Flutter, e assistidas apresentações de ambas as tecnologias.
- Experimentações – Foram realizadas experiências de desenvolvimento de PowerApps e Flutter.
- Análise de documentação – Foi analisada a documentação de PowerApps e Flutter

4.3 Elaboração de artefactos

Nesta secção são apresentados os artefactos que auxiliam a documentação da análise e engenharia de requisitos.

4.3.1 Detalhes da Plataforma Powerapps

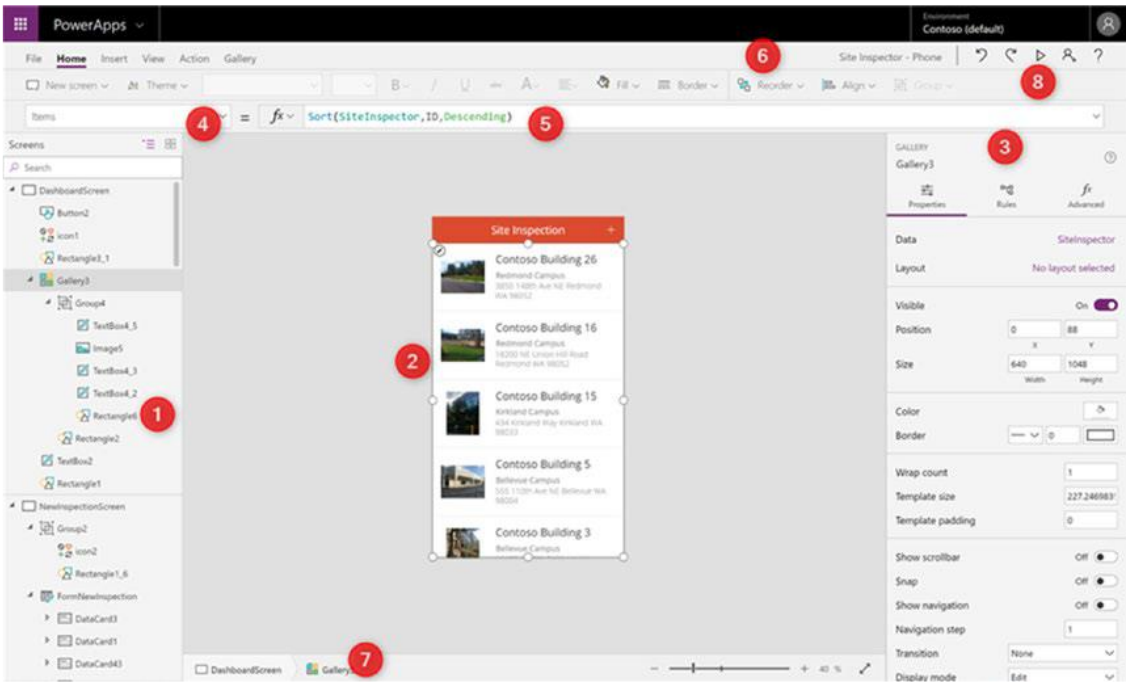
Existem 3 tipos de apps que se podem desenvolver em PowerApps: Canvas apps, Model-Driven apps e Portais (Pearson et al., 2020).

As Canvas apps permitem criar aplicações com um nível grande de customização, mas requerem mais trabalho de implementação que as outras. Podem conectar-se a centenas de fontes de dados, e a outros conectores que disponibilizam ações, como o envio de mensagens.

As model-driven apps são mais focadas em apresentar dados e em seguirem um fluxo definido. Este tipo apenas usa como fonte de dados o Dataverse, uma plataforma de dados.

Os portais são sites de intranet, ou extranet, que permitem o acesso anónimo, ou através de uma autenticação, como o Facebook ou Linkedin, entre outros. Também requerem o Dataverse como fonte de dados.

Ambiente de desenvolvimento:



1 – Barra de navegação à esquerda – Onde se encontram os ecrãs e os componentes da aplicação.

2 – Painel central – A preview da aplicação.

3 – Coluna à direita – Configuração dos vários componentes.

4 – Lista de propriedades – Onde se seleciona a propriedade a configurar.

5 – Barra da fórmula – Onde adicionar código.

6 – Fita – Tal como outras aplicações do office, onde se pode navegar entre as áreas principais da plataforma.

7 – Barra de fundo – Mostra a parte da aplicação selecionada.

8 – Botão de preview – Onde correr a app em modo preview.

Connectors

Os conectores podem ser pré-existentes na plataforma, ou podem ser criados pelo desenvolvedor. Podem ser de vários tipos, como conexões a bases de dados, ou, outro exemplo, envio de emails.

Controls

Os *controls* são os elementos que constituem os ecrãs, que, por sua vez, constituem a aplicação. Eles podem ser inseridos nos ecrãs e depois configurados consoante várias propriedades, como a posição, cor, tamanho e ações a desempenhar aquando de algum acontecimento, como exemplo, ao clicar, ou quando este se tornar visível. Alguns dos *controls* mais populares são retângulos, caixas de texto, caixas de inserção de texto, botões, etc.

Expressões

As expressões em Powerapps são utilizadas para definir a lógica da aplicação. Utilizam uma linguagem muito parecida à das fórmulas de Excel para facilitar o desenvolvimento a não programadores. Uma expressão pode conter várias funções disponibilizadas pelo Power Apps.

O powerapps disponibiliza os metadados que constroem uma aplicação num ficheiro com a extenção .msapp que no fundo é uma pasta comprimida com a estrutura presente na figura X

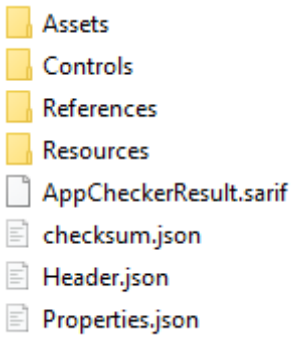


Figura 13 – Constituição da pasta .msapp

Dentro desta pasta encontram se os seguintes componentes:

- Assets – Uma pasta com os assets da aplicação

- Controls – Uma pasta com um ficheiro de configurações da app e um ficheiro de configurações relativo a cada ecrã da aplicação.
- References – Uma pasta com 4 ficheiros, um relativo as fontes de dados, recursos, templates e temas.
- Resources
- AppCheckerResult
- Checksum
- Header
- Properties

4.3.2 Meta-modelo das Powerapps

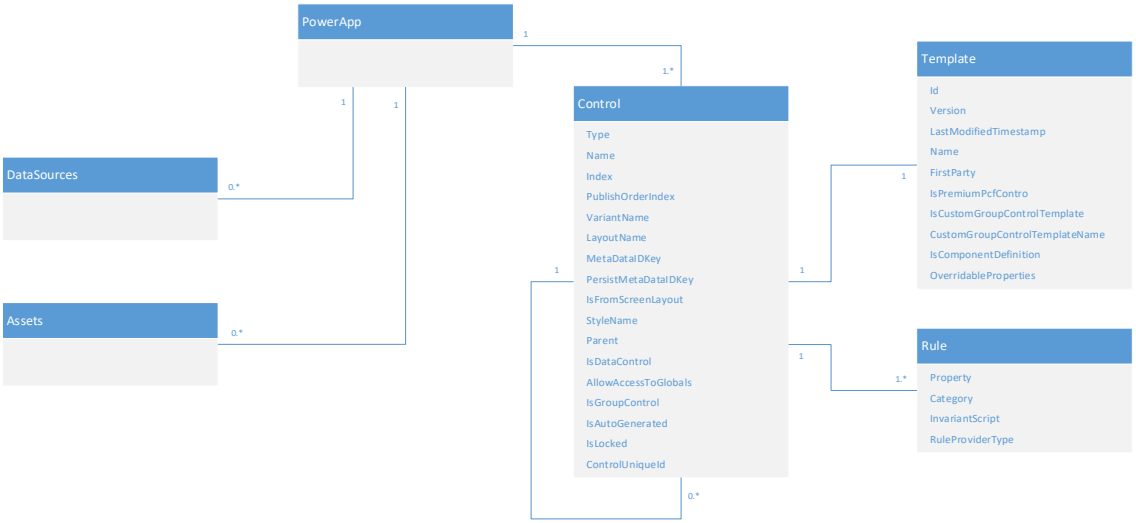


Figura 14 – Meta-modelo das Powerapps

Este meta-modelo representa os meta-dados de uma PowerApp.

PowerApp - aplicação desenvolvida na plataforma de Powerapps e que agrega todas as suas definições.

Control – o *control* pode ser ele próprio um conjunto de *controls*. Como exemplo, um *control* pode ser um ecrã e conter outros *controls* que são os que dão corpo ao próprio ecrã.

Template – contém a informação do tipo de *control* que se trata.

Rule – descreve as propriedades de um *control*, como a cor, tamanho, posição, etc. É também aqui que são definidas propriedades do comportamento da aplicação, como exemplo, numa propriedade OnSelect pode ser definida a navegação para outro ecrã. É aqui que são especificadas as Expressões de Powerapps, anteriormente referidas.

DataSources – fontes de dados que alimentam a app.

Assets – representam recursos utilizados na app como imagens.

4.3.3 Casos de uso

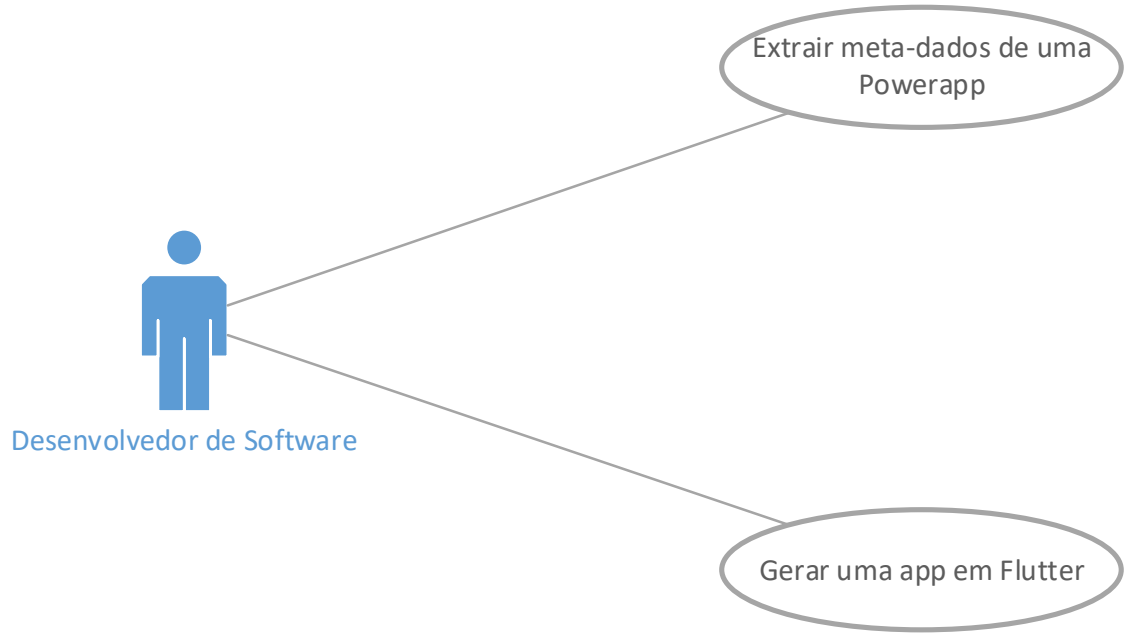


Figura 15 – Diagrama de casos de uso

Neste diagrama podemos ver que foram apontados 2 casos de uso, são estes, Extrair meta-dados de uma Powerapp e Gerar uma app em Flutter. Ambos partilham o mesmo ator, o Desenvolvedor de Software.

Apesar de os 2 casos de uso fazerem sentido em conjunto e poderem até terem sido apontados como um só, foi decidido separá-los porque têm também valor fora deste contexto.

O caso de uso Extrair meta-dados de uma Powerapp representa um fluxo iniciado pelo Desenvolvedor de software, de seguida este seleciona qual a Powerapp a desejada, e são extraídos os meta-dados da Powerapp construindo-se uma estrutura de dados mais percetível ao desenvolvedor.

O caso de uso Gerar uma app em Flutter é também iniciado pelo desenvolvedor, selecionando a estrutura extraída pelo caso de uso anterior, gerando posteriormente os ficheiros que constroem uma app em Flutter.

5 Desenho da Solução

Este capítulo apresenta diferentes alternativas para a solução do problema, as vantagens e desvantagens de cada uma.

5.1 Alternativa da solução acoplada

Nesta alternativa como podemos verificar nas figuras existe apenas uma componente responsável por extrair os dados das Powerapps e de também gerar a aplicação de flutter.

Uma vantagem desta alternativa é não haver a necessidade de criar a comunicação entre componentes diferentes, visto que a solução está centralizada no Gerador de Flutter app.

Em contrapartida esta solução só será reutilizada num contexto idêntico a este, tornando difícil a sua reutilização.

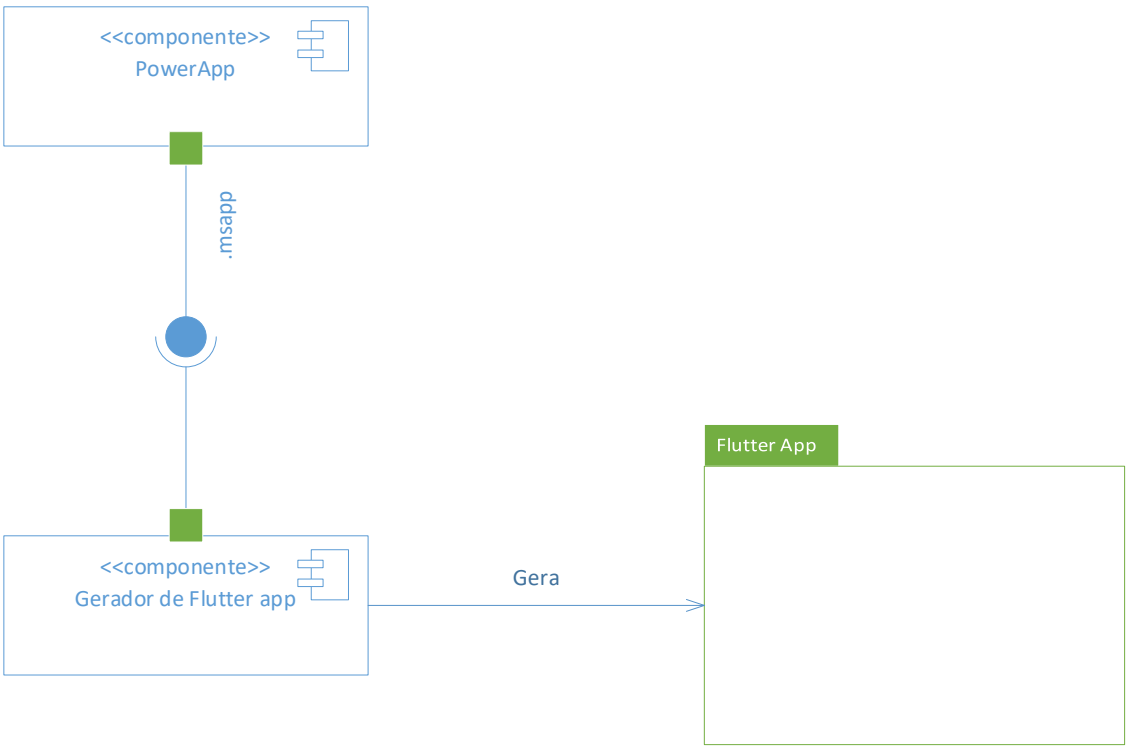


Figura 16 – Diagrama de componentes da solução acoplada

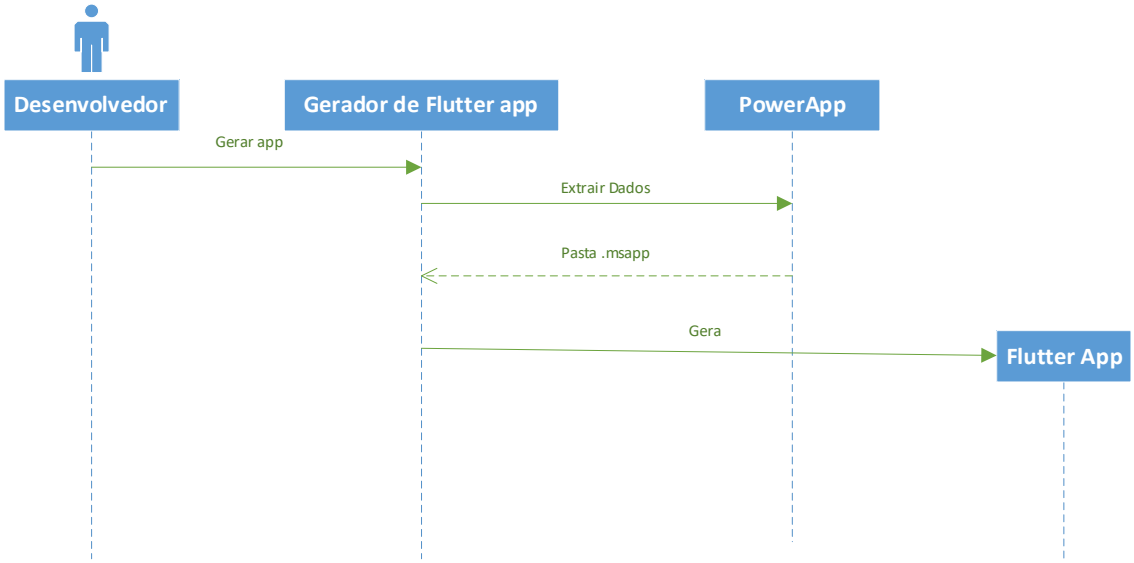


Figura 17 – Diagrama de sequência da solução acoplada

5.2 Alternativa da solução desacoplada

Nesta alternativa temos 2 componentes que em conjunto realizam a migração da aplicação, são eles o Transformador de Dados e o Gerador Flutter. O primeiro é responsável por extrair os dados da PowerApp e transformá-los numa estrutura mais facilmente percetível e mais facilmente trabalhável. O segundo é responsável por gerar a app em Flutter.

A grande vantagem desta alternativa é separar as responsabilidades podendo cada componente dedicar-se áquilo que realmente lhe importa, facilitando também a sua manutenção.

Esta solução permite ainda que cada componente seja reutilizado em cenários que não o deste projeto. Como exemplo a estrutura de dados construída a partir do primeiro componente seja reutilizada para elaborar a documentação de uma PowerApp. E um exemplo para o segundo componente seria gerar uma app em Flutter através de uma estrutura que o componente está à espera construída por um desenvolvedor.

Assim optou-se por usar esta alternativa para responder ao problema.

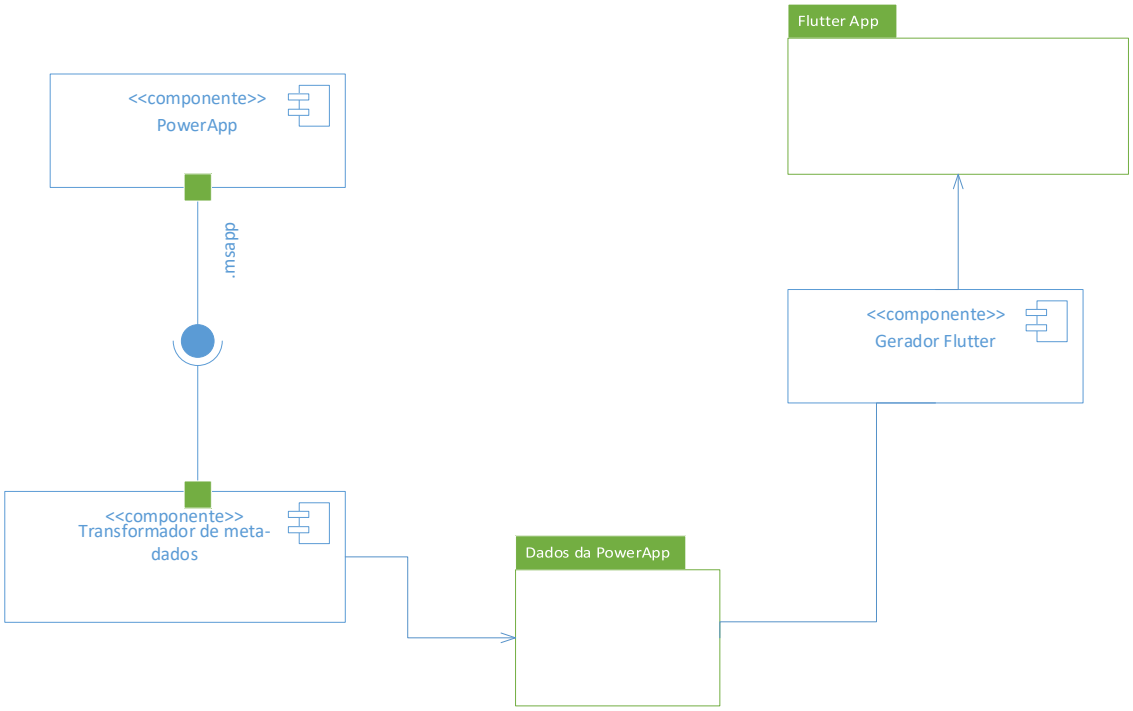


Figura 18 – Diagrama de componentes da solução desacoplada

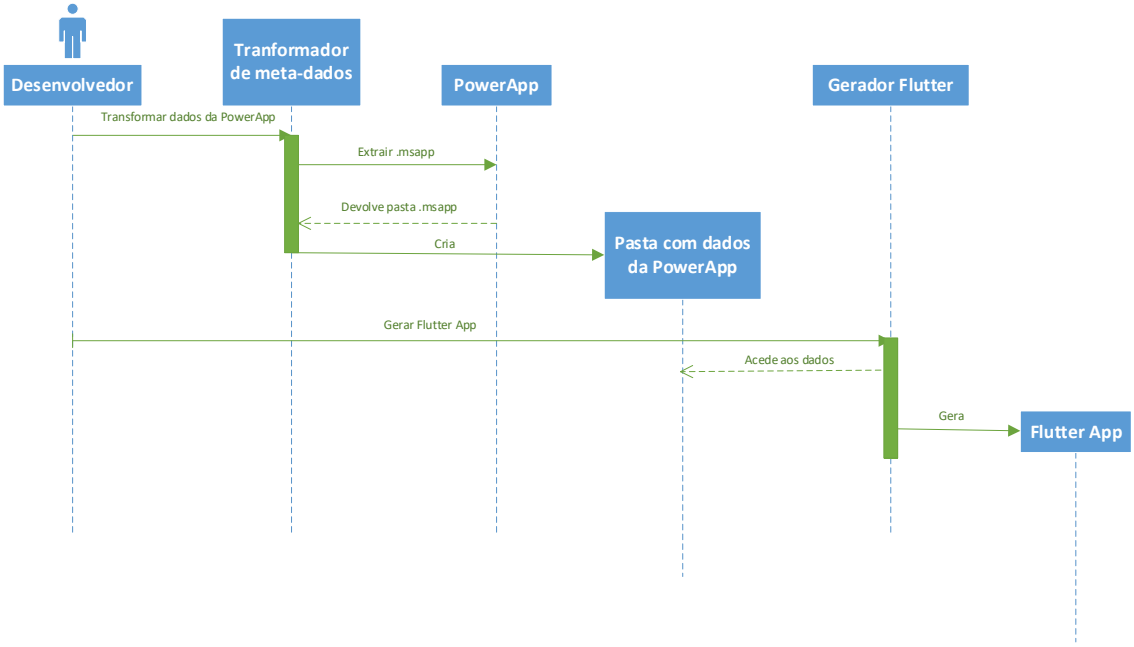


Figura 19 – Diagrama de sequência da solução desacoplada

5.2.1 Componente Transformador de meta dados

Este componente será o responsável por extrair dos meta-dados provenientes de uma powerapp apenas os meta-dados necessários para a geração da app em Flutter, e disponibilizá-los numa estrutura mais facilmente trabalhável.

Como já referido no capitulo anterior os meta-dados provém numa pasta com a extensão .msapp, sendo que as propriedades de cada componente se encontram dentro da pasta Controls num ficheiro json referente a cada ecrã.

De uma forma gráfica esta é a estrutura da origem:

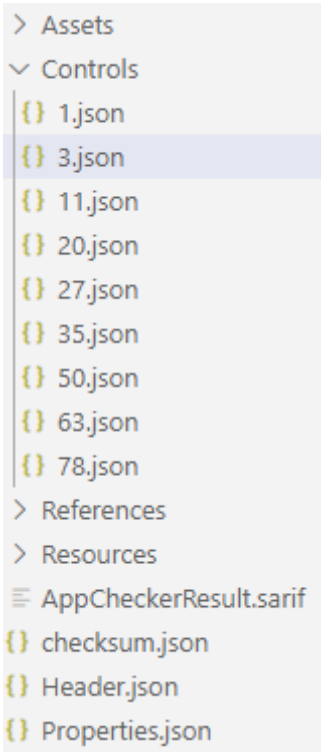


Figura 20 – Estrutura dos meta-dados de uma Powerapp

Sendo a Figura 20 a pasta msaap, e os ficheiros json identificados com um numero os ecrãs da Powerapp, e o excerto seguinte uma parte do ficheiro 3.json onde são visíveis as propriedades referentes a um control.


```
"Type": "ControlInfo",
    "Name": "TextBox1_9",
    "HasDynamicProperties": false,
    "Template": {
        "Id": "http://microsoft.com/appmagic/label",
        "Version": "2.3.2",
        "LastModifiedTimestamp": "0",
        "Name": "label",
        "FirstParty": true,
        "IsPremiumPcfControl": false,
        "IsCustomGroupControlTemplate": false,
        "CustomGroupControlTemplateName": "",
        "IsComponentDefinition": false,
        "OverridableProperties": {}
    },
    "Index": 0.0,
    "PublishOrderIndex": 0,
    "VariantName": "",
    "LayoutName": "",
```

```
"MetaDataIDKey": "",
"PersistMetaDataIDKey": false,
"IsFromScreenLayout": false,
"StyleName": "defaultLabelStyle",
"Parent": "Home_Screen",
"IsDataControl": false,
"AllowAccessToGlobals": true,
"IsGroupControl": false,
"IsAutoGenerated": false,
"Rules": [
    {
        "Property": "Text",
        "Category": "Data",
        "InvariantScript": "\"Fabrikam Connect\"",
        "RuleProviderType": "Unknown"
    },
```


A estrutura destino seria então uma pasta referente a uma Powerapp, e dentro uma subpasta para cada ecrã,

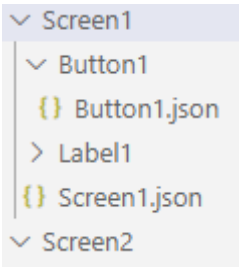


Figura 21 – Estrutura destino dos meta-dados provenientes duma Powerapp


```
{
    "WidgetType": "button",
    "Name": "Button1",
    "Text": "Get started",
    "Color": "RGBA(255,255,255,1)",
    "Fill": "RGBA(0, 0, 0, 0)",
    "X": "82.85714285714286",
    "Y": "1005.857142857143",
    "Width": "472",
    "Height": "99",
    "ZIndex": "4",
    "Size": "25",
    "OnSelect": "Navigate(Landing_Screen)"
}
```

5.2.2 Componente Gerador de Flutter

O componente Gerador de Flutter terá como ficheiros de entrada os ficheiros criados no componente anterior, e através de templates criados com hygen terá como saída ficheiros dart que componham uma aplicação em Flutter.

6 Construção da solução

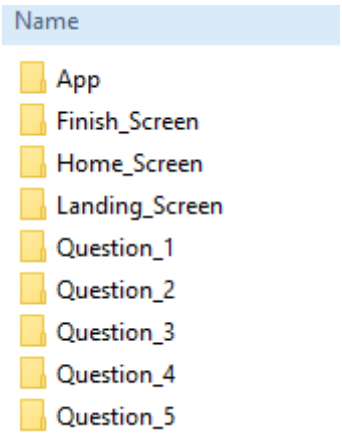
Neste capítulo é apresentada a solução implementada.

6.1 Preparação dos metadados

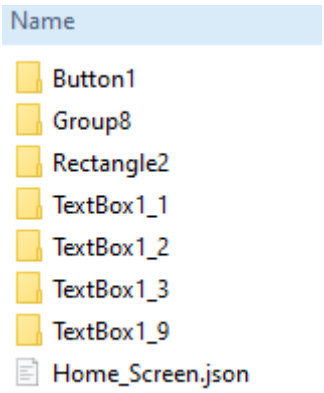
Nesta secção está descrita a transformação feita aos metadados provenientes de uma aplicação desenvolvida em powerapps.

Para realizar esta transformação foi escrito um script em powershell que tivesse como input a pasta com a extensão .msapp e como output uma pasta mais facilmente trabalhável.

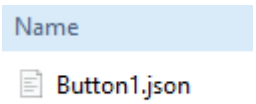
Este script retira da pasta msapp apenas os metadados utilizados para gerar a aplicação nativa, e coloca-os numa pasta com uma subpasta dedicada para cada ecrã.



Sendo que nessa subpasta se encontram outras subpastas, uma para cada componente do ecrã, e ainda um ficheiro com os metadados relativos ao ecrã.



Dentro de cada subpasta correspondente a cada componente, está um ficheiro com as definições de cada um.




```
{
    "WidgetType": "button",
    "Name": "Button1",
    "Text": "Get started",
    "BorderColor": "RGBA(245, 246, 247, 1)",
    "DisabledBorderColor": "ColorFade(Button1.BorderColor, 70%)",
    "PressedBorderColor": "Button1.BorderColor",
    "HoverBorderColor": "ColorFade(Button1.BorderColor, 20%)",
    "BorderStyle": "Solid",
    "Color": "RGBA(255,255,255,1)",
    "DisabledColor": "ColorFade(RGBA(47, 41, 43, 1), 70%)",
    "PressedColor": "Button1.Color",
    "HoverColor": "Button1.Color",
    "Fill": "RGBA(0, 0, 0, 0)",
    "DisabledFill": "ColorFade(RGBA(159, 167, 165, 1), 50%)",
    "PressedFill": "Button1.Fill",
    "HoverFill": "ColorFade(Button1.Fill, 20%)",
    "Font": "Font.Lato",
    "FontWeight": "Lighter",
    "Align": "Align.Center",
    "VerticalAlign": "VerticalAlign.Middle",
    "X": "82.85714285714286",
    "Y": "1005.857142857143",
    "Width": "472",
    "Height": "99",
    "ZIndex": "4",
    "Size": "25",
    "RadiusBottomLeft": "80",
    "RadiusBottomRight": "80",
    "RadiusTopLeft": "80",
    "RadiusTopRight": "80",
    "FocusedBorderThickness": "4",
    "TabIndex": "0",
    "DisplayMode": "DisplayMode.Edit",
    "FocusedBorderColor": "Self.BorderColor",
    "OnSelect": "Collect(ResponsecollectResponses); Navigate(Landing_Scre
en ScreenTransition.Fade)"
}
```

6.2 Preparação da geração do código

Neste capítulo são apresentados os componentes desenvolvidos em flutter bem como cada template de hygen.

6.2.1 Componentes de flutter

Para facilitar o desenvolvimento em flutter são desenvolvidos alguns componentes de uma forma modular, com fim a reduzir o código duplicado. Então para tal foi desenvolvido um componente para cada widget necessário para a migração. Isto facilita a migração bem como o future desenvolvimento da aplicação.

```
import 'package:flutter/material.dart';

class Button extends StatelessWidget {
  final String text;
  final String alignment;
  final Color color;
  final Color fill;
  final double x;
  final double y;
  final double width;
  final double height;
  final double size;
  final void Function(BuildContext) onSelect;

  const Button({
    @required this.text,
    @required this.color,
    @required this.fill,
    @required this.x,
    @required this.y,
    @required this.width,
    @required this.height,
    @required this.size,
    @required this.alignment,
    @required this.onSelect,
  });

  @override
  Widget build(BuildContext context) {
```

```
    return Container(
      child: Positioned(
        left: this.x,
        top: this.y,
        child: Container(child: _alignment(this.alignment, _content(conte
xt))),
      ),
    );
  }

  Widget _content(BuildContext context) {
    return FlatButton(
      height: this.height,
      minWidth: this.width,
      child: Text(
        this.text,
        style: new TextStyle(fontSize: this.size),
      ),
      textColor: this.color,
      color: this.fill,
      onPressed: () {
        this.onSelect(context);
      },
    );
  }
```

Aqui podemos verificar o componente que constrói os botões na app em Flutter, e como podemos ver, este recebe as propriedades vindas da PowerApp e mapeia num widget em Flutter. Para os restantes widgets migrados seguiu-se a mesma lógica.

6.2.2 Templates Hygen

Para utilizar a ferramenta Hygen é necessário escrever templates para gerar os ficheiros. Foram escritos templates para cada um dos widgets bem como para os screens.

```
---
force: true
inject: true
to: app/screens/<%= name %>.dart
after: return Scaffold\( body.  Stack\(children. <Widget>\[
---
<%
if(typeof(PaddingTop)=="undefined"){PaddingTop=5;}
if(typeof(Text)=="undefined"){Text="";}
Color = Color.replace("RGBA", "RGBO");
Fill = Fill.replace("RGBA", "RGBO");
```

```
if(OnSelect.includes("Navigate(")){
  let reg = new RegExp('(.*Navigate.[" "]*)');
  Screen = OnSelect.replace(reg, '');
  reg = new RegExp('([ ].*)');
  Screen = Screen.replace(reg, '');
  OnSelect = "(context) {"+
          "Navigator.push("+
            "context,"+
            "MaterialPageRoute(builder: (context) => "  + Screen + "()),"
+
          ");"+
        "}"
}
%>
new Button(
  text: "<%= Text %>",
  color:  Color.from<%= Color %>,
  alignment: "<%= Align %>",
  fill:  Color.from<%= Fill %>,
  height: <%= Height %>/ 1.5,
  width: <%= Width %>/ 1.63,
  size: <%= Size %>,
  x: <%= X %>/ 1.63,
  y: <%= Y %>/ 1.5,
  onSelect: <%- OnSelect %>,
),
```

6.3 Ferramenta de geração

No fundo a ferramenta de geração é um script em powershell que executa comandos do hygen. Neste script, são percorridos os ficheiros json correspondentes a cada control gerados na primeira fase, para cada ficheiro é executado um comando hygen que utiliza o template do control respetivo e assim gera o código relativo ao widget.

```
Set-Location "C:\Users\joao.santos\Source\Repos\202103.UpsizingLCS"

$powerAppPath = "C:\Users\joao.santos\source\repos\testDir"
$controlsScreens = Get-ChildItem -Path $powerAppPath

Foreach ($Screen in $controlsScreens){
    $file = Get-
Content ($powerAppPath + "\" + $Screen.Name + "\" + $Screen.Name + ".json
")|ConvertFrom-Json
    $widgetType = $file.WidgetType
```

```
    hygen awesome-generator new:$widgetType $Screen.Name --
json  ($powerAppPath + "\" + $Screen.Name + "\" + $Screen.Name + ".json")

    $controlsWidgets = Get-ChildItem -
Path ($powerAppPath + "\"+ $Screen.Name)

    $list = New-Object Collections.Generic.List[PSCustomObject]

    $controlsWidgets.foreach({
        $file2 = Get-
Content ($powerAppPath + "\" + $Screen.Name + "\" + $_.Name + "\" + $_.Na
me + ".json")|ConvertFrom-Json
        $list.Add($file2)
    })

    $controlsWidgets = $list | Sort-Object -Descending -Property ZIndex

    $controlsWidgets.foreach({
        $template = $_.WidgetType
        $template
        hygen awesome-generator new:$template $Screen.Name --
json  ($powerAppPath + "\" +  $Screen.Name + "\" + $_.Name + "\" + $_.Nam
e + ".json")

    })
}
```

6.4 Prova de conceito

Para perceber a possibilidade de construir esta solução foi realizada uma prova de conceito, através de uma powerapp mais simples.

E foi então decidido implementar a migração de uma forma incremental, em que cada incremento se responsabilizaria por um problema a solucionar. O primeiro foi então migrar a componente gráfica da powerapp de seguida migrar a componente lógica e por fim migrar o acesso às fontes de dados.

6.4.1 Geração dos componentes gráficos

Para migrar a componente gráfica foi desenvolvida uma powerapp com os 5 controls mais populares que são o rectangulo, o botão, caixa de texto, a caixa de inserção de texto, e o selecionador de data.

Foram então desenvolvidos os componentes e os templates necessários para esta migração.

Foi realizada uma proporção para os componentes ficassem com o tamanho e posição idêntica.

E foi então possível fazer a migração deste componente.

6.4.2 Geração da lógica da aplicação

Para migrar a lógica foi escolhida uma das funções mais utilizadas nas powerapps, a função Navigate, responsável por fazer a navegação entre screens.

Para isto foi realizado um regex para detetar em cada formula da powerapp se esta contém a função apontada, e caso esta função exista é injetado na app em flutter uma função global encarregue de fazer a navegação.

Aqui foi então acrescentado um screen novo a powerapp já utilizada para fazer a migração anterior, e no botão do primeiro screen foi incluído a navegação para o segundo screen e no segundo screen um botão para voltar ao primeiro screen.

Foi então aplicado a migração e a app em flutter foi capax de navegar entre os dois ecrãs.

6.4.3 Geração do acesso às fontes de dados

Para verificar esta migração foi implementado na powerapp uma ligação a uma lista sharepoint. Esta é uma das fontes de dados mais utilizada pela equipa de low-code, poi permite uma aproximação aos utilizadores com os seus dados.

Para tal é necessário registar app no azure e permitir a o acesso via graphapi, de seguida +e passada a ligação a lista bem como a password para a app em flutter para abrir o autenticador onde o utilizador introduz as suas credenciais e assim ter acesso à lista de sharepoint.

7 Avaliação e Experimentação

Neste capítulo são apresentadas as métricas de avaliação e especificação de hipóteses, os indicadores e fontes de informação e por último a metodologia de avaliação

7.1 Métricas de avaliação e especificação de hipóteses

As hipóteses apontadas foram as seguintes:

- É possível gerar um modelo com os dados relativos a uma aplicação Low-code.
- É possível gerar uma aplicação nativa a partir de um modelo.

7.2 Indicadores e fontes de informação

Os indicadores e fontes de informação serão aplicações já desenvolvidas em plataformas Low-code e migradas para apps nativas, e assim perceber o quão idêntico é a app gerada pelo gerador à app desenvolvida pela equipa de desenvolvedores.

7.3 Metodologia de avaliação

Os dois métodos que serão utilizados para a validação das hipóteses são testes e simulações.
Os testes servirão para detetar possíveis falhas no sistema antes de sua utilização a um nível de produção. E as simulações serão executadas sobre apps já existentes e já migradas por equipas de desenvolvimento e assim perceber quão idêntico é a app gerada pelo gerador à app desenvolvida pela equipa de desenvolvedores.

Os tipos de testes a realizar são:

- Testes Unitários: Este tipo de testes verifica o funcionamento de pequenas partes do código implementado.

- Testes de Aceitação: Estes testes verificam os requisitos pretendidos do sistema, tanto ao nível funcional como ao nível de desempenho.

- Testes de Integração: Estes testes irão validar a integração do sistema que gera o modelo com o sistema que gera a app nativa.

7.4 Testes

7.4.1 Testes Unitários

Os testes unitários são testes que se focam apenas num componente do programa implementado.

Neste projeto foram realizados testes unitários ao componente de extração de meta-dados, tendo como entrada ficheiros json com a estrutura dos ficheiros provenientes da pasta msaap e como resultado esperado ficheiros json com a estrutura da pasta destino com os mesmos valores que a pasta msaap.

Foram também realizados testes unitários ao componente de geração de flutter, com dados de entrada ficheiros json e como resultados esperados ficheiros dart que compõem widgets de Flutter com as mesmas propriedades que os ficheiros json de origem.

7.4.2 Testes de integração

Os testes de integração conferem o bom funcionamento entre mais do que um componente.

Neste projeto foram realizados testes de integração entre o componente de extração dos meta-dados de uma powerapp e o componente de geração de Flutter. Estes testes tiveram como dados de entrada uma pasta msaap e como resultados esperados ficheiros dart que compõem widgets em Flutter.

7.4.3 Testes de aceitação

Os testes de aceitação verificam os requisitos do cliente, dando garantias de que os objetivos foram cumpridos.

Os testes realizados foram os seguintes:

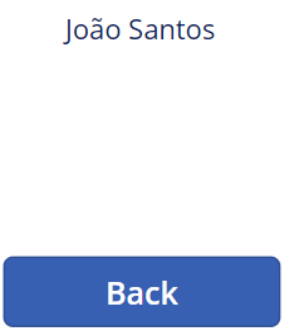
Requisito	*Resultado*
Extração dos meta-dados de uma PowerApp	Cumprido
Geração da componente gráfica em Flutter	Cumprido
Geração da componente lógica em Flutter	Cumprido
Geração de um data source em Flutter	Cumprido

7.5 Descrição das experiências

Nesta secção são apresentadas duas experiências feitas com a ferramenta desenvolvida, a primeira com uma aplicação denominada de SimpleApp que foi também a app que serviu de guia para a implementação, e a segunda denominada Employee Engagement Survey, que é uma app de um questionário aos empregados.

7.5.1 SimpleApp

A SimpleApp como o nome indica é uma aplicação simples, onde se pretendia apenas migrar componentes básicos para perceber o comportamento da ferramenta desenvolvida.



Esta Powerapp tem apenas os seguintes 2 ecrãs.

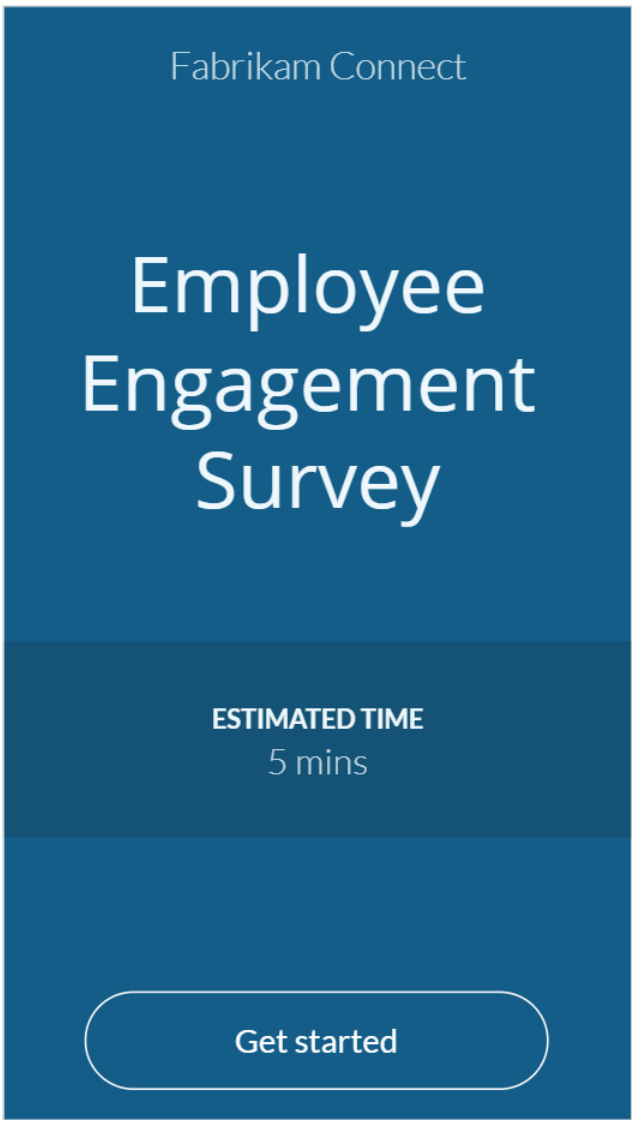
E o resultado da migração foi o seguinte:

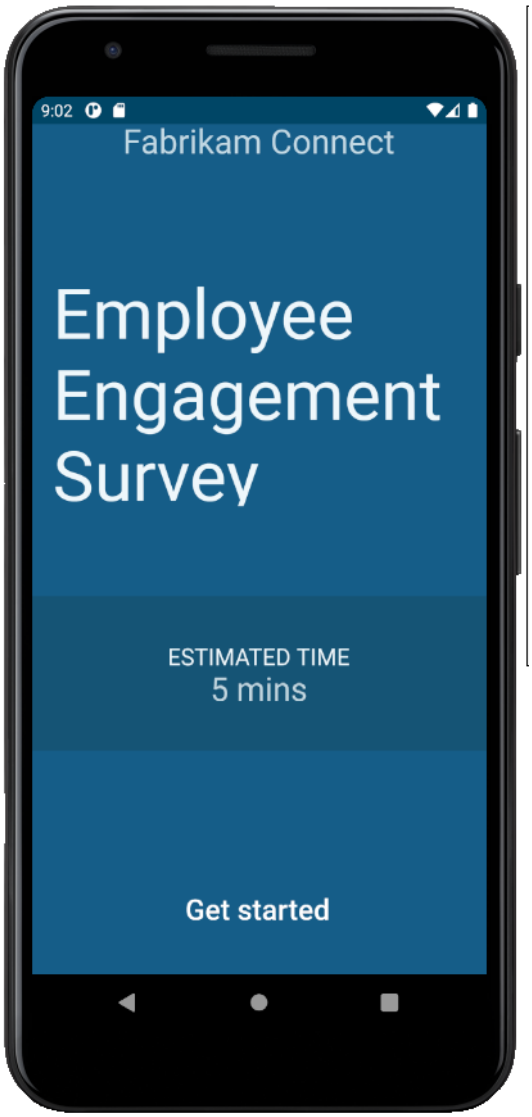
Esta tabela mostra o tempo que a app demorou a ser gerada, tempo que levaria a ser implementada de uma forma manual, e por último o tempo que levou a ser implementada a ferramenta da migração.

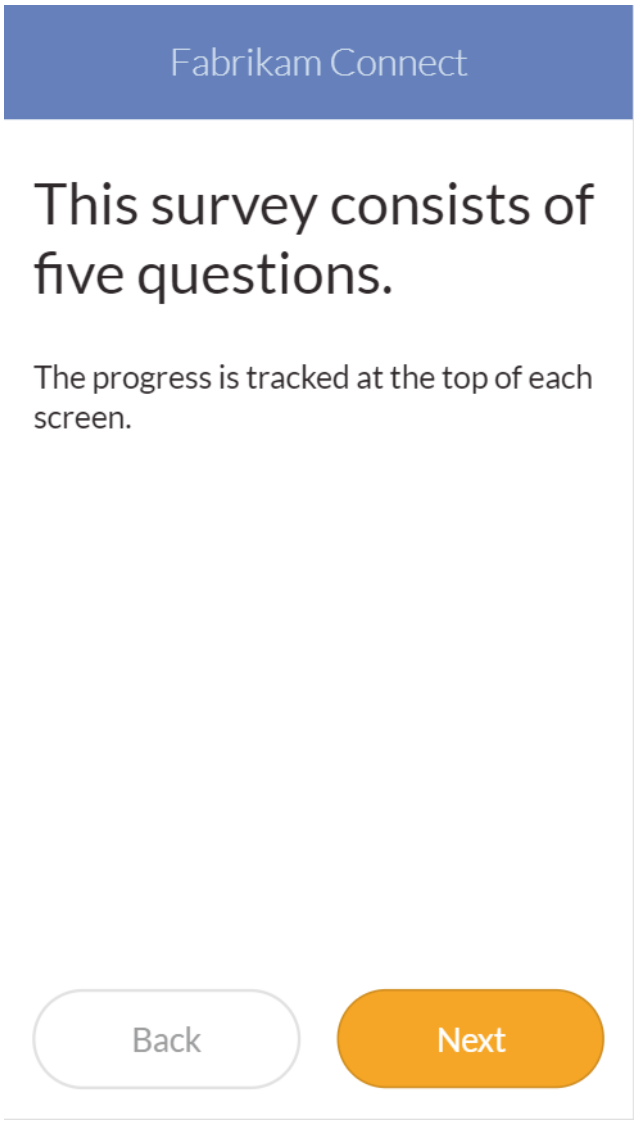
Tempo de geração da app	Tempo de implementação da app	Tempo de implementação da ferramenta
~1 min	~3 h	~320 horas

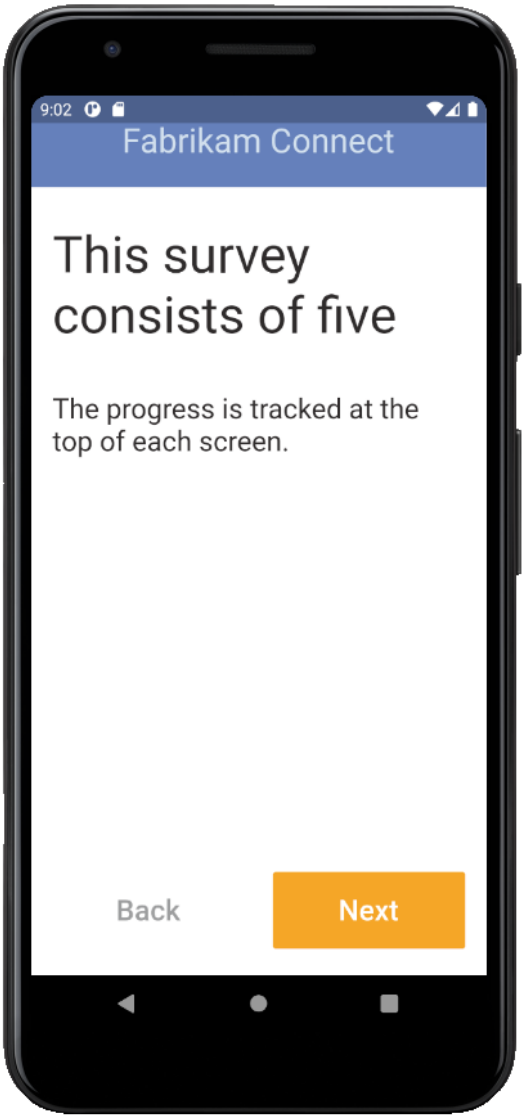
7.5.2 Employee Engagement Survey

Esta é uma app de um questionário aos empregados, que já contém mais controls para além dos populares, e nas imagens seguintes podemos ver a comparação entre a Powerapp e a aplicação gerada em Flutter.

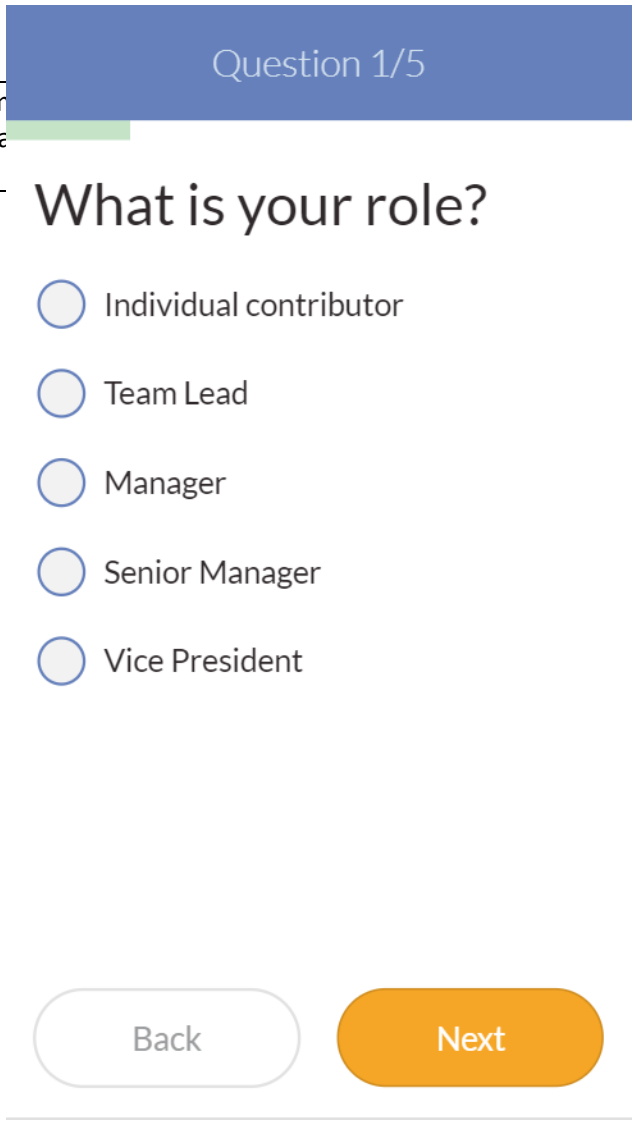

Fabrikam Connect
Employee Engagement Survey
ESTIMATED TIME
5 mins
Get started

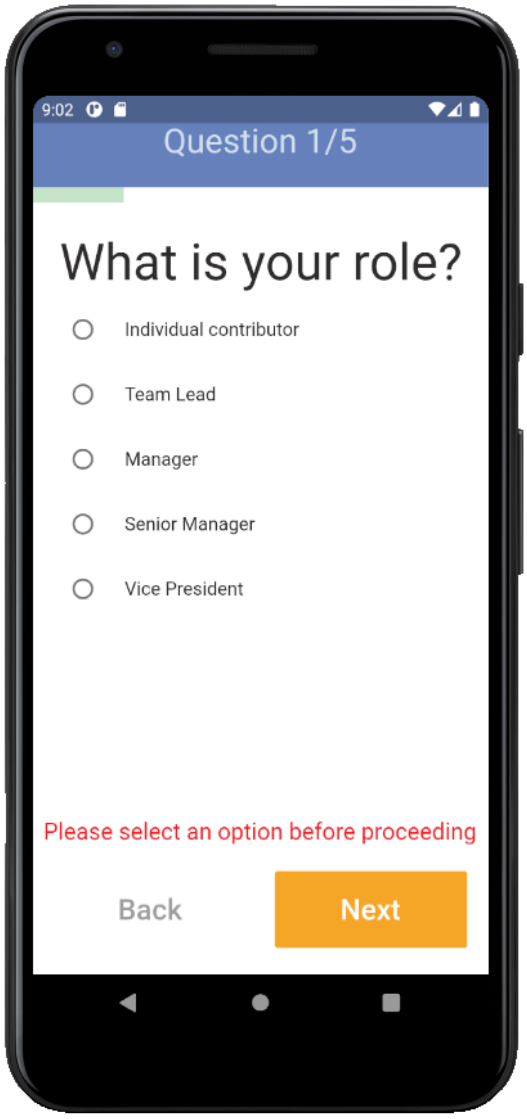

9:02
Fabrikam Connect
Employee Engagement Survey
ESTIMATED TIME
5 mins
Get started


Fabrikam Connect
This survey consists of five questions.
The progress is tracked at the top of each screen.
Back
Next


9:02
Fabrikam Connect
This survey consists of five
The progress is tracked at the top of each screen.
Back
Next

Tem
gera


Question 1/5
What is your role?
Individual contributor
Team Lead
Manager
Senior Manager
Vice President
Back
Next


9:02
Question 1/5
What is your role?
Individual contributor
Team Lead
Manager
Senior Manager
Vice President
Please select an option before proceeding
Back
Next

~3 min	~20 horas	~80 horas

Nesta tabela podemos verificar o tempo de geração da app, o tempo que levaria a ser implementada, e o tempo que levou a ser implementada a transformação dos componentes que a ferramenta de migração ainda não era capaz de transformar.

7.6 Resumo dos resultados

Após a realização dos testes e das experiências, os resultados mostram que a ferramenta é capaz de migrar aplicações desenvolvidas em Powerapps para aplicações em Flutter.

É ainda percetível que o tempo de implementação de transformações de novos controls será menor, devido a ser só necessário seguir o padrão das transformações já implementadas.

Os resultados mostram ainda que a ferramenta poupa imenso tempo aos desenvolvedores de software, como era esperado.

8 Conclusões

Este capítulo resume o documento expondo os objetivos alcançados, as suas limitações e trabalho futuro, e ainda a apreciação final.

8.1 Objetivos alcançados

O objetivo principal deste projeto era provar que é possível automatizar o processo de migração do desenvolvimento low-code para o desenvolvimento tradicional, mais especificamente de Powerapps em Flutter.

Para atingir esse objetivo foram traçados objetivos adjacentes:

- Sintetizar o conhecimento já existente nas áreas em que o projeto se insere:
    - Plataformas de desenvolvimento de soluções Low-code
    - Ferramentas de geração de código
    - Desenvolvimento Cross Platform Mobile
- Desenvolver uma solução que permita criar um modelo, com toda a informação necessária relativa a uma app em soluções Low-code, adaptado para o gerador da app nativa.
- Desenvolver uma solução que a partir de um modelo gera uma aplicação nativa.
- Realizar os testes necessários à validação da solução desenvolvida.
- Avaliar a solução através dos resultados obtidos nos testes realizados.

8.2 Limitações e trabalho futuro

- Perceber quais os próximos controls mais importantes a migrar e desenhar os seus componentes e templates.
- Perceber quais as próximas funções mais importantes a migrar.
- Perceber quais os conectores mais importantes a migrar.

8.3 Apreciação final

A documentação executada neste projeto mostra a viabilidade, e como é possível fazer uma ferramenta capaz de migrar soluções low-code em soluções nativas.

Referências

Acerbis, R., Bongio, A., Brambilla, M., Tisi, M., Ceri, S., & Tosetti, E. (2007). Developing eBusiness Solutions with a Model Driven Approach: The Case of Acer EMEA. Em L. Baresi, P. Fraternali, & G.-J. Houben (Eds.), *Web Engineering* (pp. 539–544). Springer. https://doi.org/10.1007/978-3-540-73597-7_51

Belliveau, P., Griffin, A., & Somermeyer, S. (2004). *The PDMA ToolBook 1 for New Product Development*. John Wiley & Sons.

bliki: UmlMode. (sem data). martinfowler.com. Obtido 29 de Julho de 2021, de https://martinfowler.com/bliki/UmlMode.html

Brambilla, M., Cabot, J., & Wimmer, M. (2017). Model-Driven Software Engineering in Practice, Second Edition. *Synthesis Lectures on Software Engineering*, *3*(1), 1–207. https://doi.org/10.2200/S00751ED2V01Y201701SWE004

Build Applications Fast, Right and For the Future | OutSystems. (sem data). Obtido 7 de Março de 2021, de https://www.outsystems.com/

FlutterFlow—Build Flutter Apps Visually. (sem data). Obtido 28 de Junho de 2021, de https://flutterflow.io/

Generators | Hygen. (sem data). Obtido 7 de Março de 2021, de https://www.hygen.io//docs/generators

Işitan, M., & Koklu, M. (2020). Comparison and Evaluation of Cross Platform Mobile Application Development Tools. *International Journal of Applied Mathematics Electronics and Computers*, *8*(4), 273–281. https://doi.org/10.18100/ijamec.832673

Jobe, W. (2013). Native Apps Vs. Mobile Web Apps. *International Journal of Interactive Mobile Technologies (iJIM)*, *7*, 27–32. https://doi.org/10.3991/ijim.v7i4.3226

Kühne, T. (2006). Matters of (Meta-) Modeling. *Software & Systems Modeling*, *5*(4), 369–385. https://doi.org/10.1007/s10270-006-0017-9

Mendix—Go Make It. (sem data). Mendix. Obtido 7 de Março de 2021, de https://www.mendix.com/

Olivé, A. (2004). On the Role of Conceptual Schemas in Information Systems Development. Em A. Llamosí & A. Strohmeier (Eds.), *Reliable Software Technologies—Ada-Europe 2004* (pp. 16–34). Springer. https://doi.org/10.1007/978-3-540-24841-5_2

Paul Vincent, Kimihiko Iijima, Mark Driver, Jason Wong, & Yefim Natis. (2019). *Magic Quadrant for Enterprise Low-Code Application Platforms*.

Pearson, M., Knight, B., Knight, D., & Quintana, M. (2020). Power Platform Integration in Power Apps. Em M. Pearson, B. Knight, D. Knight, & M. Quintana (Eds.), *Pro Microsoft Power Platform: Solution Building for the Citizen Developer* (pp. 333–342). Apress. https://doi.org/10.1007/978-1-4842-6008-1_26

Rich, Nick and Holweg, Matthias. (2000). *Value analysis*.

Richard Paige, Jordi Cabot, Marco Brambilla, Louis Rose, & James H. Hill. (2014). *Model-Driven Engineering on and for the Cloud*.

Richardson, C., & Rymer, R. (2016). *The Forrester Wave™: Low-Code Development Platforms, Q2 2016*.

Rymer, J. R., & Koplowitz, R. (2019). *The Forrester Wave™: Low-Code Development Platforms For AD&D Professionals, Q1 2019*. 17.

Sahay, A., Indamutsa, A., Ruscio, D. D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 171–178. https://doi.org/10.1109/SEAA51224.2020.00036

Sendall, S., & Kozaczynski, W. (2003). Model transformation: The heart and soul of model-driven software development. *IEEE Software*, *20*(5), 42–45. https://doi.org/10.1109/MS.2003.1231150

Stack Overflow Developer Survey 2020. (sem data). Stack Overflow. Obtido 7 de Março de 2021, de https://insights.stackoverflow.com/survey/2020/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2020

Stack Overflow Trends. (sem data). Obtido 7 de Março de 2021, de https://insights.stackoverflow.com/trends?tags=

The web's scaffolding tool for modern webapps | Yeoman. (sem data). Obtido 7 de Março de 2021, de https://yeoman.io/

wibjorn. (sem data). *Developer tools, technical documentation and coding examples*. Obtido 7 de Março de 2021, de https://docs.microsoft.com/en-us/

Apêndice A

SimpleApp – PowerApps

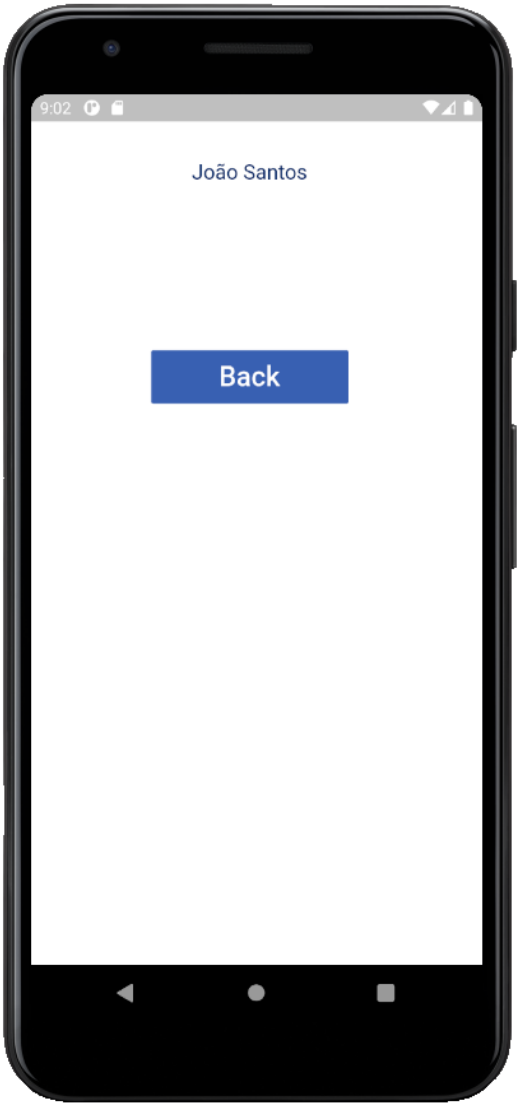

Simple Title
Text input
5/28/2021
Next

João Santos

Back

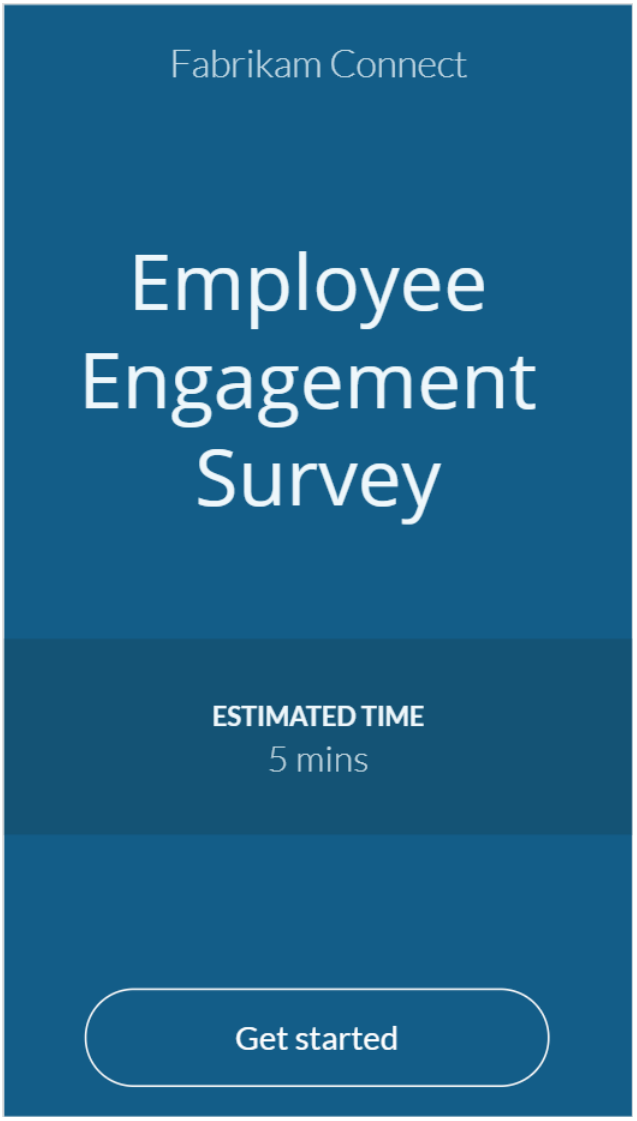
Apêndice B

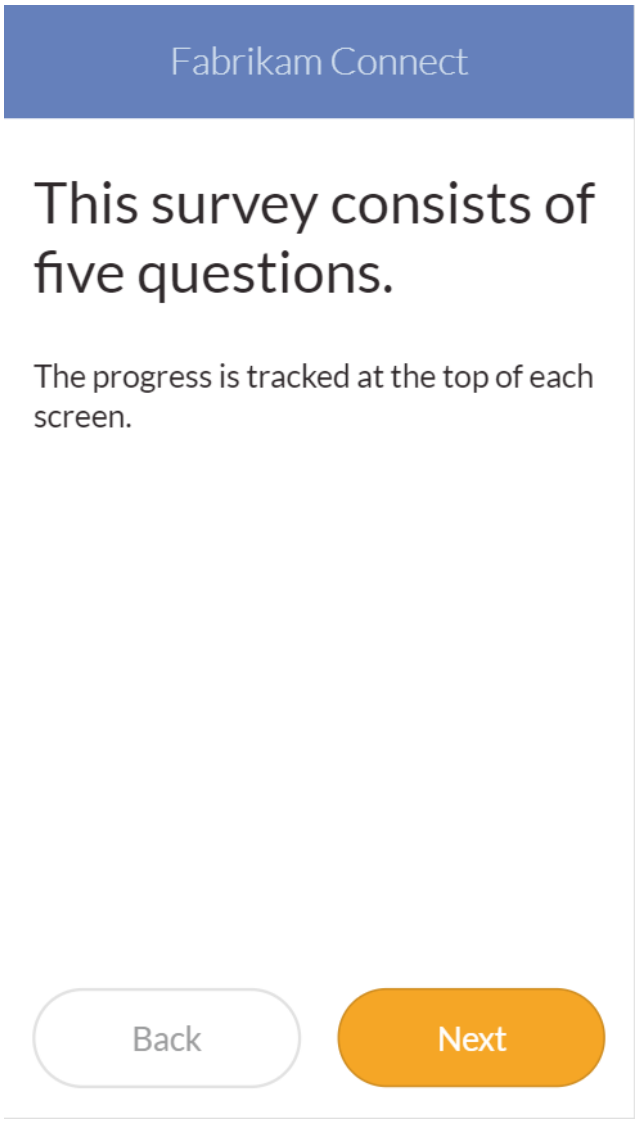
SimpleApp – Flutter

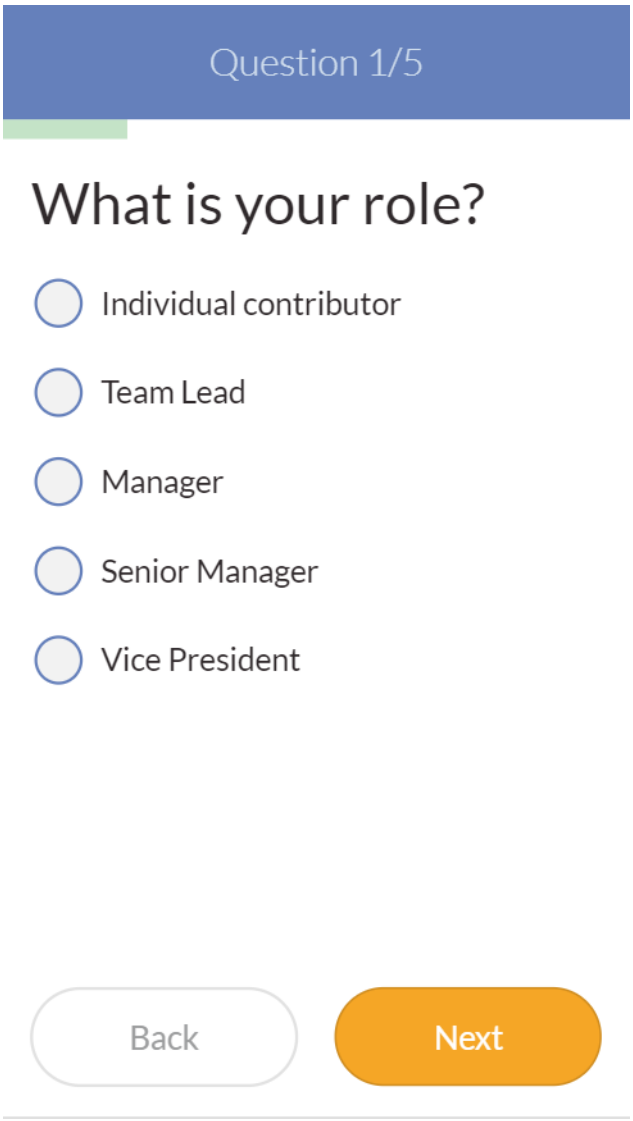

9:02
João Santos
Back

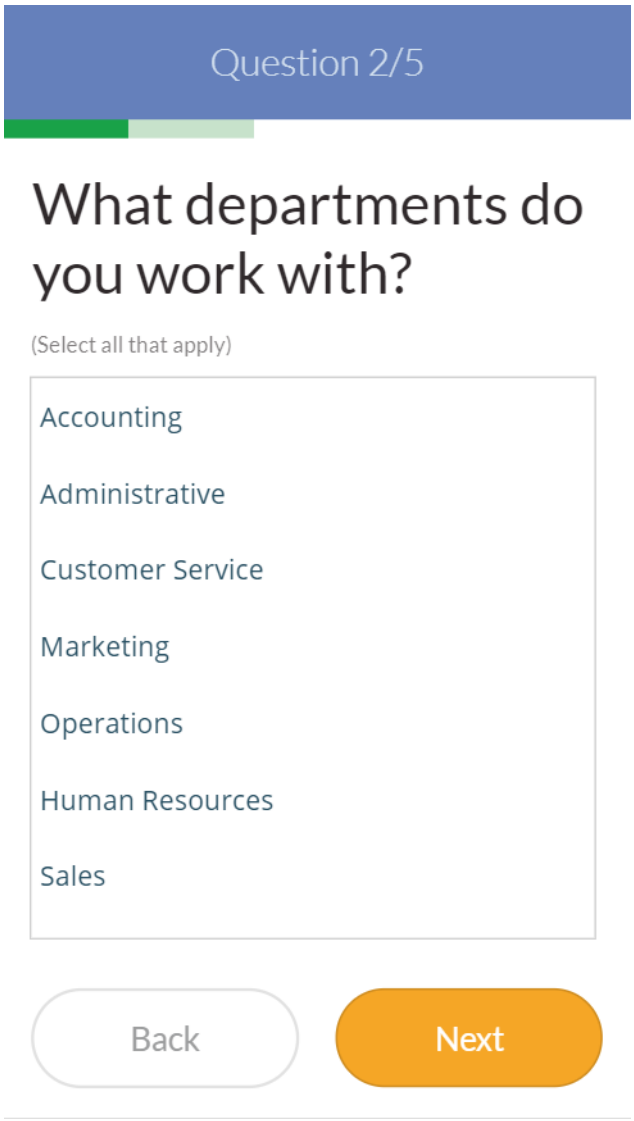
Apêndice C

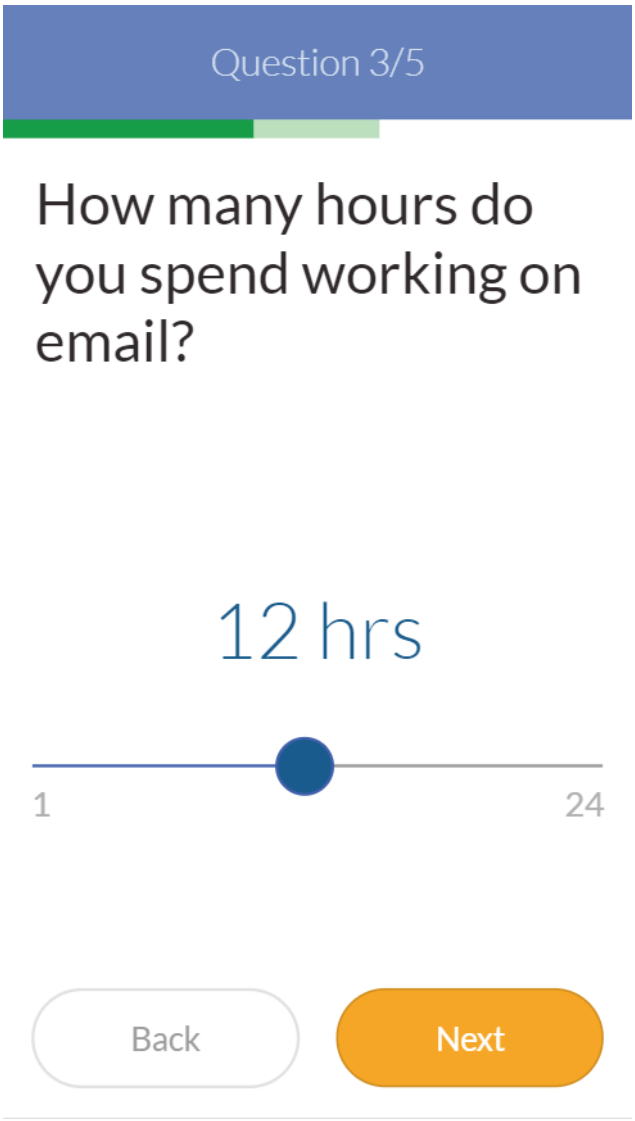
Employee Engagement Survey – PowerApps




Fabrikam Connect
This survey consists of five questions.
The progress is tracked at the top of each screen.
Back
Next


Question 1/5
What is your role?
Individual contributor
Team Lead
Manager
Senior Manager
Vice President
Back
Next


Question 2/5
What departments do you work with?
(Select all that apply)
Accounting
Administrative
Customer Service
Marketing
Operations
Human Resources
Sales
Back
Next

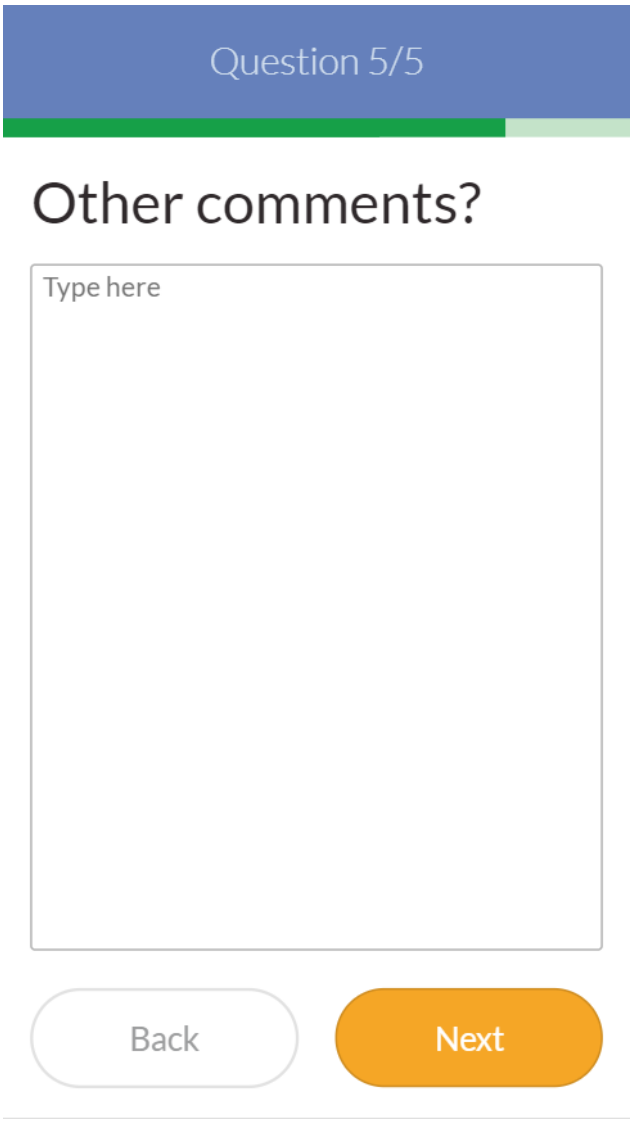

Question 3/5
How many hours do you spend working on email?
12 hrs
1
24
Back
Next

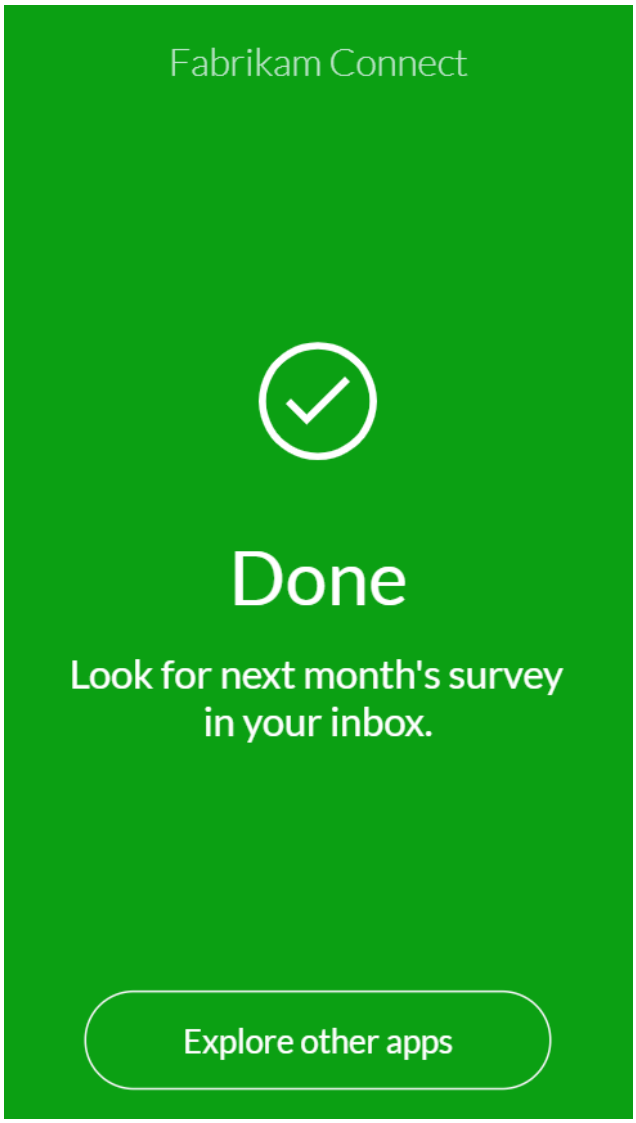
Question 4/5

Do you see yourself working here in one year?

- Yes
- No

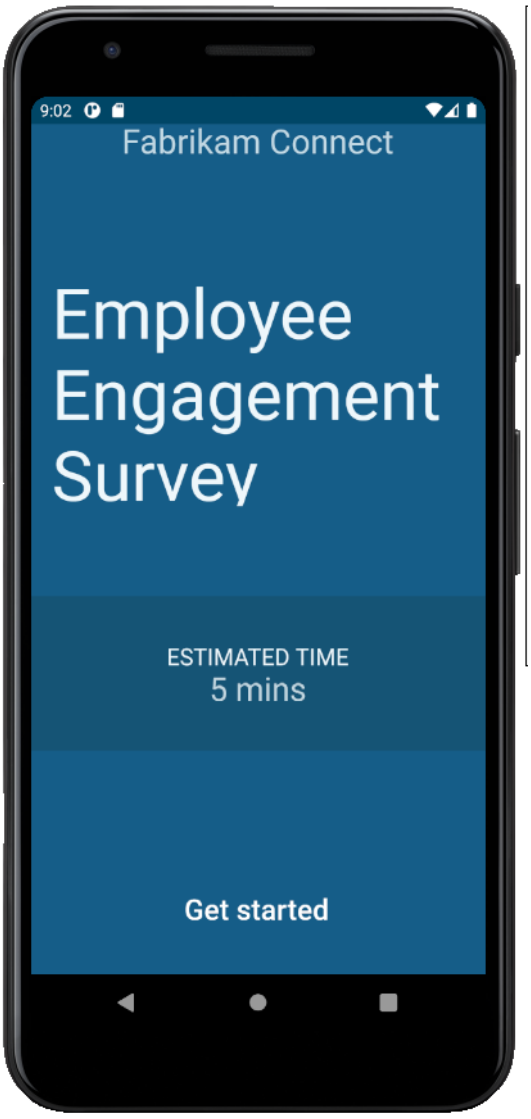
Back Next

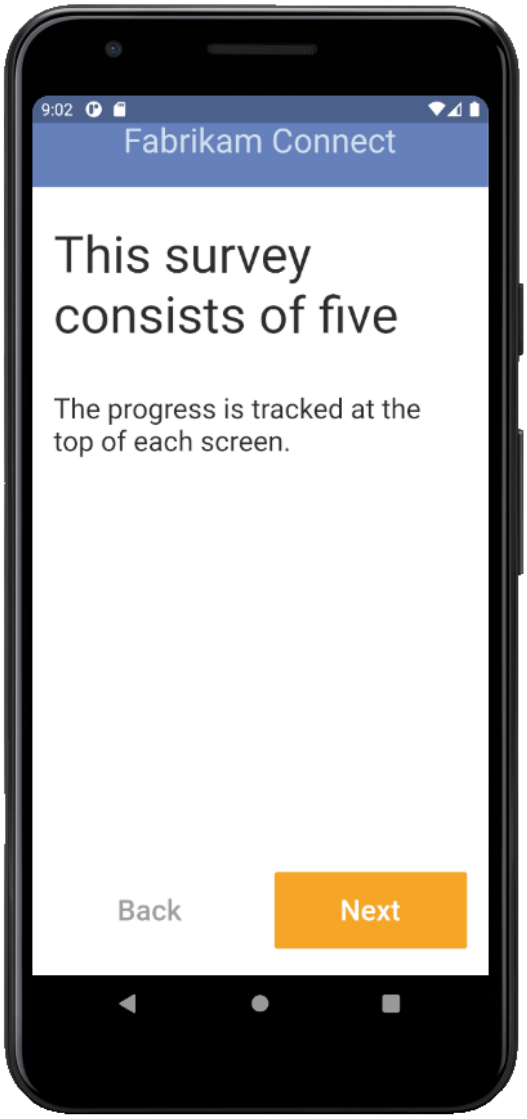

Question 5/5
Other comments?
Type here
Back
Next

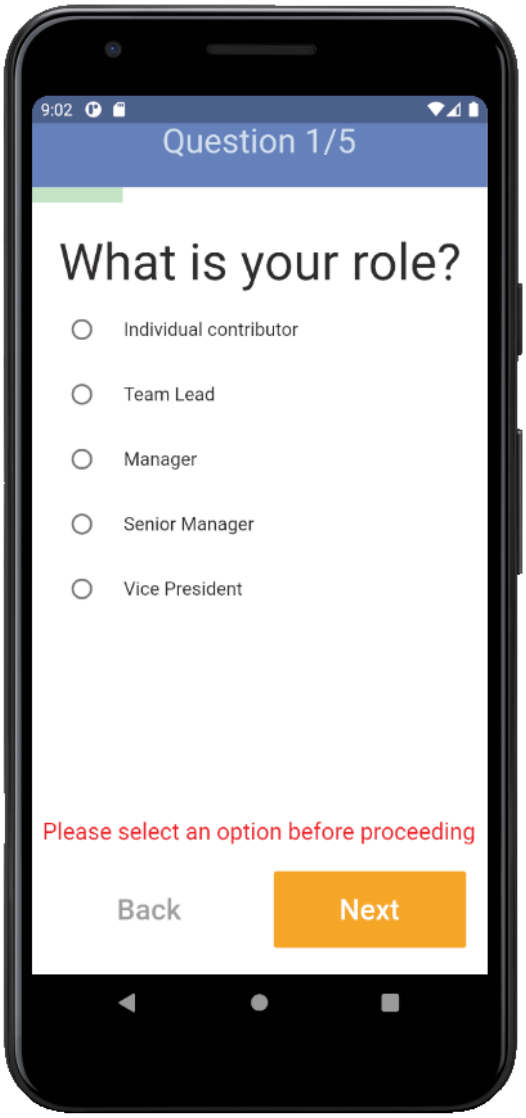

Fabrikam Connect
Done
Look for next month's survey in your inbox.
Explore other apps

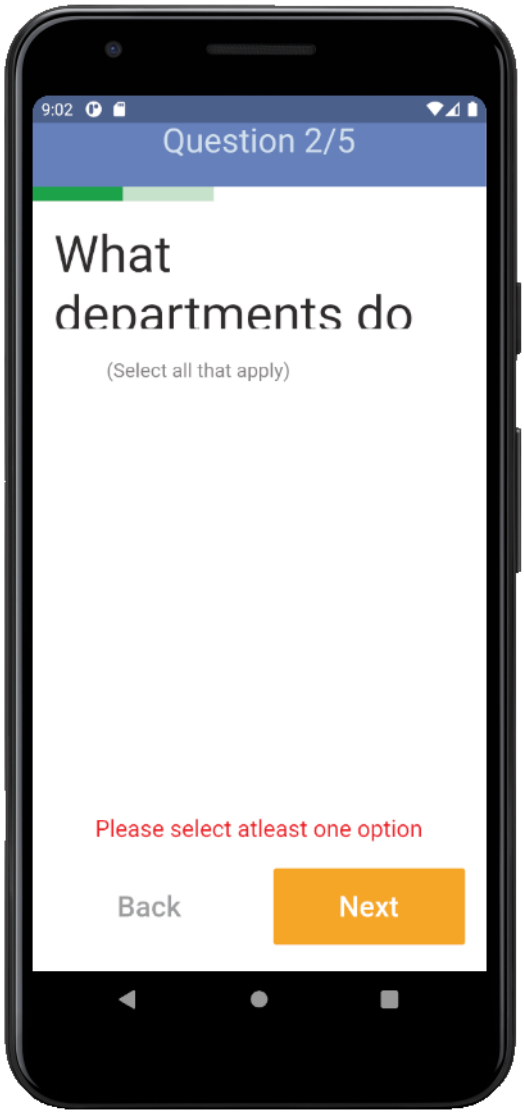
Apêndice D

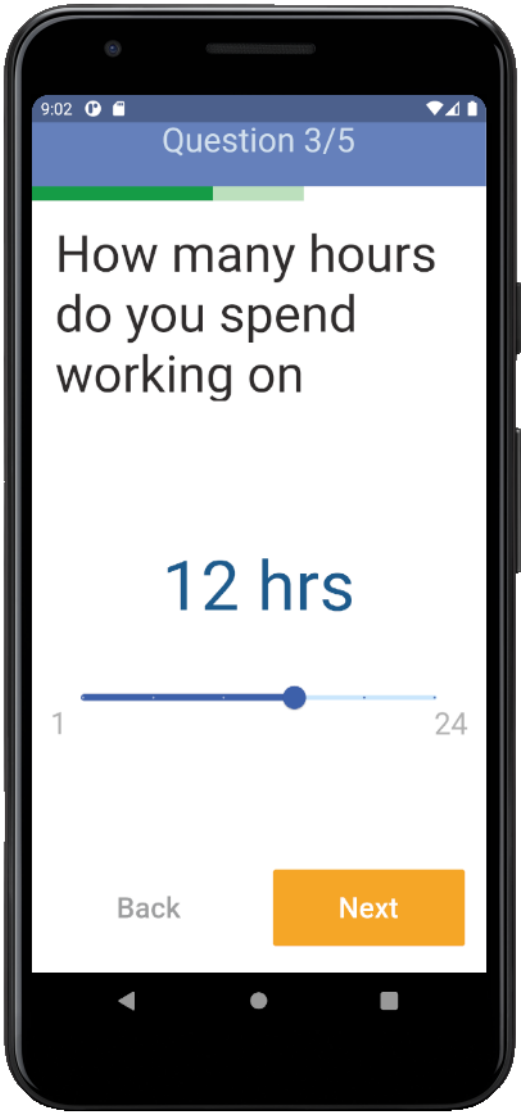
Employee Engagement Survey – Flutter




9:02
Fabrikam Connect
This survey consists of five
The progress is tracked at the top of each screen.
Back
Next

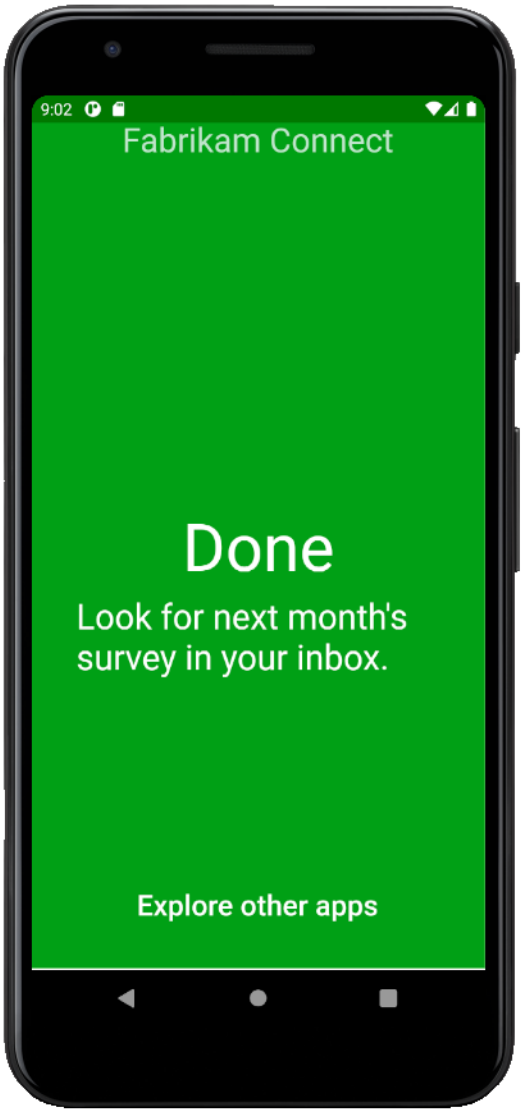

9:02
Question 1/5
What is your role?
Individual contributor
Team Lead
Manager
Senior Manager
Vice President
Please select an option before proceeding
Back
Next


9:02
Question 2/5
What
departments do
(Select all that apply)
Please select atleast one option
Back
Next


9:02
Question 3/5
How many hours do you spend working on
12 hrs
1
24
Back
Next


9:02
Question 4/5
Do you see yourself working here in one year?
Yes
No
If no, please explain *
Please fill out the required field*
Please select an option before proceeding
Back
Next


9:02
Question 5/5
Other comments?
Back
Next


9:02
Fabrikam Connect
Done
Look for next month's survey in your inbox.
Explore other apps