POLITÉCNICO GUARDA

Escola Superior de Tecnologia e Gestão

NOESIS
POWER PLATFORM, DATAVERSE E DYNAMICS 365

RELATÓRIO DE ESTÁGIO
PARA OBTENÇÃO DO GRAU DE LICENCIADO EM ENGENHARIA INFORMÁTICA

Nuno Galinho
Julho / 2023

POLITÉCNICO GUARDA

Escola Superior de Tecnologia e Gestão

NOESIS
POWER PLATFORM, DATAVERSE E DYNAMICS 365

RELATÓRIO DE ESTÁGIO
PARA OBTENÇÃO DO GRAU DE LICENCIADO EM ENGENHARIA INFORMÁTICA

Professora Orientadora: Professora Coordenadora Maria Clara Silveira

Nuno Galinho

Julho / 2023

Agradecimentos

Um profundo agradecimento á minha família em especial aos meus tios e irmão Bruno Cerdeira, que sempre esteve ao meu lado e sempre me apoiou sem julgamento.

À Professora Doutora Maria Clara Silveira, minha orientadora, cuja presença, incentivo e disponibilidade incondicionais foram essenciais para ultrapassar as dificuldades deste conturbado processo.

À Noesis Portugal, pela oportunidade de estágio, em especial ao meu tutor Abel Espirito Santo e meu formador João Mascarenhas, pela orientação, disponibilidade e integração estimulante no ambiente de trabalho.

A todos os meus professores e colegas desta enorme ESTG por terem feito parte do meu crescimento, como aluno e como pessoa, vocês fazem parte da minha identidade.

Por fim, para os meus pais e avós, Abel Bernardo do Fojo e Maria Ilidia Chantre:

Ficha de Identificação

Aluno
Nome: Nuno Filipe do Fojo Galinho
Número: 1008043
Licenciatura: Engenharia Informática

Estabelecimento de Ensino
Instituto Politécnico da Guarda (IPG)
Escola Superior de Tecnologia e Gestão (ESTG)

Entidade Acolhedora do Estágio
Nome: Noesis
Morada: Rua Tomás Da Fonseca, Torre E 14º Lisboa 1600-209
Contacto Telefónico: 21 423 5430
Duração do Estágio: 14/11/2023 - 13/01/2023

Supervisor de Estágio
Nome: Abel Espírito Santo
Função: Senior Manager

Docente Orientador de Estágio
Nome: Maria Clara Silveira
Grau Académico: Doutoramento em Engenharia Eletrotécnica e Computadores

Resumo

Este relatório apresenta o projeto desenvolvido como parte da unidade curricular Projeto de Informática da Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda. O projeto foi realizado em contexto de estágio na empresa Noesis.

O objetivo do projeto foi adquirir conhecimentos sobre Power Platform, Dataverse e Dynamics 365 e em seguida, aplicar esses conhecimentos no desenvolvimento de uma *Canvas App* de demonstração chamada *Event Manager*, que permite gerir eventos, seus organizadores, tarefas atribuídas e clientes. Através do desenvolvimento da aplicação, foi possível aplicar de forma concreta e funcional vários conceitos e explorar o potencial dessas tecnologias para soluções empresariais.

Palavras-Chave: *Power Platform*, *Dataverse*, *Dynamics 365, Canvas App, Event Manager*

Abstract

This report describes the project carried out within the scope of the Informatics Project, of the Computer Engineering Degree from the School of Technology and Management of the Polytechnical Institute of Guarda. The project, in the context of an internship, was carried out at the company Noesis.

The objective of the project was to acquire knowledge about Power Platform, Dataverse and Dynamics 365 and then apply this knowledge in the development of a demo Canvas App called Event Manager, which allows managing events, their organizers, assigned tasks and clients. Through the development of the application, it was possible to apply in a concrete and functional way several concepts and explore the potential of these technologies for business solutions.

Keywords: Power Platform, Dataverse, Dynamics 365, Canvas App

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

Este relatório descreve o trabalho realizado pelo aluno Nuno Filipe do Fojo Galinho durante seu estágio na empresa Noesis. O estágio decorreu como parte da unidade curricular Projeto de Informática, do 3º ano da Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda.

1.1 Enquadramento e motivação

As soluções de negócio modernas estão cada vez mais dependentes de tecnologias avançadas para impulsionar a inovação, a eficiência e a produtividade. Nesse contexto, a *Dataverse*, *Power Platform* e *Dynamics 365* emergem como componentes essenciais, fornecendo ferramentas poderosas para as empresas alcançarem seus objetivos e enfrentarem os desafios do mercado atual.

A *Dataverse*, como uma plataforma de dados, oferece um ambiente seguro e escalável para o armazenamento, gestão e integração de informação. Ao centralizar e organizar os dados duma organização, a *Dataverse* facilita a criação de aplicações personalizadas e fluxos de trabalho pré-construídos. Isso permite que as empresas sejam mais ágeis na adaptação às mudanças do mercado e nas necessidades específicas do negócio.

Referindo *Microsoft* [1] *Power Platform*, é uma *suíte* de ferramentas composta por *Power Apps*, *Power Automate*, *Power BI e Power Virtual Agents*. Com essas ferramentas é possível criar aplicações, automatizar processos e analisar dados sem a necessidade de conhecimentos avançados em programação.

O *Dynamics 365* é uma *suíte* de aplicações inteligentes baseados na *Cloud* que abrange diversas áreas de negócio, como vendas, atendimento ao cliente, operações, finanças e recursos humanos.

Estas ferramentas são importantes e relevantes para soluções de negócios porque permitem que as empresas façam a gestão dos seus dados e processos de maneira eficiente e escalável. Elas oferecem recursos avançados para armazenamento, análise e automação de dados, permitindo que as empresas tomem decisões informadas e melhorem seus processos. Além disso, elas são altamente personalizáveis e podem ser adaptadas às necessidades específicas de cada empresa. Promovem a colaboração e a integração entre diferentes departamentos e equipas, resultando uma maior produtividade, agilidade e eficácia das operações empresariais.

Além disso, a *Dataverse*, a *Power Platform* e a *Dynamics 365* possibilitam a automação de processos, reduzindo o tempo e os recursos necessários para tarefas manuais. Isso permite que as empresas se concentrem em atividades de maior valor, impulsionando a inovação e o crescimento.

1.2 Caraterização sumária da instituição de acolhimento

A Noesis Portugal é uma empresa de consultoria tecnológica líder em Portugal. Fundada em 1995, possui um vasto conhecimento e experiência em diversas áreas, incluindo transformação digital, desenvolvimento de software e soluções de negócios.

A Noesis estabeleceu parcerias estratégicas com empresas de renome, como a Microsoft, a OutSystems e a Oracle, fortalecendo sua oferta de serviços e soluções inovadoras.

Com escritórios em Lisboa, Porto e Coimbra, a Noesis está presente em todo o país, oferecendo suporte e *expertise* em projetos de Tecnologias de Informação para empresas de diferentes setores [2].

1.3 Objetivos

O objetivo principal deste relatório consiste em descrever de forma abrangente o trabalho desenvolvido durante o estágio realizado na empresa Noesis Portugal. Ao longo desse curto período, foram adquiridos conhecimentos acerca dos conceitos fundamentais do *Dynamics 365*, *Dataverse* e *Power Platform*, que são elementos-chave no âmbito das soluções empresariais.

Dentre os tópicos abordados, destacam-se a importância dos grupos de segurança (*security groups*), das funções de segurança (*security roles*), assim como o papel das unidades de negócio (*business units*) e a relevância da auditoria (*auditing*). Foram também explorados temas relacionados aos ambientes de trabalho (*environments*), soluções (*solutions*), *plugins* e módulos do *Dynamics 365*.

Para exemplificar e aplicar alguns destes conceitos, foi desenvolvida uma *canvas app* na *Power Platform*, *Power apps*, com o nome *"Event Manager"*. É uma aplicação simples, desenvolvida apenas para efeitos de exemplificação e tem por função gerir eventos. A aplicação permite criar eventos, registar clientes, associar eventos a organizadores, associar eventos a clientes e atribuir tarefas aos organizadores. Nesse contexto, foram definidos três tipos de organizadores: *senior organizer, executive organizer e junior organizer*, cada um com funções e permissões específicas. O *executive organizer* possui acesso completo, podendo criar, atualizar e apagar, eventos, tarefas e clientes, pode ainda atribuir eventos a cliente e organizadores a eventos. O *executive organizer* tem permissão para criar e atualizar tarefas, enquanto o *junior organizer* possui apenas acesso de consulta.

Adicionalmente, foram implementados dois plugins na aplicação: O primeiro, cria automaticamente um evento denominado "Novo Cliente!" e uma tarefa "Contactar Cliente!" direcionada ao Organizador Sênior, sempre que um novo cliente é adicionado. O segundo, exclui todas as tarefas relacionadas a um evento quando este é apagado.

Assim, este relatório visa apresentar, de forma clara e detalhada, os conceitos aprendidos durante o estágio, bem como demonstrar a sua aplicação prática por meio da aplicação desenvolvida. O objetivo é fornecer um panorama abrangente sobre o *Dataverse*, *Power Platform* e *Dynamics 365*, evidenciando a sua importância e relevância como ferramentas poderosas para soluções empresariais.

1.4 Estrutura do relatório

O presente relatório encontra-se dividido em seis capítulos. O primeiro capítulo apresenta a introdução ao projeto, bem como os objetivos, o enquadramento e motivação. No segundo capítulo aborda-se de forma detalhada os conceitos relacionados à *Power Platform, Dataverse* e *Dynamics 365*. O terceiro capítulo descreve a metodologia utilizada durante o desenvolvimento da *canvas app "Event Manager"*. No quarto capítulo é feita a análise dos requisitos relativa á *app "Event Manager"*. O quinto capítulo é dedicado à implementação da aplicação, descrevendo as etapas e os recursos utilizados. Por fim, o sexto e último capítulo apresenta as conclusões alcançadas ao longo do trabalho.

2. Formação em Power Platform, Dataverse e Dynamics 365

Durante o estágio, a formação adquirida centrou-se em torno de três tecnologias: *Power Platform*, *Dataverse* e *Dynamics 365*. Estas tecnologias estão interligadas e oferecem recursos poderosos para a criação de aplicações, gestão de dados e gestão de processos de negócio.

A *Power Platform* é uma plataforma de aplicações que permite criar soluções eficientes e flexíveis com ferramentas de *low code*. Essas ferramentas possibilitam a criação de aplicações com um mínimo de codificação manual, recorrendo a interfaces gráficas e componentes reutilizáveis, como botões, formulários, tabelas e outros elementos de interface. O *Dataverse* é uma plataforma de dados que permite o armazenamento e gestão dos mesmos, num ambiente integrado com as tecnologias da Power Platform e *Dynamics 365*. O *Dynamics 365* é um conjunto de aplicações de negócios que inclui ferramentas para vendas, atendimento ao cliente, operações e muito mais. Estas três tecnologias estão intimamente relacionadas e projetadas para trabalhar em conjunto.

2.1. Power Platform

A *Power Platform* é um conjunto de ferramentas que permite criar aplicações personalizadas sem a necessidade de conhecimentos avançados de programação. Inclui o *Power Apps* para criação de aplicações *low code,* o *Power Automate* para automação de fluxos de trabalho, *Power BI* para análise de dados e o *Power Virtual Agents* para criação de *chatbots*.

A plataforma em foco neste relatório será a *Power Apps* sobre a qual foi criada uma aplicação *canvas*, *Event Manager* (gestor de eventos) e que serviu de base para aplicar os conhecimentos principais adquiridos durante o estágio.

O *Power Apps* é um conjunto de aplicações, serviços e conectores, bem como uma plataforma de dados que oferece um ambiente de desenvolvimento rápido para criação de produtos personalizados para os seus negócios [3].

Dentro do ecossistema do *Power Apps* existem dois tipos de aplicações que podem ser criadas, as *Canvas Apps* e as *Model Driven Apps.* As *Canvas Apps* são mais orientadas para aplicações visuais, que exijam *interfaces* mais personalizados. As *Model Driven Apps*, possuem uma estrutura mais orientada a dados e são ideais para aplicações com casos de uso complexos.

Qualquer aplicação criada pelas ferramentas da *Power Platform* pode fazer uso do *Dataverse*. O *Dataverse* oferece armazenamento centralizado e seguro de dados, possui integração nativa com o *Power Apps*, recursos avançados de segurança e integração com outras ferramentas da *Power Platform* como *o Power Automate e o Power BI*. Estas funcionalidades permitem criar aplicativos seguros, com acesso fácil aos dados e gestão eficiente de recursos.

2.2 Dataverse

O *Dataverse* é um conjunto de APIs (*Application Programming Interfaces*) que nos permitem armazenar dados em tabelas e gerir as nossas aplicações no ecossistema da *Power Platform*. Ele oferece armazenamento centralizado e seguro de dados, facilitando o seu acesso e gestão num único local.

O *Dataverse* inclui um conjunto de tabelas padrão que abrange vários cenários típicos, como contas (tabela *Account*), contactos (tabela *Contact*), utilizadores (tabela *User*) e muitas mais. Permite ainda criar tabelas personalizadas específicas da organização [4].

Uma das vantagens do *Dataverse* é a possibilidade de diferentes aplicações poderem partilhar o mesmo ambiente e, portanto, o mesmo *Dataverse*. Isso significa que vários produtos de *software* podem utilizar os mesmos conjuntos de dados, evitando assim a sua duplicação e manter a consistência dos dados entre as aplicações. Por exemplo, as aplicações A e B podem partilhar o mesmo *Dataverse* num ambiente, enquanto acedem e atualizam os dados comuns de forma coordenada.

Além disso, o *Dataverse* oferece recursos avançados de segurança. É possível definir permissões granulares ao nível da tabela, campo e registo para controlar o acesso aos dados. Isso significa que é possível especificar quem pode visualizar, editar ou excluir dados em diferentes partes do *Dataverse*, garantindo uma segurança adequada.

2.3 Ambientes (Environments)

Citando o artigo da Microsoft [5]:

“Um ambiente é um espaço para armazenar, gerir e partilhar os dados de negócio, as aplicações e os fluxos da sua organização. Também serve como um recipiente para aplicações separadas que podem ter diferentes funções, requisitos de segurança ou públicos-alvo. O que escolhe para fazer uso dos ambientes depende da sua organização e das aplicações que está a tentar criar. Por exemplo:

- Pode optar por criar apenas as suas aplicações num único ambiente. Poderá criar ambientes separados que agrupem as versões de teste e de produção das suas aplicações.
- Poderá criar ambientes separados que correspondam a equipas ou departamentos específicos na sua empresa, cada um contendo os dados e aplicações relevantes para cada audiência.
- Também poderá criar ambientes separados para diferentes ramos globais da sua empresa.”

Dito de outra forma, ambientes na *Power Platform* são espaços de trabalho isolados onde se podem desenvolver, gerir aplicações e recursos. Cada ambiente possui as suas próprias configurações, dados e permissões de acesso. Eles oferecem um ambiente separado para testar,

implementar e manter as soluções criadas sem interferir noutros ambientes. Com os ambientes, podemos organizar e controlar o ciclo de vida das aplicações, garantindo a segurança e a separação entre diferentes projetos e equipas.

2.3.1 Tipos de ambientes

Existem vários tipos de ambientes que podem ser criados no *Power Apps*, cada ambiente serve um objetivo diferente e tem as suas próprias características, descrevem-se os 5 principais [6]:

1. ***Default environment*:** Este ambiente é criado automaticamente para cada *tenant* e é partilhado por todos os utilizadores da organização. Um *tenant* representa uma instância da Azure AD (*Cloud* da *Microsoft*), que permite fazer a gestão das identidades e permissões de acesso dos utilizadores da organização.
   Este ambiente não é vocacionado para desenvolver aplicativos críticos e não deve ser usado para hospedar aplicativos de produção. Este ambiente não pode ser desativado ou eliminado.

2. ***Developer environment*:** É um ambiente dedicado para testar soluções e funcionalidades antes de implementá-las num ambiente de produção.

3. ***Production environment*:** O ambiente principal onde as aplicações e recursos são implementados e utilizados pelos utilizadores finais.

4. ***Sandbox environment*:** Um ambiente isolado para realizar testes avançados, experimentar configurações e explorar novas funcionalidades sem afetar os ambientes de produção ou de teste.

5. ***Trial environment*:** Um ambiente usado para mostrar as capacidades e funcionalidades das aplicações aos *stakeholders* e potenciais utilizadores.

2.3.2 Acesso de uma aplicação num ambiente

Como já referido nas secções anteriores, dentro de uma organização, é possível haver vários ambientes, cada um com seu próprio conjunto de tabelas e dados. Cada ambiente é uma instância separada do *Dataverse*, possui os seus próprios dados e personalizações que não são partilhados com outros ambientes. Isso permite criar ambientes separados para diferentes propósitos, como desenvolvimento, teste, produção, gestão de dados e personalizações de forma independente em cada ambiente.

Quando se publica uma aplicação num ambiente, somente os utilizadores que têm acesso a esse ambiente e estão devidamente "autorizados", poderão aceder e utilizar essa aplicação. É o administrador do ambiente, que define quais as permissões que cada utilizador tem sobre o ambiente e seus recursos. Para que outros utilizadores na organização possam aceder e utilizar a aplicação, esta terá de ser partilhada com eles, especificando no Azure Active Directory (Azure AD) quais utilizadores ou grupos de segurança podem aceder à aplicação.

2.4 Soluções (Solutions)

As soluções são utilizadas para transportar aplicações e componentes de um ambiente para outro ou para aplicar um conjunto de personalizações a aplicações existentes [7].Por exemplo, imagine-se que foi desenvolvida uma solução chamada "Solução de Vendas" que inclui uma aplicação de gestão de vendas, uma tabela de clientes e um fluxo de trabalho de aprovação de pedidos. Com essa solução, é possível transportar e implementar a aplicação de gestão de vendas, juntamente com todos os outros componentes, em diferentes ambientes, como ambiente de desenvolvimento, ambiente de teste e ambiente de produção.

Diferentes soluções dentro do mesmo ambiente partilham a mesma instância do *Dataverse*. Isso significa que se houver várias soluções dentro do mesmo ambiente, todas elas poderão aceder e usar as mesmas tabelas e dados dentro do *Dataverse*. Por exemplo, suponhamos que existe uma solução de "Gestão de Projetos" e uma solução de "Gestão de Recursos Humanos" no mesmo ambiente, ambas as soluções podem interagir com os mesmos dados de clientes e funcionários no *Dataverse*, permitindo uma integração e partilha de informação entre as diferentes soluções.

Em suma, as soluções são ferramentas que permitem o transporte e a personalização de aplicações e componentes, facilitando a implementação e a partilha de funcionalidades em diferentes ambientes. Elas ajudam a simplificar o processo de desenvolvimento e personalização de aplicações, permitindo a reutilização de componentes e a manutenção de todo o ambiente de trabalho. Abaixo apresenta-se a Figura 1 que representa a Solução *Event Manager* dentro do *Ambiente Developer Environment*. A aplicação criada foi desenvolvida dentro da solução mencionada.

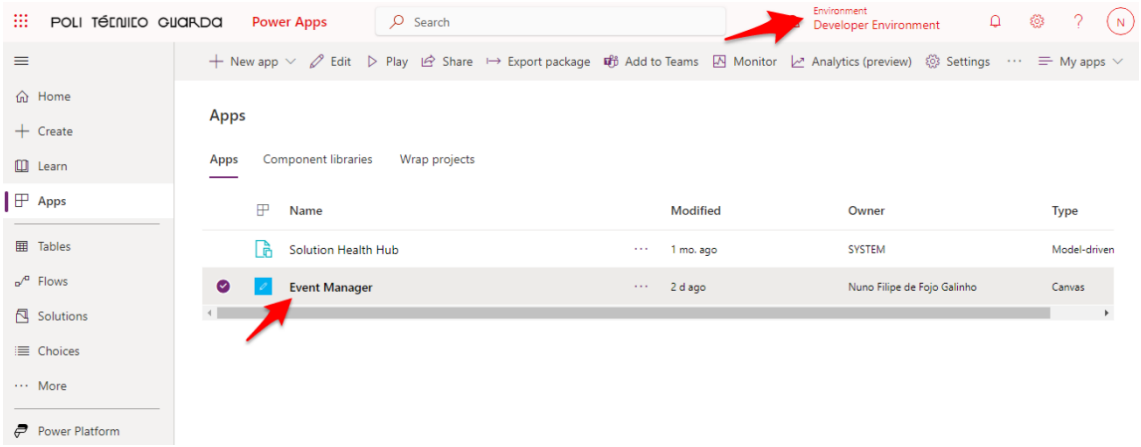


Figura 1 - Solução Event Manager

2.5 Segurança

A segurança de aplicações e dados em *Power Apps* é gerida a partir de unidades de negócios (*Business Units*), equipas (*Teams*) e utilizadores (*Users*), bem como através de grupos de segurança (*security groups*) e *security roles*. Os *security groups* são usados para controlar o acesso a ambientes, enquanto as *security roles* são usadas para controlar o acesso a dados dentro desses ambientes.

Unidades de negócios e equipas são conceitos relacionados à organização e gestão de utilizadores. Unidades de negócios representam divisões ou departamentos dentro de uma organização, enquanto equipas são grupos de utilizadores que trabalham juntos com um objetivo comum. Os utilizadores são os indivíduos que têm acesso à *Power Apps* para criar ou usar aplicações.

Security groups e *security roles* são conceitos relacionados ao controlo de acesso a ambientes e dados respetivamente. *Security groups* são usados para controlar o acesso a ambientes, permitindo que se especifique quais utilizadores ou grupos de utilizadores têm permissão para aceder a um ambiente. *Security roles*, por outro lado, são usadas para controlar o acesso a dados dentro desses ambientes. É possível atribuir *security roles* a utilizadores ou equipas para especificar quais dados eles podem aceder e quais ações eles podem executar.

Em suma a segurança no *Power Apps* é gerida através de unidades de negócios, equipas e utilizadores, bem como através de *security groups* e *security roles*. *Security groups* são usados para controlar o acesso a ambientes, enquanto as *security roles* são usadas para controlar o acesso a dados dentro desses ambientes.

2.5.1 Unidades de Negócio (Business Units)

As unidades de negócio são usadas para modelar a estrutura hierárquica de uma organização e controlar o acesso aos dados e às permissões do utilizador dentro dessa organização. As *Business units* podem ser usadas para representar divisões, departamentos ou equipas e podem ser encadeadas para refletir a estrutura hierárquica da organização [8].

Nesta estrutura, uma unidade de negócio pode ter várias equipas, enquanto uma equipa está associada a apenas uma unidade de negócio. Um utilizador pode pertencer a várias equipas e várias unidades de negócio. Por exemplo, imagine-se uma empresa com duas unidades de negócios: Vendas e *Marketing*. A unidade de negócios de Vendas tem duas equipes: Vendas Internas e Vendas Externas. A unidade de negócios de *Marketing* tem uma equipa: *Marketing* Digital. Um utilizador pode pertencer à equipa de Vendas Internas e à equipa de *Marketing* Digital, o que significa que ele tem acesso aos dados das unidades de negócios de Vendas e *Marketing*.

Cada unidade de negócios tem uma equipa padrão (*default team*), mas é impossível adicionar ou remover utilizadores da equipa padrão da unidade de negócios. Isso significa que os utilizadores são automaticamente adicionados à equipa padrão da unidade de negócios à qual estão atribuídos e não podem ser de lá excluídos.

Em resumo, as unidades de negócios e as equipas permitem que as organizações criem uma estrutura hierárquica para controlar o acesso aos dados e garantir que os utilizadores tenham acesso apenas às informações necessárias para desempenhar suas funções. Isso ajuda a manter a segurança dos dados e a eficiência operacional.

2.5.2 Security Roles

Security Roles definem a forma como diferentes utilizadores acedem a diferentes tipos de registo. Para controlar o acesso a dados e recursos, podem-se criar ou modificar *Security Roles* e alterar os direitos de acesso atribuídos aos seus utilizadores.

Um utilizador pode ter vários *security roles*. Os privilégios dos *security roles* são cumulativos. São concedidos aos utilizadores os privilégios disponíveis em cada função que lhes é atribuída [9].

Na *Power Platform*, as funções (*roles*) e permissões são definidos quer ao nível da aplicação quer ao nível dos dados. Ao nível da aplicação, são usados *security groups* já ao nível dos dados são usadas *security roles.* Através dos *security groups* é possível definir que utilizadores têm acesso a determinado ambiente (*Environment*) do *Power Apps*. Se um utilizador for adicionado a um ambiente, mas não pertencer a esse ambiente esse utilizador não terá acesso a ele e por consequência aos seus recursos.

No entanto em termos de segurança, a base de dados subjacente desempenha um papel crucial. Se a aplicação estiver ligada ao *Dataverse*, é possível refinar ainda mais o acesso ao nível dos dados e é aí que entram as *security roles*.

Através de *security roles*, é possível no *Dataverse*, definir permissões ao nível de tabela, coluna e linha, isso permite especificar quais utilizadores ou grupos como *Teams* ou *Business Units* têm acesso de leitura, escrita ou exclusão a tabelas ou colunas específicas dentro da base de dados. Também é possível incorporar segurança ao nível da linha (registo) para restringir o acesso a registos específicos com base em certas condições ou critérios.

Ao combinar permissões ao nível da aplicação, com *security groups* e permissões ao nível dos dados, com *security roles*, é possível garantir que os utilizadores certos têm acesso às funcionalidades apropriadas dentro da aplicação. Essa abordagem fornece uma estrutura de segurança abrangente, onde tanto a aplicação quanto os dados inerentes são protegidos.

2.5.2.1 Atribuição de Security Roles

A atribuição de *security roles* desempenha um papel crucial na gestão de acessos e permissões dentro do ambiente *Power Platform*. Uma forma eficaz de estruturar e controlar o acesso aos dados é utilizando unidades de negócio (*Business Units*) e equipas (*Teams*).

As unidades de negócio permitem agrupar utilizadores e dados de forma hierárquica, criando uma estrutura organizada. Por exemplo, podemos ter uma unidade de negócio chamada "Vendas" e dentro dela, equipas como "Equipa de Vendas 1" e "Equipa de Vendas 2". Cada equipa tem um conjunto específico de utilizadores. Atribuir *security roles* a essas equipas permite controlar o acesso aos dados por cada uma delas. Por exemplo, podemos ter uma *security role* chamada "Vendedor" que concede permissões para ler e escrever em determinadas tabelas de vendas. Essa *security role* "Vendedor" é atribuída à equipa "Equipa de Vendas 1". Desta forma, apenas os utilizadores pertencentes a esta equipa terão acesso às tabelas de vendas e poderão realizar as ações permitidas pela *security role*.

É importante destacar que as *security roles* são cumulativas. Isso significa que um utilizador pode ter várias *security roles* atribuídas, resultando nas permissões combinadas de todas as *security roles*. Por exemplo, se um utilizador A pertence à equipa "Equipa de Vendas 1" e também à equipa "Equipa de *Marketing*", e as duas equipas têm *security roles* atribuídas, então o utilizador A terá as permissões resultantes das duas *security roles*.

A atribuição de *security roles* baseada em unidades de negócio e equipas oferece um controle granular sobre o acesso aos dados. É uma abordagem eficaz para garantir que apenas os utilizadores autorizados tenham acesso aos dados relevantes para as suas responsabilidades e funções dentro da organização.

2.5.2.2 Privilégios, Níveis de Acesso e Herança das Security Roles

No contexto das *Power* Apps, as *security roles* e seus privilégios são herdados de cima para baixo na hierarquia da unidade de negócio. Isso significa que o acesso de um utilizador aos dados é determinado pelas *security roles* atribuídas a eles no nível da sua unidade de negócio e em todas as unidades de negócio acima na hierarquia.

Como já mencionado em secções anteriores, um utilizador também pode herdar *security roles* das equipas às quais pertence. Se um utilizador é membro de uma equipa que possui uma *security role* atribuída, esse utilizador herdará os privilégios associados a essa *security role*. Relembramos também que os privilégios são cumulativos, o que significa que se um utilizador tem várias *security roles* atribuídas, ao nível do utilizador (*User*) mais as da equipa e unidades de negócio, ele terá todos os privilégios associados a cada uma dessas *security roles*.

Os privilégios determinam o que um utilizador pode fazer com os dados (criar, ler, atualizar…), enquanto os níveis de acesso determinam a que nível o utilizador pode realizar essas operações. Os níveis de acesso incluem utilizador (*User*), Equipa (*Team*), Unidade de Negócio (*Business Unit*), Unidade de Negócio Pai-Filho (*Parent Child Business Unit*) e Organização (*Organization*). Na Figura 2 podem-se visualizar os privilégios e níveis de acesso de uma *security role* chamada de "*My Account Manager*".

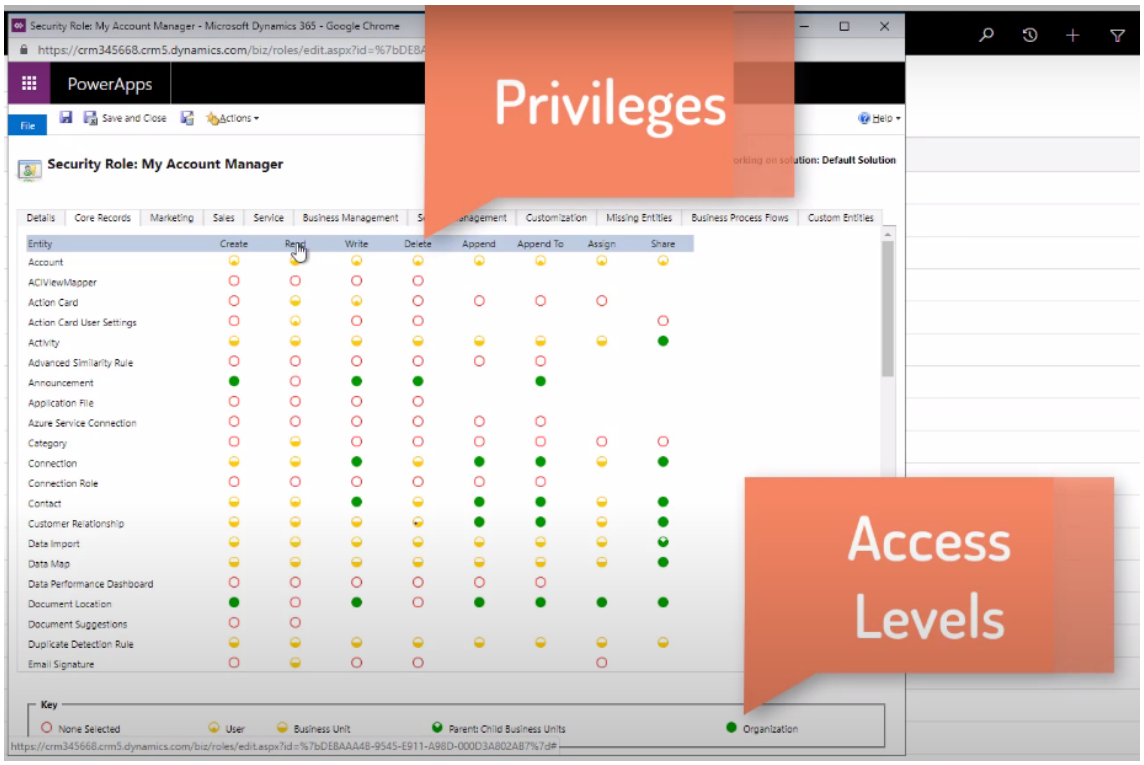


Figura 2 - privilégios e níveis de acesso da security role "My Account Manager"

Apresenta-se um exemplo: se um utilizador pertencer à unidade de negócio "Vendas" com as equipas A e B e tem uma *security role* "*My Account Manager*" que concede permissão para escrever ao nível da equipa na tabela de *Account*, isso significa que o utilizador pode escrever nos projetos da Equipa A que utilizam essa tabela. No entanto, ele não terá acesso de escrita aos projetos da Equipa B, que também utilizam a mesma tabela. Por outro lado, se o utilizador tiver permissão para escrever ao nível da unidade de negócio na tabela *Account*, isso permitirá que ele escreva nos projetos quer da Equipa A quer da Equipa B, já que a permissão é concedida a um nível mais alto que engloba ambas as equipas dentro da Unidade de Negócio (Ver Figura 3).

Desta forma, os privilégios, níveis de acesso e herança das *security roles* proporcionam uma estrutura flexível para controlar o acesso aos dados com base na hierarquia das unidades de negócio e nas atribuições de equipas. Isso permite que as organizações personalizem as permissões de acesso de acordo com suas necessidades e garantam que os utilizadores tenham acesso apropriado aos dados relevantes para suas funções.

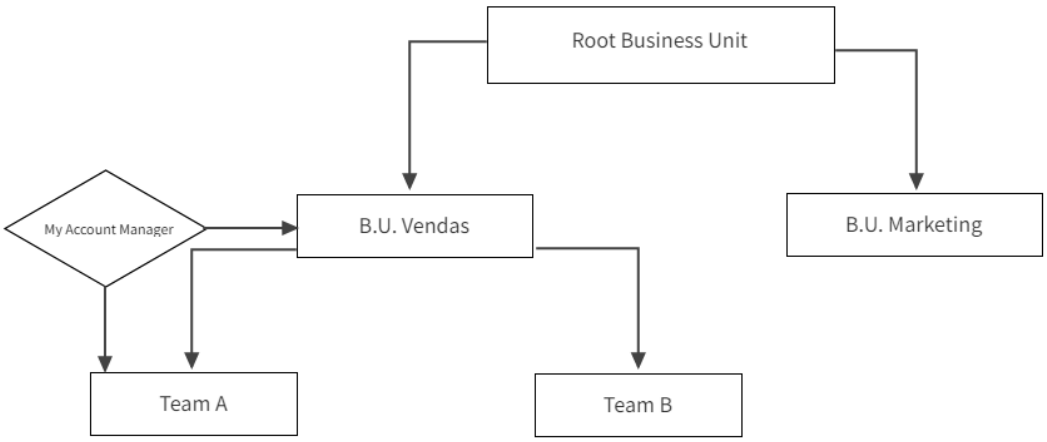


Figura 3 - Exemplo de Diagrama Hierárquico de Business Units numa organização

2.5.3 Security Groups

Os grupos de segurança (*security groups*) são usados para gerir o acesso dos utilizadores aos ambientes no *Power Platform*. Quando se associa um grupo de segurança a um ambiente, apenas os membros desse grupo terão acesso ao ambiente e por consequência aos seus recursos. Se uma empresa possuir vários ambientes, podem-se usar grupos de segurança para controlar quais utilizadores podem ser membros de um ambiente específico. Pode-se criar um *security group* na *Azure Active Directory* e atribuir membros a ele. Em seguida, pode-se associar o *security group* a um ambiente [10].

No entanto, os grupos de segurança não controlam diretamente o acesso dos utilizadores a aplicações específicas ou recursos específicos (como dados) dentro de um ambiente. O controlo de acesso a dados é feito com *security roles* e foi abordado nas secções imediatamente acima. O controlo de acesso a aplicações específicas dentro de um ambiente está fora da fronteira deste trabalho.

Na criação da *canvas app Event Manager* não foi possível atribuir *security groups* aos utilizadores do ambiente por restrições da nossa conta da *Microsoft* ao *Azure Active Directory*.

2.5.4 Auditoria (Auditing)

A auditoria permite controlar as alterações que ocorrem nos dados e o acesso dos utilizadores a um ambiente específico. Permite detetar e registar as atividades realizadas dentro do sistema, como as operações de criação, leitura, atualização e exclusão de registos, bem como alterações feitas em campos específicos. Ela fornece um histórico completo das ações realizadas pelos utilizadores, permitindo uma maior transparência e visibilidade das atividades no sistema.

Com a funcionalidade de auditoria, é possível monitorizar as alterações feitas nos dados, identificar qualquer atividade suspeita ou não autorizada, identificar problemas de desempenho, detetar erros e analisar tendências de utilização do sistema.

2.6 Dynamics 365

O *Dynamics 365* é um portfólio de aplicativos de negócios inteligentes que oferece eficiência operacional superior e experiências de clientes inovadoras permitindo que as empresas se tornem mais ágeis e reduzam a complexidade sem aumentar os custos [11].

A presente secção abordará, essencialmente, o módulo de Vendas (*Sales*) dentro do Dynamics 365. Apresentará uma breve definição de alguns módulos que o constituem, fará uma definição sucinta do que é um CRM e terminará apresentando o módulo de *Sales* e ciclo de vida de uma venda.

2.6.1 Módulos de Dynamics 365

Alguns dos módulos disponíveis no *Dynamics 365* incluem:

- ***Dynamics 365 Sales***: Ajuda as organizações a gerir os seus processos de vendas, desde a geração de *leads* até ao fecho de negócios.
- ***Dynamics 365 Marketing***: Fornece ferramentas para criar e gerir campanhas de marketing, acompanhar *leads* e analisar o desempenho do *marketing*.
- ***Dynamics 365 Customer Service***: Ajuda as organizações a gerir as interações com os clientes e a fornecer suporte através de múltiplos canais.
- ***Dynamics 365 Field Service***: Fornece ferramentas para gerir operações de serviço externo, incluindo agendamento, despacho e acompanhamento de técnicos de campo.
- ***Dynamics 365 Finance***: Ajuda as organizações a gerir as suas operações financeiras, incluindo contabilidade, orçamentação e relatórios financeiros.
- ***Dynamics 365 Supply Chain Management***: Fornece ferramentas para gerir as operações da cadeia de abastecimento, incluindo compras, gestão de inventário e logística.

Estes são apenas alguns dos módulos disponíveis no *Dynamics 365*. A *suite* foi projetada para ser flexível e escalável, permitindo que as organizações escolham os módulos que melhor atendam às suas necessidades e adicionem módulos adicionais à medida que o seu negócio cresce e evolui [11].

2.6.2 O que é um CRM no contexto de Dynamics 365?

CRM é a sigla para *Customer Relationship Management*, que pode ser traduzido como Gestão do Relacionamento com o Cliente. É um *software* desenvolvido para tornar o relacionamento entre uma empresa e a sua base de clientes mais eficiente. O CRM é desenhado com o intuito de melhorar o relacionamento entre clientes e empresa, resultando numa melhor qualidade de serviço quer para clientes existentes quer outros potenciais a ser cativados.

O CRM permite melhorar o tempo de resposta às solicitações dos clientes, acompanhar novas oportunidades de negócio, criar campanhas de *marketing* mais eficazes, analisar padrões de compra e venda, garantir a qualidade do serviço e automatizar tarefas repetitivas. Essa ferramenta ajuda as empresas a aumentar a satisfação do cliente e impulsionar o crescimento dos negócios.

O *Dynamics 365* é uma plataforma abrangente que engloba o CRM e oferece uma série de aplicações de negócios inteligentes. O *Dynamics 365 Sales*, *Marketing* e *Customer Service* são alguns dos módulos disponíveis que permitem às empresas fazer a gestão de todas as interações com os clientes de forma integrada e eficiente.

Com o *Dynamics 365*, as empresas podem centralizar as informações do cliente, coletar informações valiosas e tomar decisões mais informadas. Essa plataforma ajuda a impulsionar o crescimento dos negócios, melhorar a eficiência operacional e cativar o cliente [12].

2.6.3 Dynamics 365 Sales

Citando a *Microsoft* [13]:

"O *Dynamics 365 Sales* permite aos representantes de vendas estabelecer relações fortes com os seus clientes, efetuar ações com base em conhecimentos aprofundados e fechar negócios mais rapidamente. Utilize o *Dynamics 365 Sales* para monitorizar as suas contas e contactos, alimentar as suas vendas da oportunidade potencial a encomenda e criar colaterais de vendas. Também pode criar listas de marketing e campanhas, e seguir incidentes de serviço associados a contas ou oportunidades específicas."

O *Dynamics 365 Sales* é uma solução completa para gerir o ciclo de vendas, desde a identificação de potenciais clientes até o fecho das negociações. Com o *Dynamics 365 Sales*, as empresas podem otimizar o processo de vendas, aumentar a eficiência das equipas comerciais e cativar o cliente em todas as etapas do ciclo de vendas.

Em seguida, passaremos á descrição do ciclo de vida de uma venda, no contexto da plataforma *Dynamics 365 Sales*.

Ciclo de vida de uma venda

Eis a forma como o *Dynamics 365 Sales* gere o ciclo de vendas nas suas diferentes etapas:

1. Criação de Leads: *Leads* representam potenciais clientes. *Leads* podem ser geradas através de várias fontes, como campanhas de *marketing*, consultas em *sites* ou participação em eventos. Por exemplo, uma *lead* pode ser criada no sistema quando um potencial cliente preenche um formulário no *site* da empresa para solicitar mais informações sobre seus produtos ou serviços. Essas *leads* são registadas como atividades no *Dynamics 365 Sales*, permitindo um acompanhamento adequado.

2. Qualificação de Leads: Após a sua criação, as *leads* são qualificados e avaliadas pela equipa de vendas. Eles avaliam a adequação da *lead*, seu nível de interesse e a probabilidade desta se converter em Oportunidade (*Opportunity*). Se a *lead* mostrar potencial para uma venda e for qualificada, são criados um Contacto (*Contact*) e uma Conta (*Account*) com os dados recolhidos até á data e a *lead* é transformada numa Oportunidade. A Figura 4, Figura 5 e Figura 6 representam a qualificação da *lead,* "SR Lead", a oportunidade criada "Interessado em adquirir produto x" e o contacto criado "SR Lead".

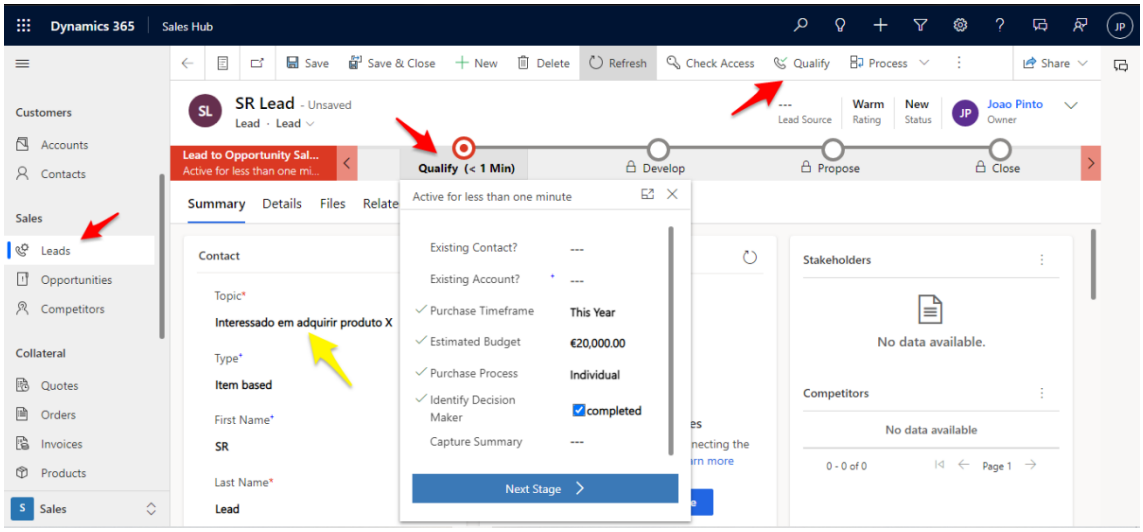


Figura 4 - Qualificação da lead, "SR Lead"

Descrição da Figura 4:

Quando se clica no botão *Qualify* o sistema faz:

1. Este contacto existe? Se não existe, cria Contacto (tabela *Contact*).
2. Esta conta existe? Se não existe, cria Conta (tabela *Account*).
3. Cria a relação entre o contacto criado e a conta.
4. Transforma esta *Lead* numa *Opportunity* e relaciona-a ao contacto e á conta.

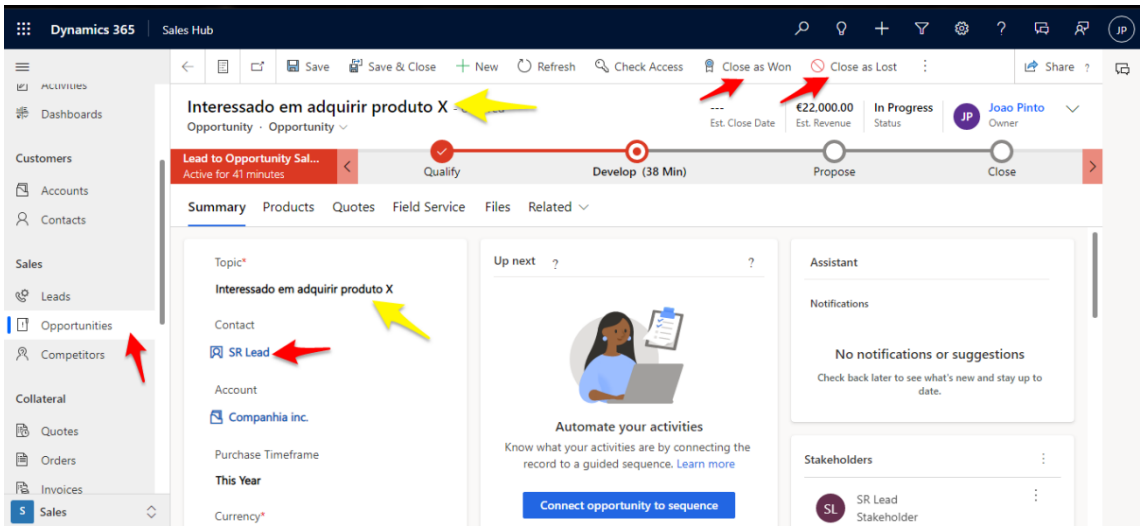


Figura 5 - Oportunidade criada, "Interessado em adquirir produto x"

A Figura 5 representa a oportunidade criada "Interessado em adquirir produto x", baseada na *lead* qualificada.

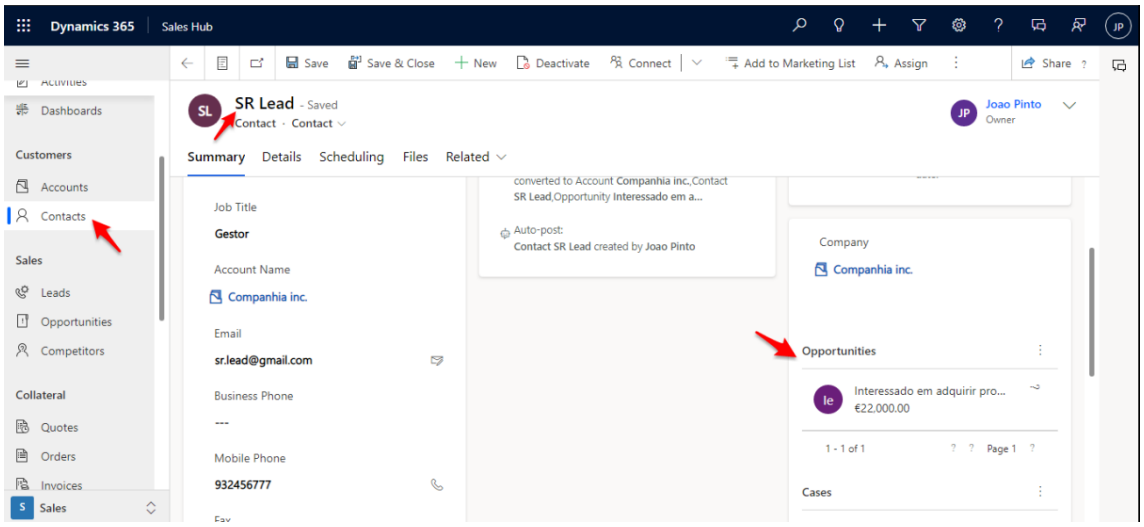


Figura 6 - Contacto criado, "SR LEAD"

A Figura 6 representa o contacto criado "SR LEAD", baseado na *lead* qualificada.

3. Gestão de Oportunidades: Uma oportunidade (*Opportunity*) representa um possível negócio. As oportunidades quando criadas são registadas como Atividades (*Activities***)**. Atividades podem ser usadas para planear, acompanhar e organizar todas as comunicações com os clientes. Por exemplo, podem-se tomar notas, enviar correio eletrónico, fazer chamadas telefónicas, marcar compromissos e atribuir tarefas, permitindo assim que a equipa de vendas acompanhe o progresso e as interações relacionadas a cada oportunidade (ver Figura 7).

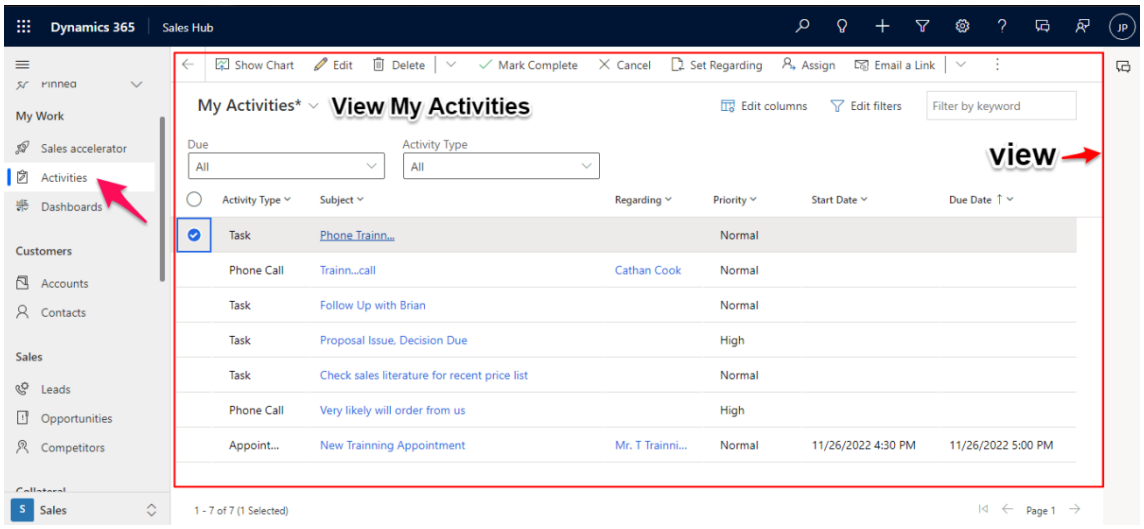


Figura 7 - View "My Activities"

A Figura 7 representa a *view My Activities* da tabela *Activities* e apresenta a lista de atividades e tarefas executadas ou a executar pelo utilizador, se pretendêssemos ver as atividades de toda a equipa, escolheríamos a *view My Team Members' Activities*. Em *Activities* podemos ficar a par das nossas tarefas, das tarefas das equipas, criar novas atividades (*e-mail*, telefonemas, reuniões…) criar e atribuir tarefas e associar atividades a contas. A visibilidade das atividades é determinada pelas *security roles* e permissões atribuídas aos utilizadores. Apenas os utilizadores com as *security roles* e permissões adequadas podem ver, editar ou eliminar atividades.

4. Gestão do Pipeline de Vendas: O *Pipeline* de Vendas fornece uma representação visual das diferentes fases de um processo de vendas, desde a criação de *leads* até ao fecho do negócio. A gestão do pipeline de vendas permite acompanhar e gerir oportunidades, analisar o desempenho das vendas e tomar decisões informadas para impulsionar o crescimento das receitas. Eis a descrição das etapas:

a) **Qualificar (*Qualify*):** Nesta fase, a equipa de vendas avalia a oportunidade e determina o seu potencial de sucesso. Avalia fatores como as necessidades, o orçamento e o processo de tomada de decisão do cliente para determinar se vale a pena prosseguir com a oportunidade. Por exemplo, um representante de vendas pode qualificar uma oportunidade confirmando que o cliente tem um interesse genuíno no produto e a capacidade financeira para efetuar a compra.

b) **Desenvolver (*Develop*):** Uma vez qualificada a oportunidade, esta passa para a fase de desenvolvimento. Aqui, a equipa de vendas concentra-se em compreender as necessidades do cliente e em adaptar a sua oferta para satisfazer essas necessidades. A equipa de vendas participa em discussões, apresentações e demonstrações para mostrar o valor e as vantagens da sua solução. Por exemplo, um representante de vendas pode desenvolver a oportunidade efetuando demonstrações de produtos e apresentando propostas pormenorizadas que descrevam a forma como a sua solução responde aos desafios específicos do cliente.

c) **Proposta (*Propose*):** Quando a oportunidade chega à fase de proposta, a equipa de vendas apresenta uma proposta formal ao cliente. Esta proposta inclui pormenores sobre preços, termos e condições e qualquer informação adicional relevante para o negócio. O representante de vendas trabalha em estreita colaboração com o cliente para responder a quaisquer preocupações ou questões que este possa ter. Por exemplo, um representante de vendas pode propor um contrato que descreva as opções de preços, os prazos de entrega e os acordos de nível de serviço para análise e consideração do cliente.

d) **Fecho (*Close*):** A fase de fecho é atingida quando o cliente se compromete a avançar com a compra. Envolve a finalização do negócio, a negociação de quaisquer termos restantes e a obtenção das aprovações necessárias. Uma vez cumpridos todos os requisitos, o representante de vendas e o cliente preenchem a documentação necessária, como contratos ou ordens de compra, para fechar oficialmente a oportunidade. Por exemplo, um representante de vendas pode fechar a oportunidade ao receber um contrato assinado e processar o pagamento, indicando uma venda bem-sucedida.

Apresenta-se a seguir a Figura 8 que representa um *Pipeline* de Vendas.

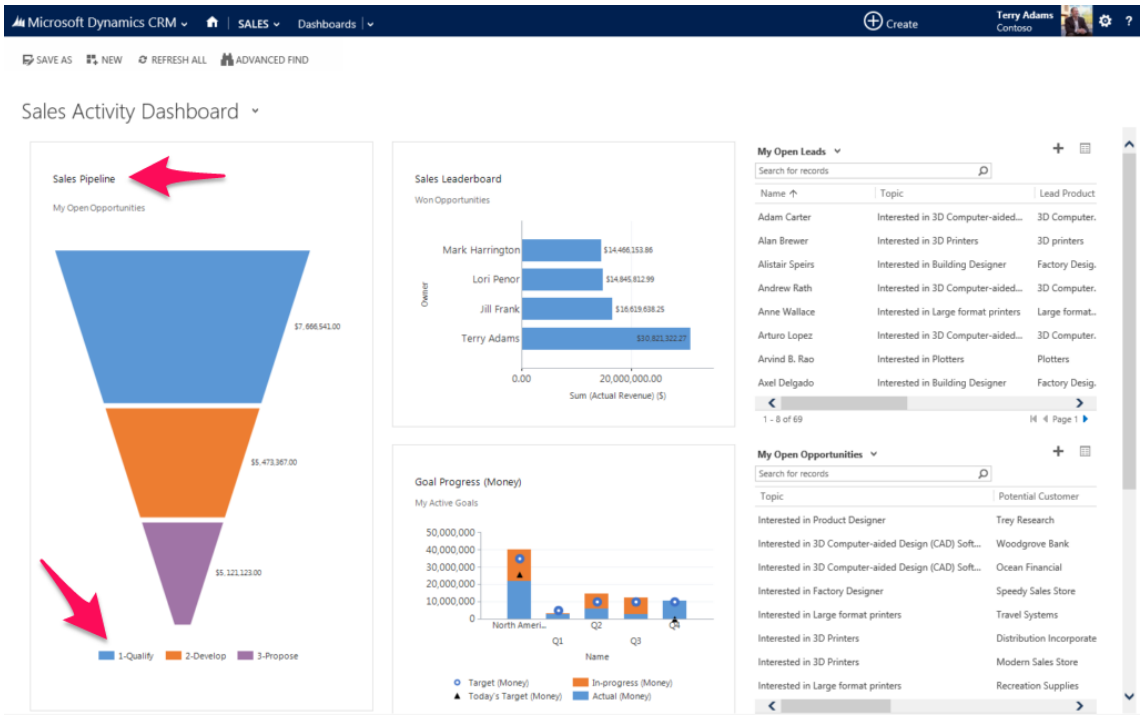


Figura 8 - Pipeline de Vendas

Fonte: https://www.selecthub.com/wp-content/uploads/2020/10/Dynamic_365_Sales_-_Sales_Pipeline-1024x640.png

Na Figura 8 podem-se observar, a diferentes cores, as diferentes etapas do *Pipeline* de vendas e a informação detalhada relativa a cada uma delas.

5. Gestão de Orçamentos e Propostas: Os representantes de vendas podem criar e gerir orçamentos e propostas, diretamente na plataforma. Podem personalizar os preços, adicionar detalhes ao produto e enviá-los ao cliente para revisão. Por exemplo, um representante de vendas pode criar uma proposta detalhada que descreva o âmbito do trabalho, as opções de preços e os termos e condições e enviar ao cliente.

6. Encerramento de vendas: O Dynamics 365 Sales apoia o fecho de negócios fornecendo ferramentas para gestão de contratos e assinaturas eletrónicas. Os representantes de vendas podem gerar contratos ou acordos, acompanhar o seu estado e facilitar o processo de assinatura digital. Assim que o cliente assina o contrato, a oportunidade é marcada como "*Close as Won*" no sistema. Por exemplo, um representante de vendas pode utilizar a plataforma para enviar um contrato digital ao cliente e receber uma assinatura eletrónica, indicando o encerramento bem-sucedido da venda.

7. Gestão de Contas e Contactos: Após o fecho da venda, a plataforma permite aos representantes de vendas gerir eficazmente as contas dos clientes e promover relações contínuas com os mesmos. Por exemplo, o sistema pode fornecer informações sobre os padrões de compra dos clientes, permitindo que um representante de vendas identifique potenciais oportunidades com base nas compras anteriores do cliente.

Em suma, o ciclo de vida de vendas do *Dynamics 365* é uma sequência de etapas que guia o processo de vendas. Começando pela identificação de *leads* promissoras, passando pela qualificação e gestão de oportunidades, até chegar ao fecho da venda. Cada etapa envolve interações continuas e especificas com os clientes, visando avançar para o próximo estágio e alcançar uma venda bem-sucedida. Com o *Dynamics 365*, as equipas de vendas podem gerir eficazmente o ciclo de vendas, identificar problemas, concentrar esforços e aumentar a probabilidade de fechar negócios. É uma abordagem estruturada e orientada a resultados que permite às empresas maximizar o seu potencial de vendas e melhorar o relacionamento com os clientes.

2.7 Plugins

Os *plugins* são componentes especiais na *Power Platform*, sendo amplamente utilizados no *Dynamics 365* para adicionar funcionalidades personalizadas e processamento de negócios específicos. Eles são executados em resposta a eventos ou ações específicas, permitindo estender as capacidades do sistema. Por exemplo, no *Dynamics 365 Sales*, um *plugin* pode ser usado para automatizar o cálculo de comissões de vendas com base numa regra de negócio simples, como atribuir uma percentagem fixa de comissão por cada venda realizada. Isso elimina a necessidade de cálculos manuais e garante que os vendedores recebam a compensação adequada de acordo com a política da empresa.

Os *plugins* também podem ser implementados nas *Power Apps*, ampliando as possibilidades de personalização e comportamento das aplicações desenvolvidas. Por exemplo, no *Power Apps*, um *plugin* pode ser utilizado para validar dados inseridos num formulário em tempo real, ou até mesmo executar cálculos complexos. Essa capacidade de adicionar lógica personalizada e comportamentos específicos torna o *Power Apps* mais flexível e adaptável às necessidades do negócio.

Em resumo, os *plugins* desempenham um papel valioso na *Power Platform*, tanto no *Dynamics 365* quanto no *Power Apps*. Eles permitem estender as capacidades das aplicações, automatizar processos de negócio e adicionar funcionalidades personalizadas. Com o uso de *plugins*, é possível criar soluções mais poderosas e adaptadas às necessidades específicas de cada organização, impulsionando a eficiência e produtividade das equipas [14].

2.7.1 Event Framework

A capacidade de alargar o comportamento predefinido do *Microsoft Dataverse* depende da deteção da ocorrência de eventos no servidor. A *Framework* de Eventos (*Event Framework*) fornece a capacidade de registar código personalizado para ser executado em resposta a eventos específicos [15].

Event Framework representa a tecnologia que permite desenvolver e integrar as nossas *business logic* (lógica que gere o negócio) através de *plugins* no *Dataverse*, colocando mais 3 etapas ou eventos no processo de pedidos e respostas entre o cliente e o servidor (ver Figura 9).

Event Framework fornece um conjunto de eventos ou gatilhos (*triggers*) predefinidos, como a criação, a atualização, ou a eliminação de um registo, que podem ser utilizados para executar lógica personalizada. Quando ocorre um evento registado, o *plugin* associado é invocado e pode executar as ações para que foi programado. O registo do *plugin* é feito com uma ferramenta chamada *Plugin Registration Tool* que referenciaremos na secção 2.7.2 Passos para criar um Plugin.

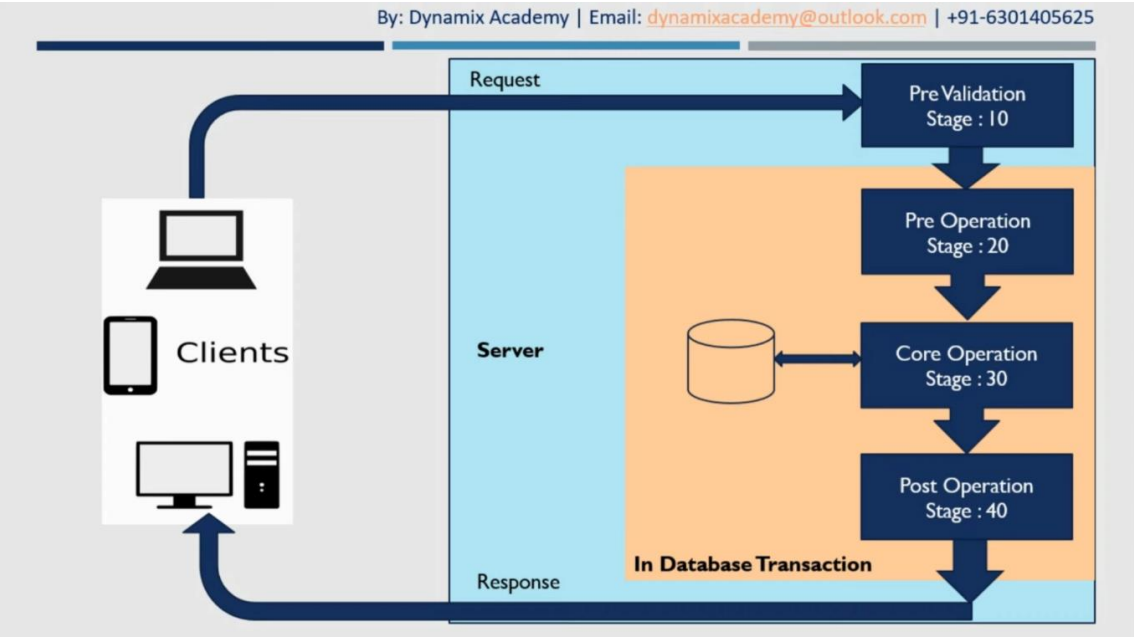


Figura 9 - Event Framework

Fonte (screenshot):
https://www.youtube.com/watch?v=ax3WiYODF_o&t=6s&ab_channel=AbhishekDhoriya%28DynamixAcademy%29

Como se pode ver na Figura 9, a *Framework* de Eventos permite a execução de *plugins* em diferentes etapas do ciclo de vida da operação e essas etapas são:

- ***Pre-validation*:** o *plugin* é executado antes da operação principal e fora da transação da base de dados. Exemplo: a operação principal é eliminar (*Delete*) de uma conta. O *plugin* implementa uma lógica de validação para impedir a eliminação dessa conta se esta tiver oportunidades associadas ou encomendas ativas.

- ***Pre-operation*:** o *plugin* é executado antes da operação principal, mas dentro da transação da base de dados. Exemplo: a operação principal é atualizar (*Update*) um registo da tabela Ordem de Venda. O plugin calcula o desconto aplicável com base na quantidade da encomenda ou no tipo de cliente. Atualiza o registo com o valor do desconto calculado antes das alterações serem guardadas.

- ***Core-operation*:** Os *plugins* não podem ser registados na etapa *Core- operation*. A *Core-operation* está reservada ao funcionamento da plataforma e não podem ser registados *plugins* para execução durante esta etapa.

- ***Post-operation*:** o *plugin* é executado depois da operação principal e dentro da transação da base de dados. Exemplo: a operação principal é criar (*Create*) um novo contacto na tabela Contacto. O *plugin* após a criação do registo, envia um e-mail de notificação à equipa de vendas, informando-a do novo contacto criado.

Finaliza-se esta secção referindo que apenas os *Post- operation plugins* podem ser executados de forma síncrona ou assíncrona, os restantes apenas de forma síncrona. Os *plugins* assíncronos são executados em segundo plano, permitindo que a operação principal seja concluída sem esperar que o *plugin* termine a sua execução. Isso proporciona mais flexibilidade e pode melhorar o desempenho geral do sistema. No entanto, é importante notar que os *plugins* assíncronos, tal como os síncronos, continuam a ter um tempo limite de 2 minutos para serem executados.

2.7.2 Passos para criar um Plugin

A criação de um *plugin* envolve uma serie de passos que merecem ser mencionados. Vamos utilizar como exemplo a criação de um *plugin* de pré-validação, que será acionado antes da criação de um registo na tabela de contas (*Account*). O objetivo do *plugin* é verificar se o campo *name* da tabela *Account* possui mais de 10 caracteres e caso positivo, enviar uma mensagem de erro ao utilizador.

O processo de criação do *plugin* segue então os seguintes passos:

1. **Criar um novo projeto no *Visual Studio*:** Cria-se uma nova biblioteca de classes C# (*.NET Framework*), tendo em consideração a versão compatível com o *Dataverse*. De momento, julho 2023, é recomendado usar o *.NET Framework 4.6.2*.
2. **Assinar os *assemblies*:** Utiliza-se a ferramenta *NuGet Packages* para instalar o pacote "*Microsoft.CrmSdk.CoreAssemblies*", isso irá instalar e adicionar os assemblies

necessários no projeto. Os assemblies são conjuntos de bibliotecas de código que contêm funcionalidades específicas. Ao adicionar um *assembly* específico ao projeto do *plugin*, estamos a incluir as bibliotecas necessárias para interagir com o *Dataverse*. Essas bibliotecas fornecem classes, métodos e propriedades que permitem aceder e manipular os dados e comportamentos do sistema [16].

3. **Estabelecer a ligação no *Dataverse* com o *Plugin Registration Tool*:** O *Plugin Registration Tool* é uma ferramenta que permite gerir e configurar *plugins* no *Dataverse*. Estabelece-se a ligação fornecendo as credenciais da organização a que o utilizador pertence e seleciona-se o ambiente onde o *plugin* será invocado (ver Figura 10 e Figura 11).

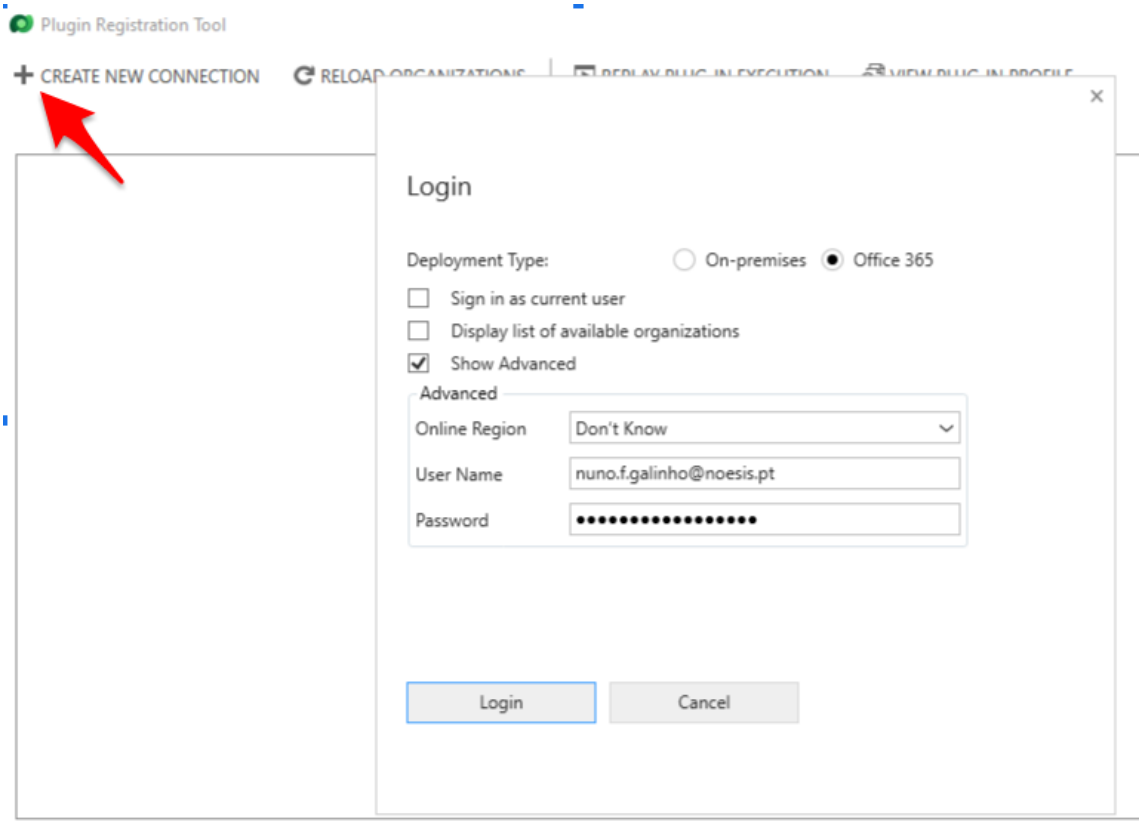


Figura 10 - Plugin Registration Tool: Credenciais de ligação ao Dataverse

A figura 10 mostra uma *interface* da Ferramenta de Registo de *Plugins* usada para gerir *plugins*. O *screenshot* representa a secção onde os utilizadores inserem as credenciais necessárias para estabelecer uma ligação com o *Dataverse*. Ao fornecer informações de autenticação, a Ferramenta fica habilitada a interagir com o *Dataverse* no ambiente onde o *plugin* será acionado.

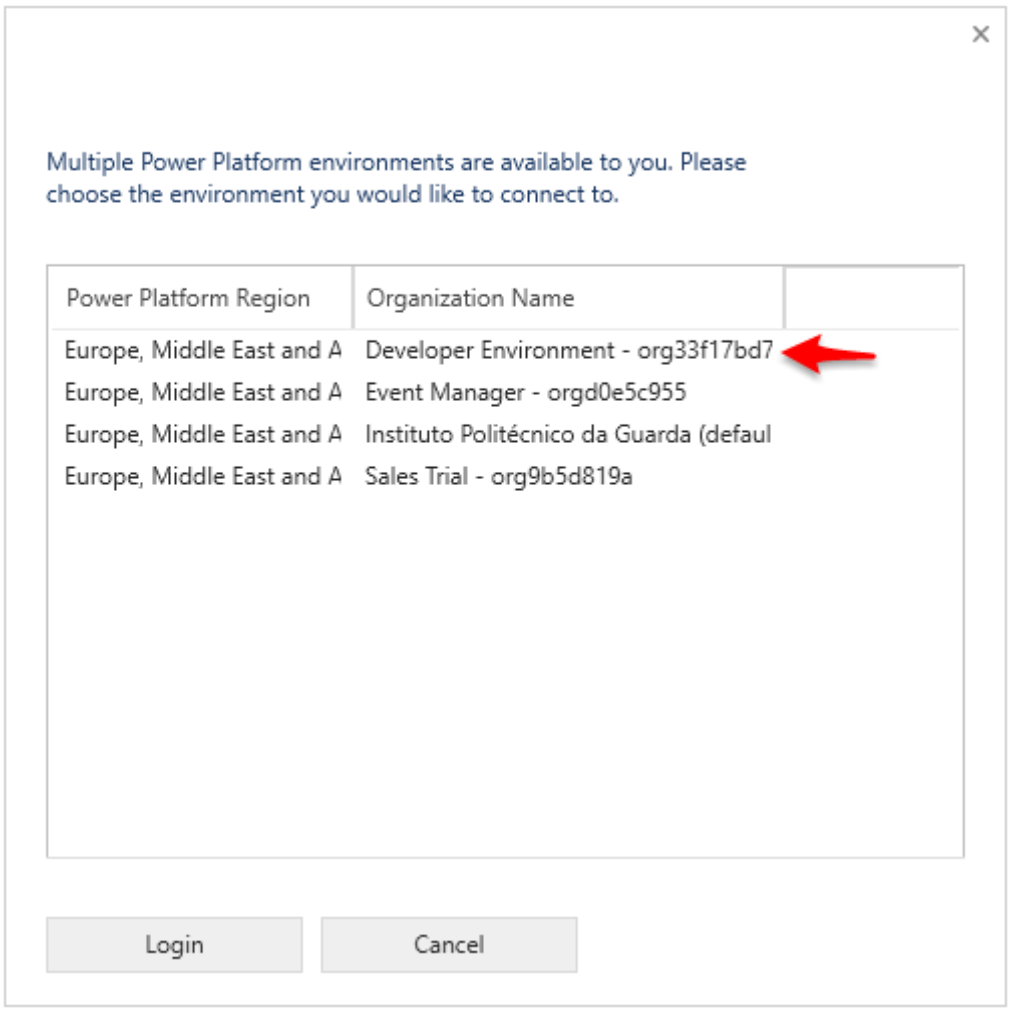


Figura 11 - Plugin Registration Tool: Escolha do ambiente

A figura 11 mostra outra *interface* da Ferramenta de Registo de *Plugins*. Neste *screenshot*, os utilizadores podem selecionar o ambiente específico onde desejam registar e configurar os seus *plugins*. A opção de escolher o ambiente é essencial quando uma organização tem várias instâncias do *Dataverse* para diferentes ambientes. Ao selecionar o ambiente correto, os utilizadores garantem que os *plugins* são aplicados apenas à instância pretendida.

4. **Registar o novo *assembly*:** Procura-se o *assembly* gerado na pasta "*bin*" do projeto e efetua-se o registo utilizando o *Plugin Registration Tool*. Isso permite que o *Dataverse* reconheça o *plugin* e possa utilizá-lo (ver Figura 12).

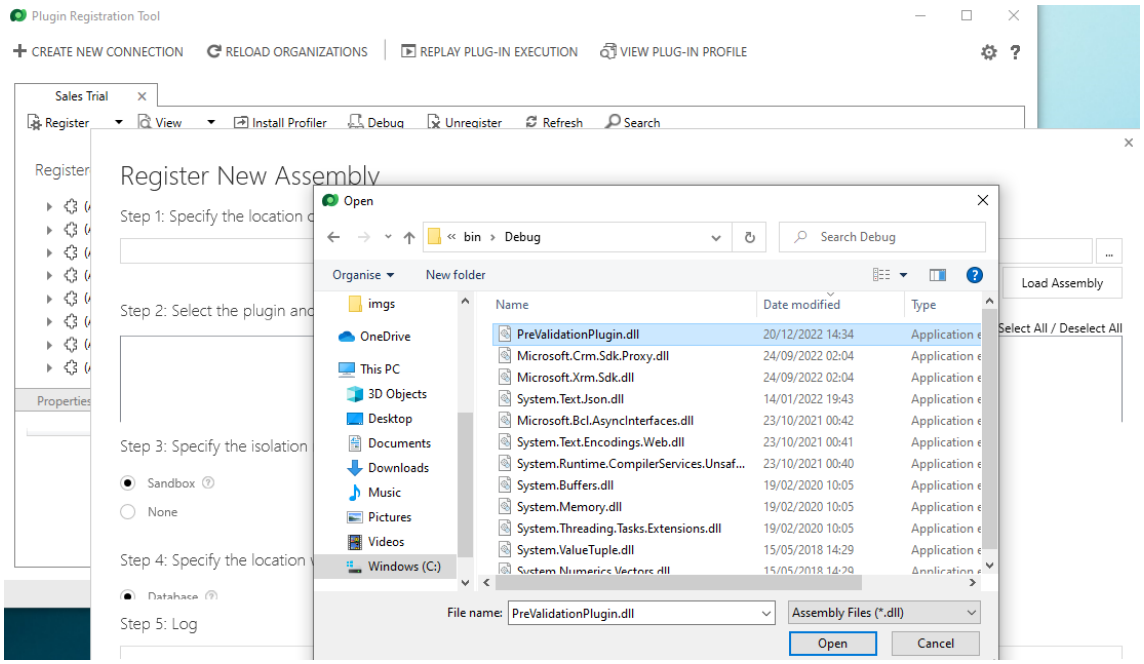


Figura 12 – Registo de um assembly

5. **Registar o novo passo:** Após registar o *assembly*, cria-se um novo passo (*step*) dentro do *assembly*. Um *assembly* pode ter vários passos, cada um com uma lógica específica a ser executada, de acordo com a ordem e circunstância (ver Figura 13).

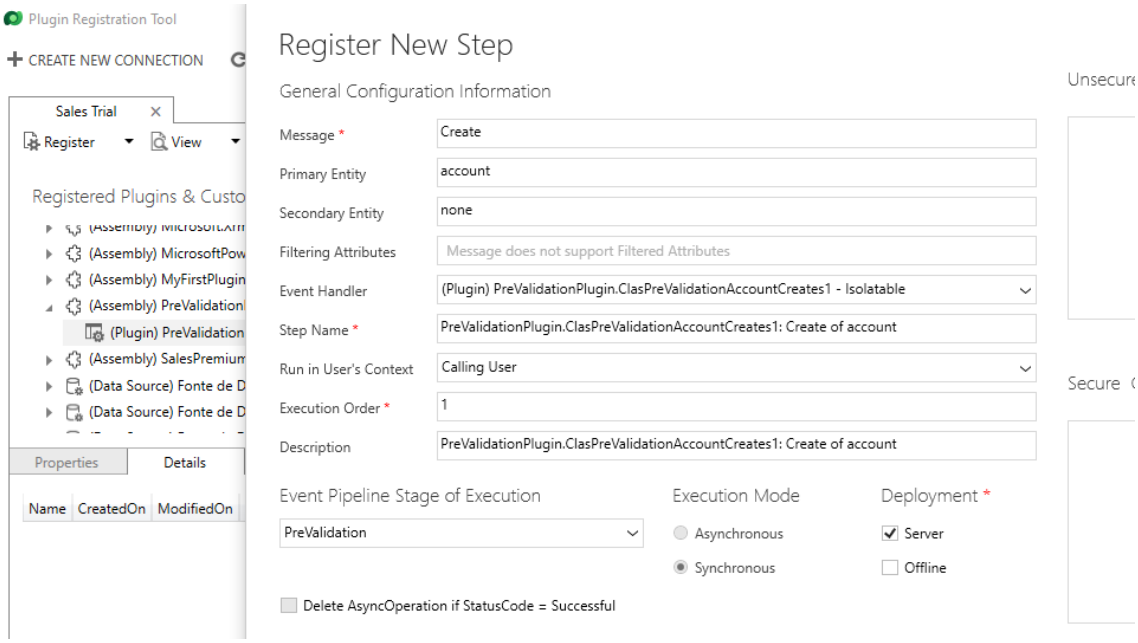


Figura 13 – Registo de um passo no assembly

6. **Programar a lógica do *plugin* no *Visual Studio*:** Utiliza-se o *Visual Studio* para desenvolver a lógica do *plugin*, definindo as ações a serem executadas durante a operação desejada. Após escrever o código, faz-se o *rebuild* do projeto e atualiza-se o *assembly*.

7. **Atualizar o *assembly*:** Por último, utiliza-se o *Plugin Registration Tool* para fazer o "*Update Assembly*" e testa-se o *plugin* na aplicação para verificar o seu funcionamento (ver Figura 14).

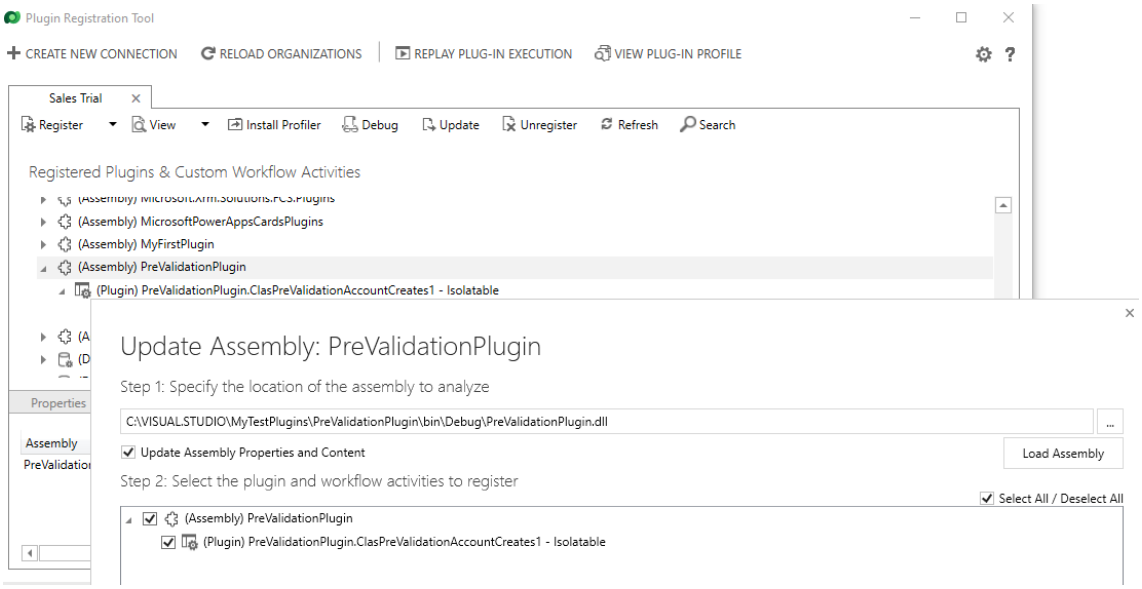


Figura 14 - Atualizar assembly

Abaixo encontra-se o código genérico do exemplo mencionado. O *plugin* verifica o campo "*name*" na tabela "*Account*" se tiver mais de 10 caracteres, envia uma mensagem de erro ao utilizador.

```
using System;
using Microsoft.Xrm.Sdk;

namespace MyPlugin
{
    public class PreValidationPlugin : IPlugin
    {
        public void Execute(IServiceProvider serviceProvider)
        {
            // Obtém o contexto de execução
            IPluginExecutionContext context =
(IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExec
utionContext));
            IOrganizationServiceFactory serviceFactory =
(IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrgani
zationServiceFactory));
            IOrganizationService service =
serviceFactory.CreateOrganizationService(context.UserId);

            // Verifica se a mensagem é de criação e no registo da
tabela "Account"
            if (context.MessageName.ToLower() == "create" &&
context.PrimaryEntityName.ToLower() == "account")
            {
                // Obtém os dados do registo criado
                Entity account =
(Entity)context.InputParameters["Target"];
                string name =
account.GetAttributeValue<string>("name");

                // Verifica o tamanho do campo "name"
                if (name.Length > 10)
                {
                    // Cria uma mensagem de erro
                    throw new InvalidPluginExecutionException("O campo
'name' deve ter no máximo 10 caracteres.");
                }
            }
        }
    }
}
```

3. Metodologia

Neste capítulo, será apresentada a metodologia utilizada no desenvolvimento do projeto, bem como uma descrição das tarefas realizadas ao longo do processo.

3.1 Metodologia Aplicada

No desenvolvimento da aplicação "*Event Manager*", foi usada a metodologia Ágil que tem como objetivo acelerar e agilizar o desenvolvimento do software, através de iterações e melhorias contínuas [17].

Referindo Wikipédia [18], "Um processo de desenvolvimento de *software* é um conjunto de atividades, parcialmente ordenadas, com a finalidade de obter um produto de *software*."

Na metodologia Ágil, o processo de desenvolvimento é dividido em iterações chamadas de *sprints*, que são períodos pré-definidos em que tarefas e funcionalidades específicas são trabalhadas. A duração de cada *sprint* pode variar de acordo com o projeto.

É relevante mencionar que durante o desenvolvimento desta aplicação, foram utilizados protótipos para validar e iterar as funcionalidades pretendidas. Os protótipos permitem visualizar antecipadamente a aparência e o funcionamento da aplicação, possibilitando a coleta de *feedback* e fazer ajustes antes e durante a etapa de desenvolvimento.

A abordagem baseada em protótipos e *sprints* oferece benefícios como responsividade, adaptação às mudanças de requisitos e a entrega contínua de valor ao cliente.

3.2 Descrição das tarefas

As principais tarefas do planeamento e desenvolvimento deste trabalho foram as seguintes:

1. **Estudo das Tecnologias** – investigação e aquisição de conhecimentos sobre *Power Platform, Dataverse* e *Dynamics 365*, este estudo foi a parte essencial do período de estágio e foi posteriormente enriquecido, estruturado e aplicado no decurso do desenvolvimento da aplicação "*Event Manager*". Este estudo foi já abordado na secção 2 deste relatório.

2. **Análise de requisitos** – definição de casos uso da aplicação, criação de protótipos e *data model*. Esta análise encontra-se documentada na secção 4 deste relatório.

3. **Recolha de documentação e tutoriais** – consulta de documentação oficial sobre *canvas apps da Microsoft* e vídeo-tutoriais no *Youtube*.

4. **Desenvolvimento da aplicação** – o desenvolvimento da aplicação foi um processo iterativo que envolveu a revisão de conceitos adquiridos na tarefa 1 assim como a aquisição de novos conhecimentos. É relevante referir a importância que as plataformas de inteligência artificial *ChatGPT* e *Microsoft Bing* tiveram no processo de aprendizagem, no anexo A 3 podem encontrar-se alguns *prompts* utilizados. A implementação da aplicação encontra-se documentada na secção 5 deste relatório.

5. **Escrita do relatório** – elaboração e formatação do relatório.

3.3 Atividades dos *sprints*

Nesta secção apresentam-se as tabelas que detalham as atividades realizadas em três *sprints*, durante o processo de desenvolvimento da aplicação. Os restantes *sprints* seguiram a mesma metodologia de atividade.

3.3.1 Primeiro *Sprint*

No primeiro *sprint*, cujas tarefas estão representadas na Tabela 1, definiram-se os casos de uso e a criação dos protótipos dos ecrãs: *Screen_Home, Screen_Clients e Screen_Organizers* da aplicação *Event Manager*. Os protótipos criados podem ser consultados no anexo A 4 .

Tabela 1 – Tarefas do primeiro Sprint

To Do	***Doing***	***Done***
Criar protótipo *Screen_Organizers*.	Criar protótipo *Screen_Clients*.	Definir atores.
		Criar casos uso.
		Criar protótipo *Screen_Home*.

3.3.2 Segundo *Sprint*

No segundo sprint foram criados o *Data Model*, as tabelas que a aplicação irá aceder e as relações entre elas. As tarefas desenvolvidas durante o segundo sprint podem ser consultadas na Tabela 2.

No processo de criação das tabelas e suas relações, foram usadas tabelas de dois tipos *standard* e *custom*. As tabelas *standard* são tabelas predefinidas que fazem parte integrante do esquema do *Dataverse*, enquanto as tabelas *custom* são tabelas criadas pelos utilizadores para atender às necessidades específicas da aplicação.

As tabelas *standard* usadas foram *Account* e *User* e visam representar Clientes e Organizadores respetivamente, enquanto as tabelas *custom* criadas foram *Event* e *Task* e visam representar os Eventos e as Tarefas atribuídas aos Organizadores, por cada Evento.

Estes conceitos serão de novo referidos, com mais detalhe, na secção 4.2 Data Model.

Tabela 2 - Tarefas do segundo Sprint

To Do	***Doing***	***Done***
Criar tabela *Task*.	Criar tabela *Event*.	Criar *Data Model*.
Relacionar tabela *Task* com tabela *Event* e tabela *standard User* (*Organizer*).	Relacionar tabela *Event* com tabela *standard Account* (*Client*).	

3.3.3 Terceiro *Sprint*

No terceiro *sprint* foram criados e programados os objetos:

- *gallery_Events* – a galeria que recebe os eventos da tabela *Event*.
- *formEventDetails* – o formulário com os campos do evento a criar ou atualizar.
- *ic_AddEvent* – o botão (*icon* "+") que cria formulário.
- *ic_SubmitEvent* – o botão (*icon* "*check*") que submete o formulário.
- *ic_deleteEvent* – o botão (*icon* "*trash*") que apaga o evento.

A Tabela 3 apresenta as tarefas desenvolvidas durante o terceiro sprint.

Tabela 3 – Tarefas do terceiro Sprint

To Do	***Doing***	***Done***
Criar *ic_AddEvent*.	Criar *formEventDetails*.	Criar *gallery_Events*.
Criar *ic_SubmitEvent*.		
Criar *ic_deleteEvent*		

4. Análise de Requisitos da aplicação Event Manager.

A análise de requisitos é um aspeto essencial no desenvolvimento de um produto e integra a primeira fase de desenvolvimento de um software. É durante esta fase que é feita a especificação do sistema e onde se define o que é que o sistema deve fazer – requisitos.

A importância da análise de requisitos reside na compreensão clara e precisa das necessidades dos utilizadores, permitindo assim que os engenheiros de *software* possam desenvolver um sistema que atenda aos requisitos e expectativas dos utilizadores.

Para recolher os requisitos usam-se Casos de Uso, que representam um modo de usar um sistema – funcionalidade que é realizada pelo sistema como resposta a uma interação por parte do ator. A análise de requisitos é um processo que envolve o estudo das necessidades do utilizador para se encontrar uma definição correta ou completa do sistema ou requisito de software [19]. Essa análise de requisitos é vital para o desenvolvimento do sistema, ela vai determinar o sucesso ou o fracasso do projeto.

4.1 Casos de uso

Casos de Uso são representações de interações entre atores e um sistema, descrevendo como o sistema é utilizado para realizar determinadas funcionalidades. Essas representações, descrevem as principais funcionalidades do sistema de forma clara e compreensível, identificando as ações que o ator realiza e as respostas esperadas do sistema. Por meio dos casos de uso, é possível especificar como o sistema se deve comportar em diferentes situações e quais são as interações esperadas com os utilizadores. A Tabela 4, apresentada a seguir, relaciona os atores com os objetivos pretendidos no uso do sistema a desenvolver.

Tabela 4 – Tabela de Casos de Uso do sistema Event Manager

Atores	Objetivos
Senior Organizer	**Consultar**: eventos, tarefas, organizadores, clientes. **Criar**: eventos, tarefas, clientes. **Atualizar**: eventos, tarefas, clientes. **Apagar**: eventos, tarefas, clientes. **Atribuir**: eventos a cliente, organizadores a eventos.
Executive Organizer	**Consultar**: eventos, tarefas, organizadores, clientes. **Criar**: tarefas. **Atualizar**: tarefas. **Apagar**: tarefas.
Júnior Organizer	**Consultar**: eventos, tarefas, organizadores, clientes.

Descrição dos atores:

- **Senior Organizer** – Organizador principal dos eventos. É um utilizador final do sistema, responsável por criar e gerir eventos, tarefas, organizadores e clientes;

- **Executive Organizer** – Organizador executivo dos eventos. É um utilizador final do sistema, responsável por criar e executar tarefas;

- **Júnior Organizer** – Colaborador da empresa. É um utilizador final do sistema, responsável por executar tarefas;

4.1.1 Diagrama de casos de uso

Segue-se o diagrama de casos de uso, a representação gráfica que ilustra as interações entre os atores e o sistema, mostrando as funcionalidades e serviços do sistema (ver Figura 15).

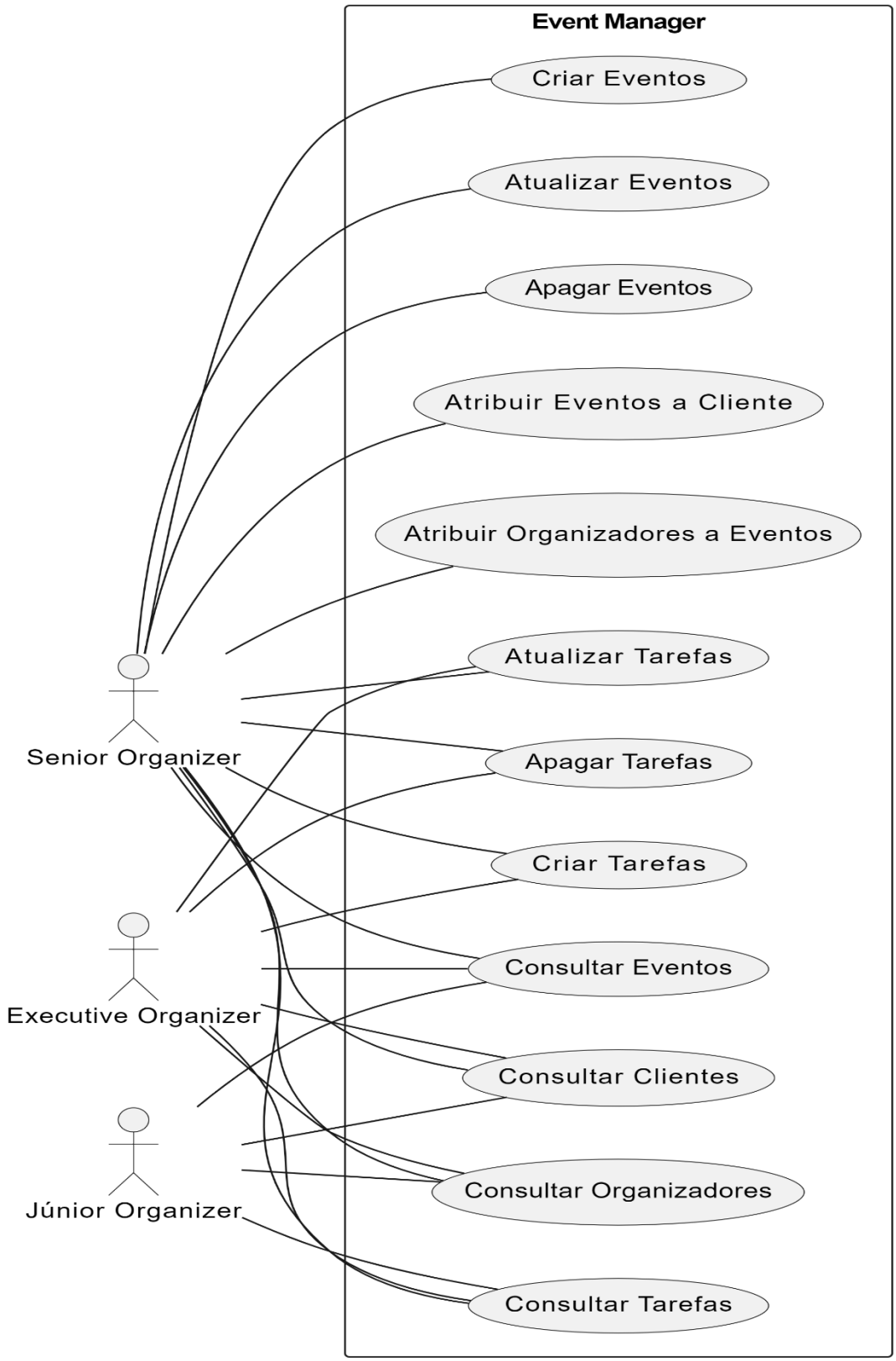


Figura 15 – Diagrama de casos de uso

4.1.2 Descrição de casos de uso

A descrição de casos de uso permite uma compreensão clara das funcionalidades do sistema e suas interações. Ela auxilia na identificação de requisitos específicos, fornecendo uma visão abrangente e detalhada do sistema em desenvolvimento.

No processo de descrição, são estabelecidos os passos necessários para que um caso de uso seja executado com sucesso, representando a interação entre o utilizador e o sistema de acordo com o esperado. Isso inclui as ações realizadas pelo ator e as respostas fornecidas pelo sistema em cada etapa.

A descrição de um caso de uso é constituída por:

• **Nome**: O nome do caso de uso descreve de forma sucinta a funcionalidade ou ação específica que será abordada na descrição.

• **Descrição**: É uma breve e clara explicação do caso de uso em questão, destacando o objetivo ou a finalidade que se pretende alcançar.

• **Pré-Condição**: A pré-condição é uma condição inicial que precisa ser atendida para que o caso de uso ocorra com sucesso. Pode ser algum estado ou requisito que deve estar presente antes da execução do caso de uso.

• **Caminho Principal**: O caminho principal descreve passo a passo como o utilizador e o sistema interagem quando o caso de uso ocorre conforme o esperado. É uma sequência de eventos que detalha as ações realizadas pelo utilizador e as respostas fornecidas pelo sistema.

• **Caminhos Alternativos**: Os caminhos alternativos representam as diferentes transações que podem ocorrer em determinado passo do caminho principal. Eles descrevem cenários alternativos que podem surgir em resposta a ações específicas do utilizador ou condições excecionais.

• **Pós-Condição**: A pós-condição é a condição em que o sistema se encontra após a conclusão do caso de uso. Descreve o estado ou resultado obtido pelo sistema após a execução bem-sucedida do caso de uso.

4.1.2.1 Caso de Uso – Criar Evento

A Tabela 5 descreve o processo de criação de um evento na aplicação Event Manager.

Tabela 5 – Caso de Uso - Criar Evento

Nome	Criar Evento.
Descrição	Este caso de uso descreve o processo de criação de um evento.
Pré-Condição	O utilizador tem de ter privilégios de escrita na tabela *Event,* tem de ser *Senior Organizer (S.O)*.
Caminho Principal	1. Utilizador clica Add_Event (*icon* +) 2. Sistema apresenta formulário vazio para registo de evento. 3. Utilizador preenche formulário (*Name, Client, Cost…*) e clica "Save" (*icon check*). 4. Sistema regista Evento.
Caminhos Alternativos	3.a) Campo *Name* encontra-se vazio. 3.b) Sistema responde com mensagem de erro: "*Field required*".
Pós-Condição	Não tem.

4.1.2.2 Caso de Uso – Criar Tarefa

A Tabela 6 descreve o processo de criação de uma tarefa na aplicação Event Manager.

Tabela 6 - Caso de Uso – Criar Tarefa

Nome	Criar Tarefa.
Descrição	Este caso de uso descreve o processo de criação de uma tarefa.
Pré-Condição	O utilizador tem de ter privilégios de escrita na tabela *Task* e *Organizer,* tem de ser *Senior Organizer (S.O)* ou *Executive Organizer (E.O)*.
Caminho Principal	1. Utilizador seleciona evento ao qual pretende atribuir tarefa. 2. Sistema apresenta os detalhes do evento. 3. Utilizador clica na *combo box Organizers*. 4. Sistema lista os organizadores existentes. 5. Utilizador seleciona organizadores e clica botão *RelateOrganizersWithEvents* (*icon check*). 6. Sistema relaciona organizadores com evento e apresenta os organizadores na galeria da aplicação. 5. Utilizador clica botão *AddTask* (*icon* +). 6. Sistema apresenta formulário vazio para registo de tarefa.

	7. Utilizador preenche formulário (*Name, Priority, Status Organize*) e clica "Save" (*icon check*). 8. Sistema regista tarefa.
Caminhos Alternativos	7.a) Campo *Name* encontra-se vazio. 7.b) Sistema responde com mensagem de erro: "*Field required*".
Pós-Condição	Não tem.

4.1.2.3 Caso de Uso – Atribuir Eventos a Cliente

A Tabela 7 descreve o processo de atribuição de um evento a um cliente.

Tabela 7 - Caso de Uso – Atribuir Eventos a Cliente

Nome	Atribuir evento a cliente.
Descrição	Este caso de uso descreve o processo de atribuição de um evento a um cliente
Pré-Condição	O utilizador tem de ter privilégios de escrita na tabela *Event* e *Client*, tem de ser *Senior Organizer (S.O)* ou *Executive Organizer (E.O)*.
Caminho Principal	1. Utilizador seleciona ecrã clientes. 2. Sistema apresenta ecrã clientes. 3. Utilizador seleciona cliente ao qual pretende atribuir evento. 4. Sistema apresenta os detalhes do cliente. 5. Utilizador clica na *combo box AssociateEvents*. 6. Sistema lista os eventos existentes. 7. Utilizador seleciona eventos e clica botão *RelateClientsWithEvents* (*icon check*). 8. Sistema relaciona cliente com eventos e apresenta os eventos na galeria.
Caminhos Alternativos	1.a) Outra forma de atribuir evento a cliente: o utilizador seleciona ecrã *Home* e atualiza detalhes do evento.
Pós-Condição	Não tem.

4.2 Data Model

Um *Data Model* é uma representação estruturada e abstrata das informações a ser armazenadas e manipuladas num sistema. Ele define a estrutura, os tipos de dados e os relacionamentos entre eles, permitindo a organização e o acesso eficiente aos dados. O *Data Model* serve como uma base para o desenvolvimento de um sistema, fornecendo uma visão clara e precisa das entidades, atributos e suas interações.

Um *Data Model* é um diagrama que representa a estrutura lógica dos dados num sistema. Ele é usado para descrever os objetos ou entidades relevantes para um domínio, bem como os relacionamentos entre eles [20].

A Figura 16 apresenta o *Data Model* do sistema *Event Manager*. Nesta imagem, é possível visualizar a estrutura da base de dados utilizada pela aplicação, incluindo as tabelas e suas respetivas relações.

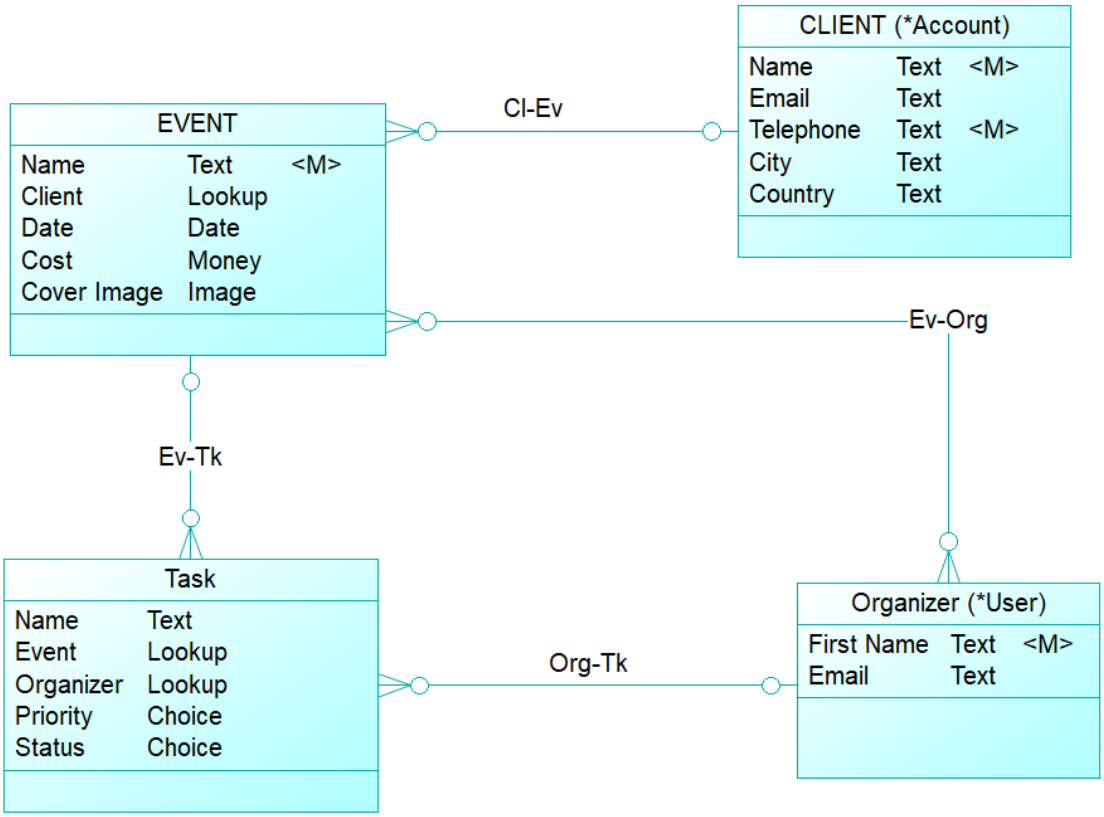


Figura 16 – Data Model do sistema Event Manager

O *Data Model* representado na Figura 16 é constituído por 2 tipos de tabelas no contexto do *Dataverse*: *Custom* e *Standard*. *Task* e *Event* são tabelas personalizadas (*custom*), enquanto *Client (*Account)* e *Organizer (*User)* são tabelas padrão (*standard*). No esquema do *Dataverse* as tabelas padrão, são fornecidas pelo sistema, aquando da criação do ambiente (*Environment*) onde a aplicação será desenvolvida.

As tabelas personalizadas, são criadas pelos utilizadores para atenderem às necessidades específicas da aplicação, enquanto as tabelas padrão, são projetadas para atenderem às necessidades comuns da maioria das aplicações como: armazenar informações sobre clientes (tabela "*Account*"), armazenar detalhes de contactos (tabela "*Contact*"), ou ainda, armazenar informações sobre os utilizadores do sistema (tabela "*User*").

A tabela *User* tem uma importância especial, porque é a partir dela que a organização pode definir como os utilizadores podem aceder à base de dados, por exemplo, pode atribuir diferentes níveis de acesso a diferentes tabelas com base nas responsabilidades e necessidades de cada utilizador. Foi a partir da tabela *User* que foram definidos as *security roles* dos Organizadores dos Eventos, para esta aplicação.

Segue-se a representação detalhada da tabela *Event* (Tabela 8), tabela *Client* (Tabela 9)**,** tabela *Task* (Tabela 10) **e** tabela *Organizer* (Tabela 11).

Tabela 8 – Descrição da tabela Event

Coluna	Tipo de Dados	Restrição	Obrigatório	Descrição
Name	Text	16 carateres	Sim	O nome do evento
Client	Lookup	--	Não	O cliente associado ao evento
Date	Date	--	Não	A data do evento
Cost	Money	[0, 999.999]	Não	O custo do evento
Cover Image	Image	--	Não	A imagem de capa do evento

A tabela *Event* representa os eventos a serem geridos pela aplicação. Está relacionada com a tabela *Client* em muitos para um (n-1), está relacionada com a tabela *Task* em um para muitos (1-n) e está relacionada com a tabela *Organizer* em muitos para muitos (n-n).

De destacar o campo, *Client*, cujo tipo de dados é *Lookup,* este é um campo que aponta para a tabela *Client,* é um campo de pesquisa, na aplicação implementa-se como um elemento *dropdown* que lista o nome de todos os clientes presentes na tabela *Client*.

Tabela 9 – Descrição da tabela Client

Coluna	Tipo de Dados	Restrição	Obrigatório	Descrição
Name	Text	16 carateres	Sim	O nome do cliente
Email	Text	---	Não	O email do cliente
Telephone	Text	16 carateres	Sim	O telefone do cliente
City	Text	16 carateres	Não	A cidade do cliente
Country	Text	16 carateres	Não	O país do cliente

A tabela *Client* representa os clientes na aplicação *Event Manager* e está relacionada com a tabela *Event* em um para muitos (1-n).

Tabela 10 – Descrição da tabela Task

Coluna	Tipo de Dados	Restrição	Obrigatório	Descrição
Name	Text	16 carateres	Sim	O nome da tarefa
Event	Lookup	---	Não	O evento associado à tarefa
Organizer	Lookup	---	Não	O organizador associado à tarefa
Priority	Choice	---	Não	A prioridade da tarefa
Status	Choice	---	Não	O estado da tarefa

A tabela *Task* representa as tarefas associadas aos eventos e aos organizadores que as vão executar. Está relacionada com a tabela *Event* em muitos para um (n - 1) e com a tabela *Organizer* também em muitos para um (n - 1).

De destacar os campos, *Event* e *Organizer*, do tipo *Lookup* e ainda os campos *Priority* e *Status* do tipo *choice*. O campo *Event* aponta para a tabela *Event* e o campo *Organizer* aponta para a tabela *Organizer*, ambos permitem pesquisas nas respetivas tabelas para onde apontam. Os campos *Priority* e *Status*, têm um *set* de valores fixos que foram definidos na altura das suas criações.

Tabela 11 – Descrição da tabela Organizer

Coluna	Tipo de Dados	Restrição	Obrigatório	Descrição
First Name	Text	16 carateres	Sim	O primeiro nome do organizador
Email	Text	---	Não	O email do organizador

A tabela *Organizer* representa os organizadores dos eventos e está relacionada com a tabela *Task* em um para muitos (1-n).

4.3 Diagramas de Sequência

Os diagramas de sequência, modelam as interações entre objetos num único caso de uso. Ilustram como as diferentes partes dum sistema interagem entre si para realizar uma função e a ordem em que as interações ocorrem quando um determinado caso de uso é executado [21].

Diagrama de Sequência - Criar tarefa

A Figura 17 mostra a sequência de interações entre o utilizador, sistema e os objetos envolvidos de forma a criar uma tarefa.

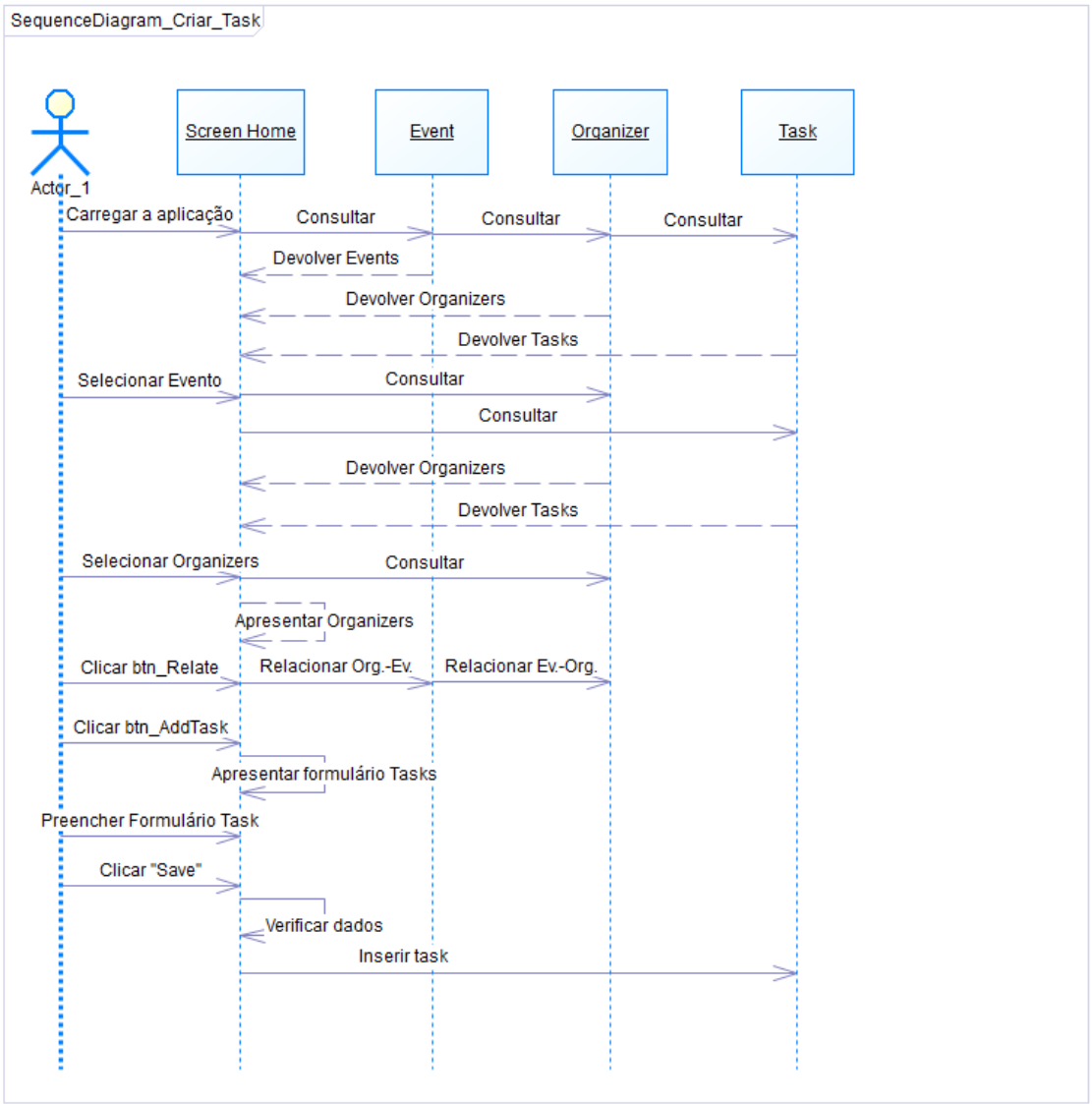


Figura 17 - Diagrama de Sequência - Criar tarefa

Quando o ator carrega a aplicação, o sistema consulta os eventos, organizadores e tarefas, em seguida devolve os eventos, os organizadores do primeiro evento da lista e as tarefas atribuídas a esse evento. Quando o ator seleciona um evento, o sistema consulta as tabelas *Organizer* e *Task* e em seguida devolve os organizadores e tarefas relacionadas com esse evento. O ator seleciona os organizadores que pretende clicando na *combo box* respetiva, o sistema consulta a tabela *Organizer* e apresenta os organizadores ao ator. O ator clica no botão *RelateOrganizers*

(*icon* "*check*") e o sistema relaciona os organizadores com o evento e o evento com os organizadores selecionados (relação n *Event* – n *Organizer*). O ator clica no botão *Add_Task* (*icon* "+") e o sistema apresenta o formulário das tarefas pronto a receber *inputs*. O ator preenche o formulário, clica *Save/Submit* (*icon* "*check*"), o sistema verifica se o campo *Name* (nome da tarefa) está preenchido e se sim insere tarefa na tabela *Task.* A Figura 18 representa o interface do ecrã *Home*, da aplicação *Event Manager* e os passos sequenciais necessários para criar uma tarefa.

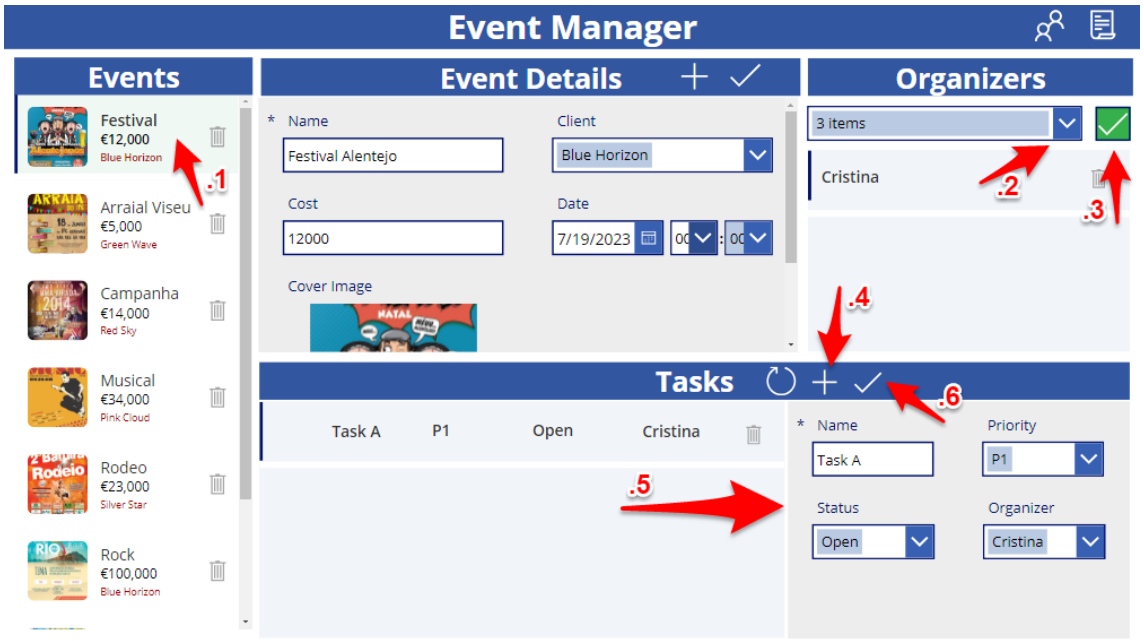


Figura 18 – Passos para criar uma tarefa na aplicação Event Manager

A Figura 18 mostra as interações do utilizador com a aplicação, desde a seleção do evento ao qual queremos atribuir tarefas, seleção dos organizadores que irão desempenhar essas tarefas, o preenchimento dos campos do formulário de tarefas, até a confirmação da criação da tarefa.

5. Implementação

Neste capítulo será apresentado o processo de implementação da *canvas app Event Manager*.

O primeiro requisito para criar uma power app é ter uma conta. A conta da presente aplicação foi criada através do *Office 365* que o IPG (Instituto Politécnico da Guarda) oferece a todos os seus estudantes.

Todo e qualquer projeto na *Power Platform* tem de estar incluída num ambiente de desenvolvimento um *Environment*, o *Environment* com mais permissões que é possível criar com uma conta criada a partir do *office 365* é o ambiente do tipo *Developer*.

Uma das principais restrições do ambiente tipo *Developer* é a limitação de permissões. Por exemplo, algumas funcionalidades avançadas podem estar desativadas, como integração com serviços externos e acesso a recursos sensíveis do sistema. Além disso, pode haver restrições em relação ao número de utilizadores que podem aceder á aplicação ou ao armazenamento de dados [22].

Nas secções seguintes serão descritas implementações como:

- Criação do ambiente de trabalho.
- Criação do *Data Model* e tabelas.
- Criação e programação dos controlos da aplicação.
- Criação de *Plugins*.

5.1 Criação do ambiente de trabalho

A criação de Ambientes (*Environments*) e Soluções (*Solutions*) permitem uma gestão eficiente das aplicações e objetos na *Power Platform*. Os Ambientes proporcionam um ambiente isolado para desenvolvimento, teste e produção, garantindo a separação adequada entre as etapas do ciclo de vida das aplicações. Já as Soluções permitem agrupar e organizar os componentes da aplicação, facilitando a implantação e manutenção [23].

Ao criar um *Publisher*, estabelecemos um registo das alterações realizadas na base de dados, permitindo o controlo e acompanhamento das ações efetuadas pelos utilizadores. Além disso, o prefixo adicionado a todos os objetos personalizados (*custom*) criados no *Dataverse*, identificam claramente o autor ou equipa responsável, pela criação dos mesmos.

No contexto do Ambiente de Desenvolvimento (*Developer Environment*), dentro da solução *Event Manager*, foi criado o *Publisher* com o nome "Nuno Galinho" e o prefixo "ng". A criação do *Publisher* referido encontra-se ilustrado na Figura 19.

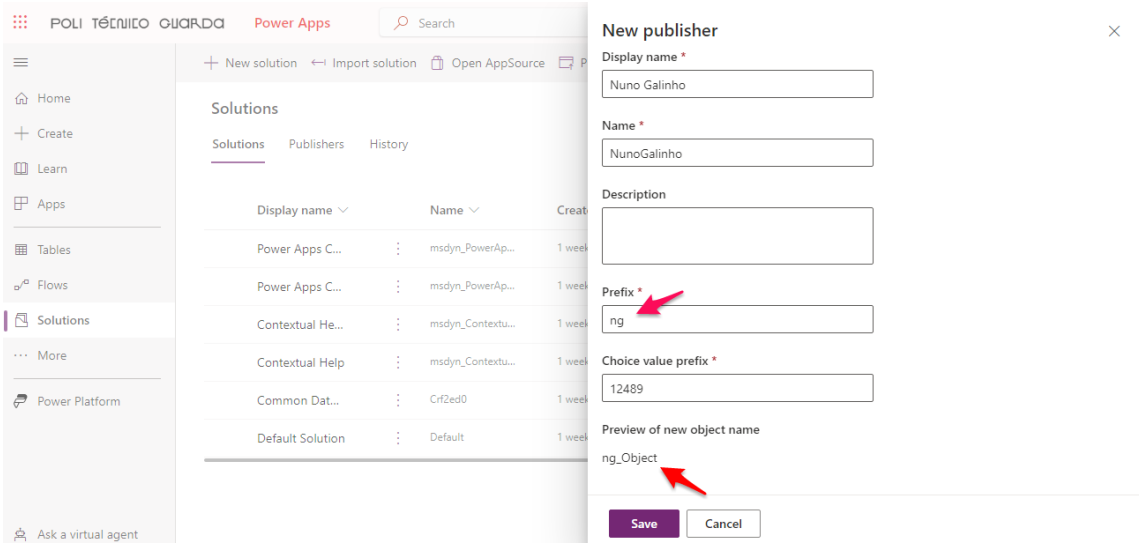


Figura 19 – Criação de um Publisher

Na Figura 19 vemos apontado o uso do prefixo "ng", essa prática permite evitar conflitos de nomes e tornar a identificação dos componentes mais clara e organizada. Por exemplo a tabela *custom* "Event" adquiriu o nome lógico "ng_event" após a sua criação.

5.2 Criação do Data Model e tabelas no Dataverse

A criação do *Data Model* é um passo fundamental para organizar e estruturar os dados de qualquer aplicação. Como referido na secção 4.2 Data Model, o modelo é constituído por quatro tabelas, duas tabelas personalizadas (*Event* e *Task*) e duas tabelas padrão (*User* e *Account*). Na Tabela 12 apresentam-se as relações entre elas.

Tabela 12 – Relação entre as tabelas do Data Model de Event Manager

Tabela	Relação	Descrição
Event	N – 1 Client	Um cliente pode estar relacionado com vários eventos.
	1 – N Task	Um evento pode ter várias tarefas associadas.
	N – N Organizer	Um evento pode estar relacionados com vários organizadores. Um organizador pode estar relacionado com vários eventos.
Task	N – 1 Event	Uma tarefa pode ter um só evento associado. Um evento pode ter várias tarefas.

	N – 1 Organizer	Uma tarefa pode ter um só organizador associado.
Client (*Account)	1 – N Event	Um cliente pode estar relacionado a vários eventos.
Organizer (*User)	1 – N Task	Um organizador pode ter várias tarefas associadas.
	N – N Event	Um organizador pode estar relacionados com vários eventos. Um evento pode estar relacionados com vários organizadores

Conhecidas as tabelas a criar, suas relações e a descrição detalhada dos campos de cada uma delas, o processo de criação das entidades e relacionamentos no *Dataverse* torna-se simples. A descrição detalhada dos campos de cada uma das tabelas encontra-se disponível na Tabela 8 – Descrição da tabela Event, Tabela 9 – Descrição da tabela *Client*, Tabela 10 – Descrição da tabela *Task* e Tabela 11 – Descrição da tabela *Organizer*. A Figura 20 e Figura 21 ilustram a criação de uma coluna na tabela *Event* e as relações criadas na tabela *Event*.

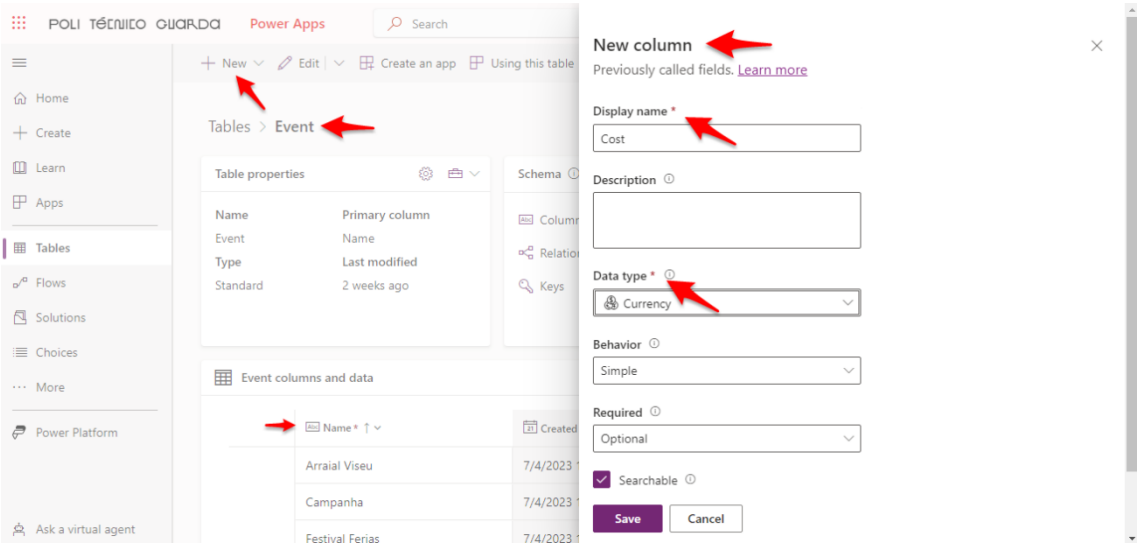


Figura 20 – Criação de uma coluna na tabela Event

A Figura 20 mostra o processo de criação de uma nova coluna na tabela "Event". Pode visualizar-se a interface do *Dataverse*, onde é adicionada a coluna “cost” com tipo de dados "Currency" à tabela "Event".

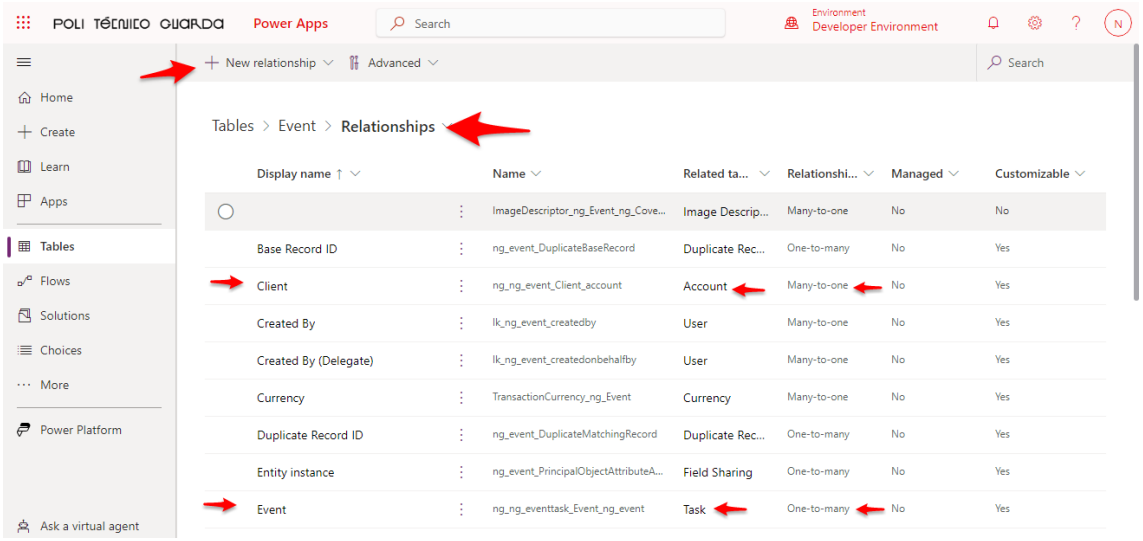


Figura 21 – Relações criadas na tabela Event

Na Figura 21, encontram-se assinaladas as relações criadas na tabela "Event". Destacam-se particularmente as relações muitos para um com a tabela *Client* (**Account*) e de um para muitos com a tabela *Task*.

5.4 Criação e programação dos controlos da aplicação

Depois de criada a base de dados, foram introduzidos alguns registos (*mock data*) nas tabelas. O objetivo foi testar imediatamente o acesso de alguns controlos da aplicação, como as galerias, à base de dados, assim que fossem incluídos na tela. Uma galeria é um controlo que contém outros controlos e que exibe uma coleção de dados. Segundo Microsoft [24] "Um controlo do tipo galeria pode mostrar vários registos de uma *data source* e cada registo pode conter vários tipos de dados".

A criação e programação dos controlos numa *canvas app* é um processo simples e intuitivo. O *Power Apps* oferece uma interface visual simples, baseado no princípio, arrastar e soltar, os controlos são adicionados na tela e as suas propriedades e lógica são configurados de acordo com os requerimentos da aplicação.

A linguagem de programação usada na criação das *Canvas Apps* é chamada *Microsoft Power Fx*. "É o novo nome para a linguagem de fórmulas para aplicações de tela em *Power Apps*. É uma linguagem *low code,* que os criadores podem trabalhar diretamente na barra de fórmulas, semelhante ao *Excel* ou na janela de texto do *Visual Studio Code*" [25].

Power Fx é uma linguagem declarativa, simples e baseada em fórmulas, projetada para ser utilizada no desenvolvimento de aplicações e automações dentro do ecossistema da *Power Platform*. A documentação do *Microsoft Power Fx* pode ser encontrada no site oficial da *Microsoft*, na seção de documentação da *Power Platform*.

Uma vez no ambiente de desenvolvimento do *Power Apps* procedeu-se á construção do *User Interface* baseado nos protótipos previamente criados (ver anexo A 4). O processo de desenvolvimento da aplicação seguiu a metodologia ágil descrita na secção 3.1 Metodologia Aplicada. A Tabela 1 – Tarefas do primeiro Sprint, Tabela 2 - Tarefas do segundo Sprint e Tabela 3 – Tarefas do terceiro Sprint, descrevem os 3 primeiros *sprints* desenvolvidos e representam a metodologia aplicada no desenvolvimento da aplicação. Os sprints seguintes seguem a mesma abordagem ágil.

Uma vez adicionados os controlos á tela, programaram-se os mesmos de acordo com as funcionalidades pretendidas. Por exemplo, a programação dos controlos referidos no *sprint 3* (Tabela 3), referem-se às operações CRUD dos eventos. Depois de adicionados e programados todos controlos do primeiro ecrã (*Screen_Home*) foram criados os dois ecrãs seguintes (*Screen_Clients* e *Screen_Organizers*), seguindo a mesma metodologia.

Apresentam-se a seguir, como exemplo, a Tabela 13, a Tabela 14 e Tabela 15 que descrevem todos os controlos envolvidos nas operações CRUD de Eventos, o respetivo código adicionado às propriedades dos controlos e a descrição das funções mais relevantes nas operações.

Tabela 13 – Descrição dos controlos envolvidos nas operações CRUD de Eventos

Controlo	Descrição
gallery_Events	*A* galeria que recebe os eventos da tabela *Event*.
formEventDetails	O formulário com os campos do evento a criar ou atualizar.
ic_AddEvent	O botão (*icon* "+") que cria formulário.
ic_SubmitEvent	O botão (*icon* "*check*") que submete o formulário.
ic_deleteEvent	O botão (*icon* "*trash*") que apaga o evento.

Tabela 14 – Programação dos controlos envolvidos nas operações CRUD de Eventos

Controlo	Propriedade	Fx
gallery_Events	*Items =*	*Events*
	TemplateFill =	*If (ThisItem.IsSelected, RGBA (1, 222, 244, 1), RGBA (0, 0, 0, 0));*
formEventDetails	*DataSource =*	*Events*
ic_AddEvent	*OnSelect =*	*NewForm (formEventDetails);*
ic_SubmitEvent	*OnSelect =*	*SubmitForm (formEventDetails);*
ic_deleteEvent	*OnSelect =*	*Remove (Events, ThisItem);*

Tabela 15 – Descrição das funções: NewForm, SubmitForm e Remove

Função	Descrição
NewForm (formEventDetails);	Limpa dados existentes no formulário *formEventDetails*, de forma a ficar pronto para receber a entrada de um novo registo.
SubmitForm (formEventDetails);	Recolhe todos os valores preenchidos nos campos do formulário *formEventDetails* e envia-os para o destino designado, neste caso para a tabela *Event*.
Remove (Events, ThisItem);	Remove um *item* específico de uma coleção ou tabela. Neste caso, refere-se à tabela "*Events*" da qual o *item* será removido, e "*ThisItem*" é uma referência ao *item* selecionado na galeria *gallery_Events* da qual é "filho".

Abaixo apresenta-se a Figura 22 que é o *screenshot* do ecrã *Home* (*Screen_Home*). Os pontos marcados na figura representam:

1. *gallery_Events* – a galeria que recebe os eventos da tabela *Event*.
2. *formEventDetails* – o formulário com os campos do evento a criar ou atualizar.
3. *ic_AddEvent* – o botão (*icon* "+") que cria formulário.
4. *ic_SubmitEvent* – o botão (*icon* "*check*") que submete o formulário.
5. *ic_deleteEvent* – o botão (*icon* "*trash*") que apaga o evento.

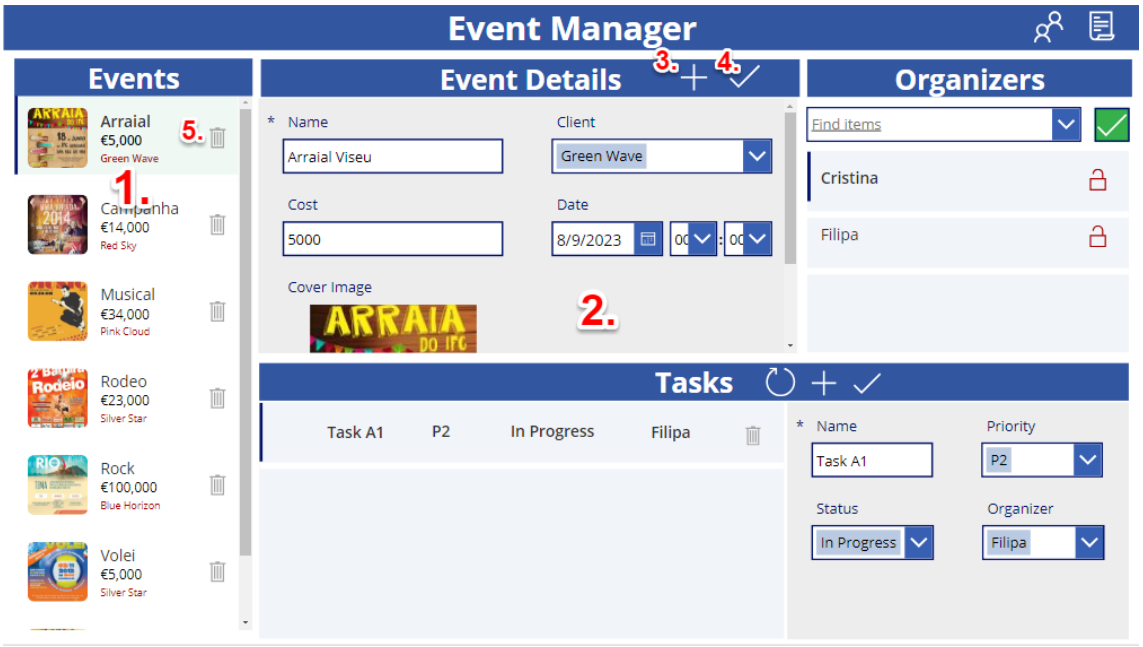


Figura 22 – Ecrã Home da aplicação Event Manager

No Anexo A 1 pode-se encontrar o código e respetivos *screenshots* dos controlos programados em cada um dos ecrãs. O anexo também inclui algumas notas pessoais sobre o processo de desenvolvimento da aplicação.

5.5 Criação de Plugins

Como explicado na secção 2.7 Plugins, a implementação de *plugins* permite estender e personalizar funcionalidades dentro de uma aplicação ou plataforma que utilize o *Dataverse* como Base de dados. Proporciona às organizações um alto nível de controlo e personalização sobre o seu ambiente, permitindo que elas atendam às necessidades específicas do negócio e melhorem a eficiência e a automação dos processos.

Na nossa aplicação foram implementados dois *plugins.* O primeiro *plugin* apaga todos as tarefas (*Tasks*) associadas a um evento (*Event*) quando esse evento é apagado. Uma vez que o *plugin* está a executar uma ação em resposta à eliminação de um evento, este deveria ser registado como um *plugin* de pré-operação. Um *plugin* de pré-operação, é executado após as verificações de segurança, mas antes da conclusão da operação principal, ou seja, antes do registo *Event* ser efetivamente eliminado. No entanto, quando o plugin foi registado na fase de pré-operação, simplesmente não eliminava as tarefas relacionadas ao evento.

Ao fazer-se o *debug* do código, verificou-se que o resultado da consulta das tarefas associadas aos eventos era nulo, o que impossibilitava a eliminação dos mesmos. O que nos levou a concluir que os dados da coluna *Event* (*lookup*) da tabela *Task,* estavam a ser eliminados antes da execução do *plugin*. A forma encontrada para garantir que o *plugin*, tinha acesso aos dados de que necessitava, para avaliar quais eram as tarefas que estavam associados ao evento eliminado, foi registar o *plugin* na fase pré-validação. Nesta fase o *plugin* é executado antes do evento ser eliminado e fora da transação da base de dados, o que significa que, quando se consultam as tarefas associadas ao evento que está a ser eliminado, os dados da coluna *Event* (*lookup*) da tabela *Task*, ainda estão disponíveis, pelo que a consulta devolve os resultados esperados e permite que o plugin tenha o desempenho esperado.

O segundo *plugin* cria um evento (*Event*) e uma tarefa (*Task*) quando um novo cliente (*Account*) é criado. O evento criado tem o nome "New client!" e fica associado ao cliente recém-criado. A tarefa tem o nome "Contact client!" e fica associada ao evento recém-criado. A tarefa fica ainda associada ao organizador # Nuno Filipe Galinho que é o *Senior Organizer* e é o responsável por contactar os novos clientes. Por fim o *plugin* associa o evento ao organizador. Uma vez que o *plugin* está a criar novos registos (eventos e tarefas) em resposta à criação de um cliente, este foi registado como um *plugin* de pós-operação. Desta forma, o *plugin* só será executado depois da conta ter sido criada com êxito na base de dados.

A Figura 23 e a Figura 24, mostram o resultado da execução do *plugin*, quando o cliente *Pink Sun* é criado, que é a operação que o invoca. O código dos *plugins* pode ser consultado nos anexos A 2.1 e A 2.2 .

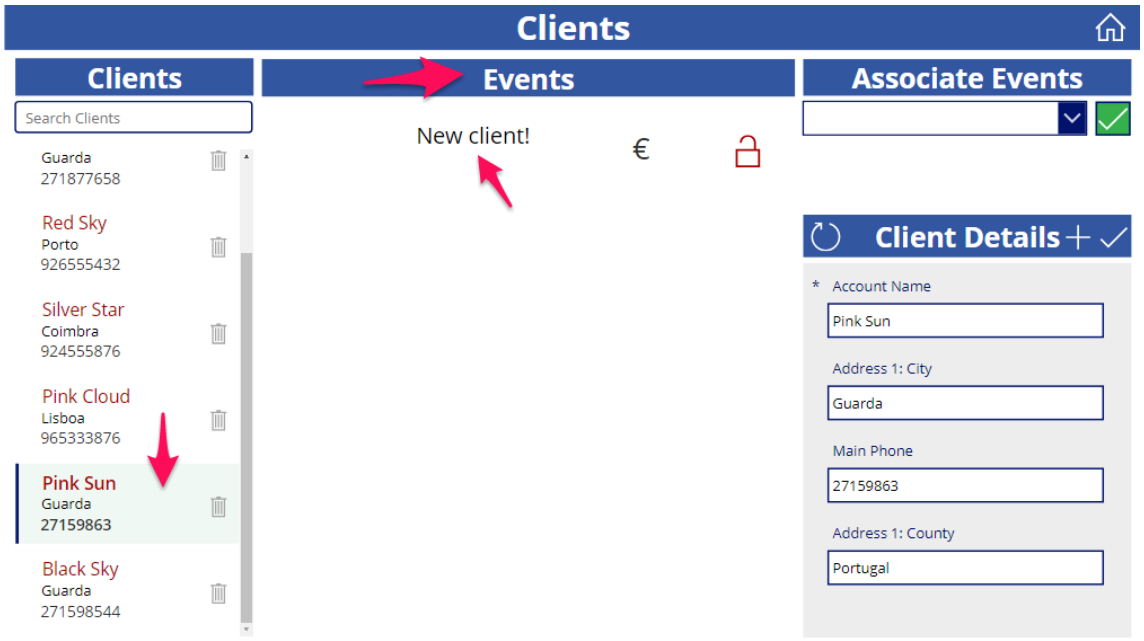


Figura 23 –Evento New Client criado por plugin

A Figura 23 representa o resultado da execução do *plugin* que cria automaticamente um novo evento de nome "New Client!" sempre que um novo cliente é criado.

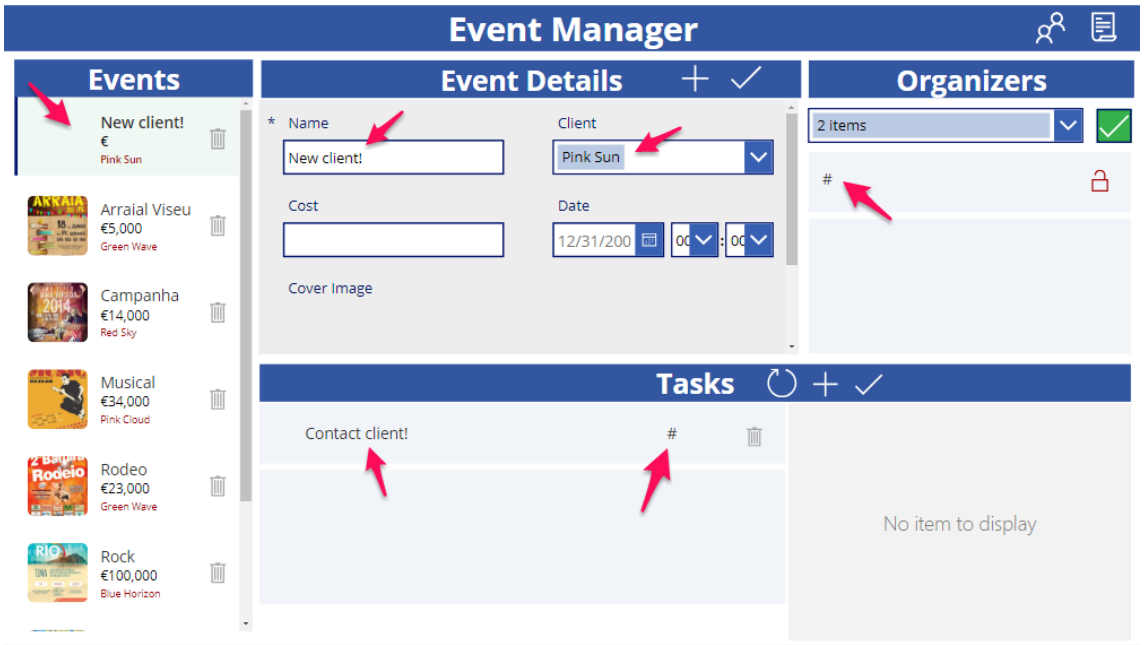


Figura 24 – Task Contact client associada a Senior Organizer criado por plugin

A Figura 24 representa o resultado da execução do *plugin* que cria automaticamente uma nova tarefa de nome "Contact Client!" e Organizer "# Nuno Galinho", associada ao Evento "New

Client!", sempre que um novo cliente é criado. "# Nuno Galinho" é o *Senior Organizer* responsável pelos novos clientes. O primeiro nome de "# Nuno Galinho" é "#" e por isso aparece assim representado na galeria de tarefas e galeria de organizadores. Como já referido a conta atribuída para desenvolvimento desta aplicação não permite eliminação ou edição de registos na tabela *User*, que representa os *Organizers* e a quem foram atribuídas diferentes permissões e acessos.

5.6 Segurança

A segurança numa aplicação *Power App*, pode ser estabelecida quer ao nível da aplicação quer ao nível dos dados. Ao nível da aplicação, são estabelecidos controlos de acesso e permissões por meio de grupos de segurança (*security groups*), a segurança ao nível dos dados é estabelecida usando *security roles.*

As regras de validação são implementadas para garantir a integridade dos dados inseridos pelos utilizadores. Ao nível dos dados, são utilizadas unidades de negócio (*Business Units*) e equipas (*Teams*) para organizar e controlar o acesso com base em hierarquias e estruturas organizacionais específicas. Os utilizadores são agrupados em unidades de negócio de acordo com departamentos ou áreas funcionais, e as equipas são criadas para colaboração em projetos ou tarefas específicas [26].

As *security roles* permitem estabelecer permissões detalhadas para utilizadores e grupos, restringindo o acesso a funcionalidades ou dados sensíveis. Combinando *security groups*, *security roles* e regras de validação é possível criar um ambiente seguro, protegendo tanto a aplicação quanto os dados armazenados.

5.6.1 Regras de validação

As regras de validação são utilizadas, essencialmente, para garantir a integridade dos dados e melhorar a experiência do utilizador, no entanto, as regras de validação podem contribuir indiretamente para a segurança da aplicação, ao impedir a submissão de dados inválidos ou maliciosos que poderiam comprometer a integridade ou funcionalidade da aplicação.

Na aplicação *Event Manager*, as regras de validação foram estalecidas ao nível dos dados, definindo restrições nas colunas, como tipo de dados, limites que os valores podem tomar e se a coluna é de preenchimento obrigatório ou não. Ao nível da aplicação, foi aplicada lógica nos controlos, que impede por exemplo a submissão de um formulário se algum critério específico for violado, com a mais-valia de passar uma mensagem de erro informativa ao utilizador. As restrições às colunas podem ser consultadas na Tabela 8 – Descrição da tabela *Event*, Tabela 9 – Descrição da tabela *Client*, Tabela 10 – Descrição da tabela *Task* e Tabela 11 – Descrição da tabela *Organizer*.

Na Figura 25 pode visualizar-se o erro que o sistema envia quando se tenta submeter o campo *Name* do formulário *Event Details* com mais de 16 carateres. Na Figura 26 pode visualizar-se o erro que o sistema envia quando se tenta submeter o campo *Account Name* e/ou *Main Phone* do formulário *Client Details*, vazio.

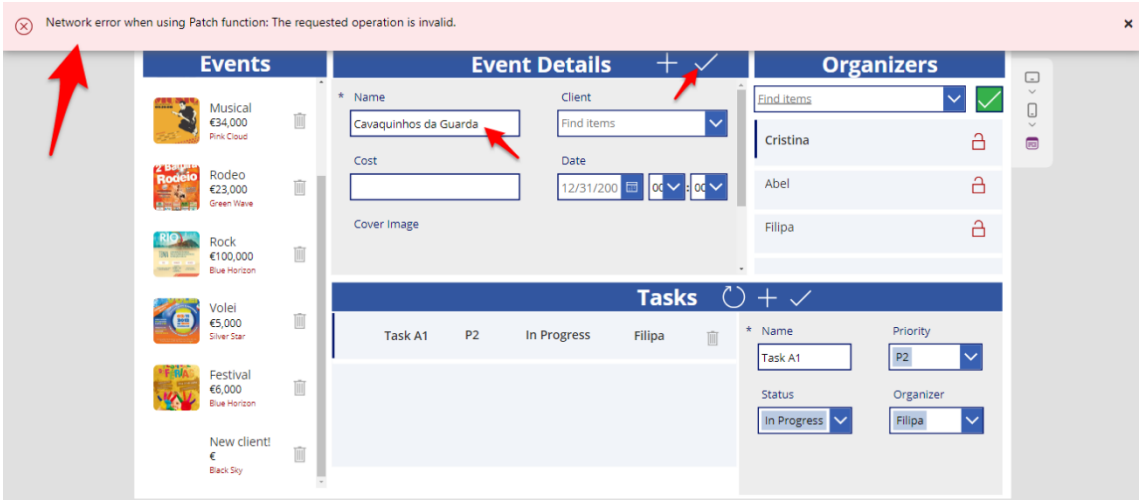


Figura 25 – Erro enviado pelo sistema ao submeter Name com mais de 16 carateres

A Figura 25 representa um *screenshot* da aplicação *Event Manager*, após uma tentativa de submeter um formulário de criação de Evento com nome (*Name*) a exceder 16 caracteres. O sistema identificou a quebra da restrição no campo *Name* da tabela *Event* e responde com uma mensagem de erro.

A Figura 26 representa um *screenshot* da aplicação *Event Manager*, após uma tentativa de submeter um formulário de criação de Cliente onde os campos *Account Name* e *Main Phone* estão vazios. O sistema responde com uma mensagem de erro a informar que os campos são obrigatórios.

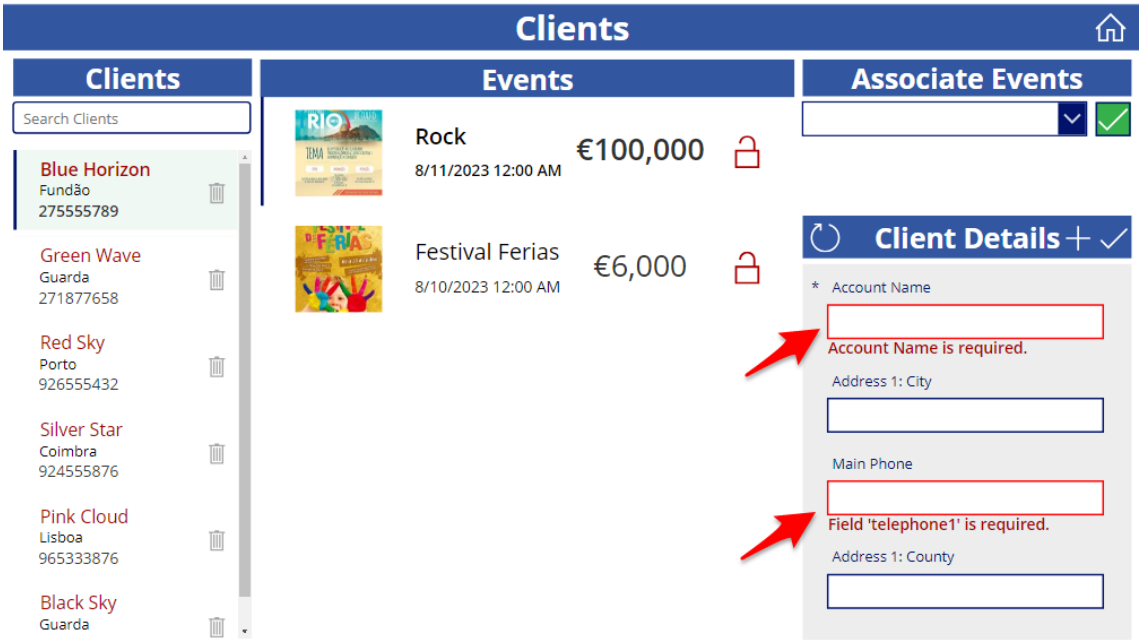


Figura 26 – Erro enviado pelo sistema ao submeter campos obrigatórios vazios

Um exemplo de implementação de uma regra de validação ao nível da aplicação pode ser encontrado na Tabela 16 que apresenta a codificação do controlo *ic_SubmitEvent*. A Tabela 17, descreve as funções usadas na programação desse controlo. A Figura 27 mostra a mensagem de erro exibida no ecrã *Home* de *Event Manager*. Nessa implementação, a aplicação não permite a submissão do formulário *Event Details* se o campo *Name* tiver mais de 15 caracteres, exibindo a mensagem de erro " Field must be less than 15 characters!".

Tabela 16 – Programação do controlo ic_SubmitEvent

Controlo	Propriedade	FX
ic_SubmitEvent	*OnSelect* =	Set (nameLength, Len(formEventDetails.Updates.Name)); If (nameLength > 15, UpdateContext({ ErrorMessage: "Field must be less than 15 characters!" }), UpdateContext ({ErrorMessage: ""}), SubmitForm(formEventDetails))
lbl_ErrorMessage	*Text* =	ErrorMessage

A Tabela 16 contém o código que descreve a lógica de validação do campo "*Name*" presente no formulário *formEventDetails*. Primeiro, é calculado o comprimento do conteúdo inserido no campo "*Name*" e armazenado na variável *nameLength*. Em seguida, é feita uma verificação condicional: se o comprimento do conteúdo for maior que 15 caracteres, uma mensagem de erro "Field must be less than 15 characters!" é exibida, se não, a mensagem de erro é limpa e o formulário *formEventDetails* é submetido para processamento.

Tabela 17 – Descrição das funções usadas na programação do controlo ic_SubmitEvent

Função	Descrição
Set	Atribui valor a uma variável.
UpdateContext	Atualiza os valores das variáveis de contexto dentro de uma aplicação. As variáveis de contexto são específicas do ecrã atual ou do contexto na aplicação.
SubmitForm (formEventDetails);	Recolhe todos os valores preenchidos nos campos do formulário *formEventDetails* e envia-os para o destino designado, neste caso para a tabela *Event*.

A Tabela 17 fornece uma explicação sucinta das funções utilizadas no código do controlo *ic_SubmitEvent*. Essas funções trabalham em conjunto para garantir que o formulário só seja submetido se o campo "*Name*" atender aos requisitos do comprimento do conteúdo inserido no campo. Caso contrário, uma mensagem de erro é exibida ao utilizador.

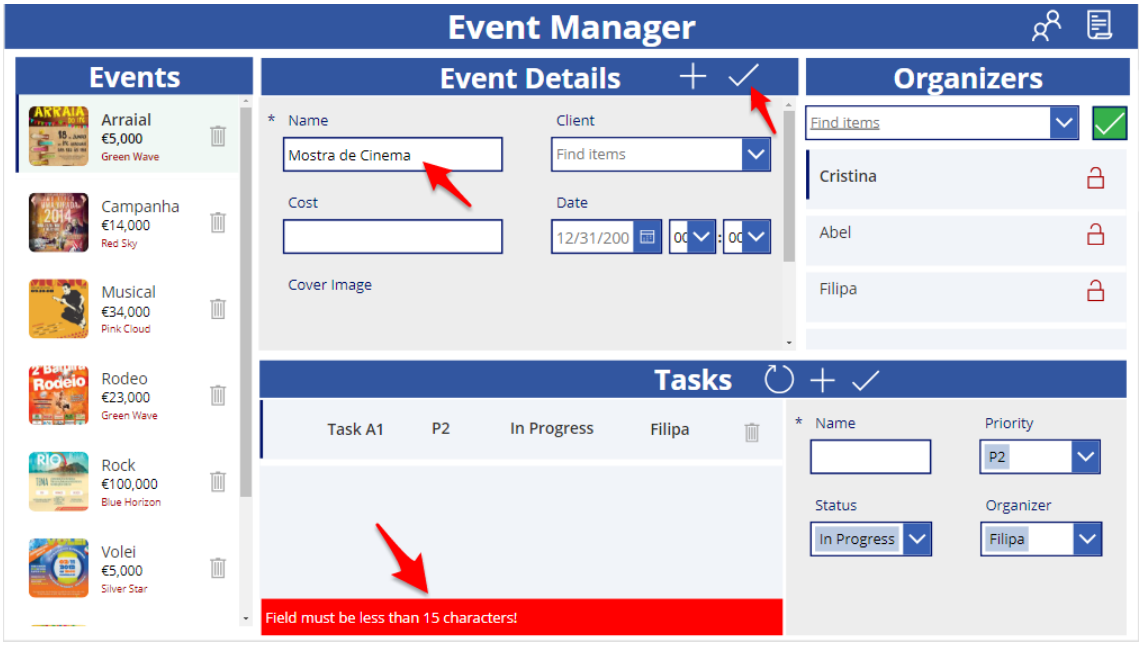


Figura 27 – Exibição de erro na tentativa de submissão de Evento com nome a exceder 15 carateres

Na Figura 27, é apresentada a exibição de erro durante uma tentativa de submissão de um evento com um nome que excede 15 caracteres. A aplicação valida o campo "Name" e ao detetar que o número de caracteres é superior ao limite permitido, exibe uma mensagem de erro para o utilizador.

5.6.2 Criação de Business Units e Teams

A *Power Platform* utiliza unidades de negócio (*Business Units*) e equipas (*Teams*) para proporcionar uma abordagem estruturada e organizada na gestão de utilizadores (*Users*), dados e processos dentro de uma organização.

As unidades de negócio são utilizadas para agrupar utilizadores, dados e processos com base em departamentos específicos, áreas funcionais ou localizações geográficas. Elas fornecem uma forma de organizar e gerir recursos dentro da organização, permitindo uma gestão e controlo mais eficientes.

As equipas, por outro lado, são grupos de utilizadores dentro de uma unidade de negócio que trabalham em conjunto em projetos ou tarefas específicas. Elas permitem a colaboração e

coordenação entre os membros, garantindo que indivíduos com papéis ou responsabilidades semelhantes possam colaborar facilmente em conjunto.

Ao utilizar unidades de negócio e equipas, a *Power Platform* permite que as organizações otimizem as suas operações, melhorem a comunicação e a colaboração e atribuam acessos e permissões adequados aos utilizadores com base nos seus papéis e responsabilidades. Essa estrutura hierárquica ajuda a criar um ambiente mais organizado e eficiente para a gestão de dados e processos.

Business units, *Security Roles* e *Users* estão ligados de uma maneira que se conforma ao modelo de segurança baseado em *roles*. *Business units* juntamente com *security roles* são uma forma eficiente de controlar o acesso aos dados, de tal forma que as pessoas tenham apenas acesso às informações necessárias para desempenhar suas funções [27].

Ao dividir os utilizadores em *Business units* e *Teams*, é possível controlar o acesso aos dados e funcionalidades da aplicação de maneira mais granular. Isso permite que os administradores do sistema concedam permissões apenas às pessoas que precisam delas para desempenhar suas funções, reduzindo o risco de acesso não autorizado ou uso indevido de informações confidenciais. Além disso, as *Business units* e *Teams* também podem ser usadas para monitorizar o uso da aplicação e identificar atividades suspeitas, ajudando a manter a segurança dos dados e do sistema.

No ambiente *Developer Environment* foram implementadas uma *Business Unit* designada *Services* e a *Services* foi atribuída uma *Team* que designámos de *Events.Organizers*. A *team* é constituída por quatro membros e cada um deles tem acumulados um dos três *security roles* criados (ver Figura 28). Como explicado na secção *2.5.2 Security Roles* são conjuntos de permissões que determinam o acesso e as ações que os utilizadores podem realizar dentro de uma aplicação ou plataforma, com o objetivo de controlar e proteger dados e recursos.

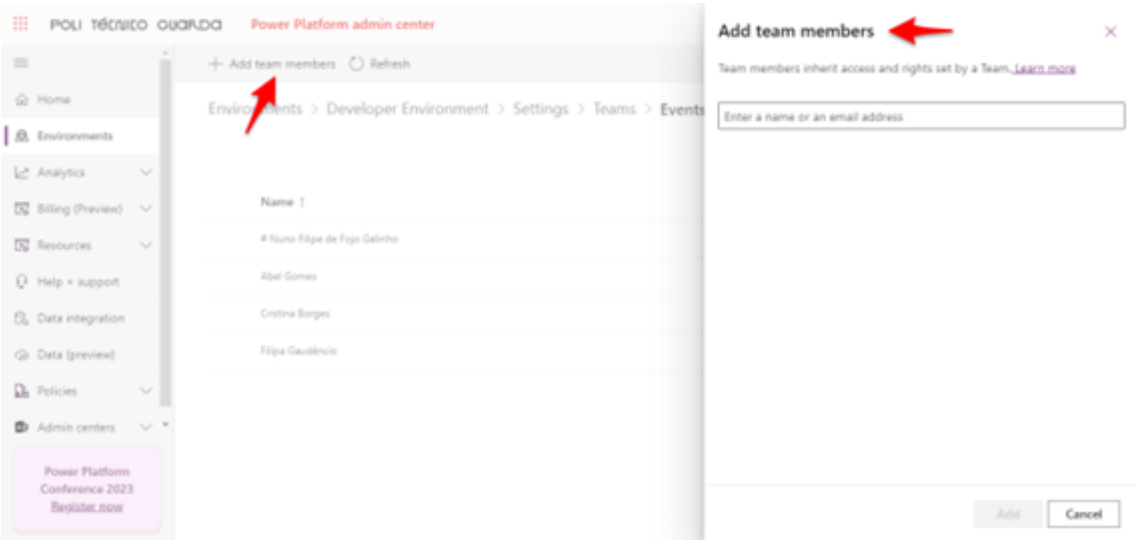


Figura 28 - Membros da equipa Events.Organizers

Na Figura 28, temos representado um *screenshot* do *Power Platform Admin Center*, que exibe a *interface* que permite gerir os membros da equipa "*Events*.*Organizers*". A equipa é constituída por quatro elementos e cada um herda as permissões atribuídas á *Team*.

Á *team* foi-lhe atribuída uma *security role* customizada chamada *Basic Organizer*. A *security role Basic Organizer* tem permissões mínimas (apenas privilégio de leitura) e são comuns a todos os membros da *team*. As restantes permissões necessárias para diferenciar os membros da equipa foram concedidas ao nível do utilizador com as *security roles, System Administrator, Executive Organizer* e *Basic Organizer*.

Como referido na secção 2.5.2 Security Roles *security roles* são cumulativas, que é outra forma de dizer que um utilizador pode ter várias *security roles* e que os seus privilégios são a soma de todas as *security roles* atribuídas a ele. Na Figura 29 pode-se visualizar o esquema hierárquico das business units teams e users criado na instancia de *Developer Environment* onde reside a aplicação *Event Manager*.

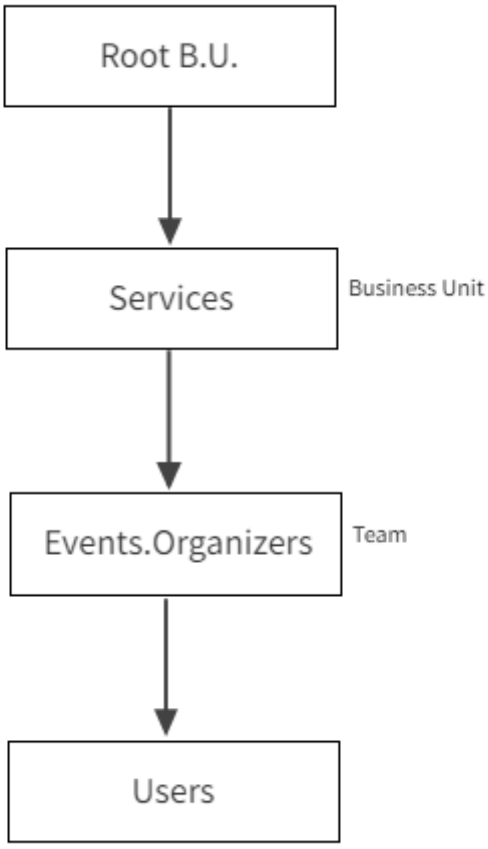


Figura 29 - Diagrama de business units, teams e users criado no Developer Environment

Fazer nota que por falta de permissões, foi impossível apagar ou editar os *users* que aparecem na tabela *User*, pois apenas a organização IPG, que é quem tem os direitos administrativos na plataforma, o pode fazer. Agora, uma vez que usámos essa tabela para representar os organizadores, todos eles aparecem listados na aplicação, no entanto, apenas os adicionados

fazem parte da *business unit Services* e da *team Events.Organizers*, os restantes pertencem por omissão á *root Business Unit* e á *root Team* com *security role Basic User*. Acrescenta-se ainda o facto que apenas os *Users* incluídos no *environment* têm acesso aos recursos do *environment* e apenas os *Users* com quem se partilha a aplicação, têm acesso á aplicação.

5.6.3 Criação de Security Roles

Como referido na secção anterior a atribuição de *security roles* pode ser feita quer ao nível da *team*, quer ao nível do utilizador (*User*), quer mesmo ao nível da *business unit* e são cumulativas.

As *security roles* são atribuídas a partir de uma lista, para tal, se quisermos criar uma ou mais *security roles* personalizadas, para inserir na lista, é boa prática primeiro definir que permissões queremos para cada uma delas. Para tal foi criada uma tabela que explicita quem pode fazer o quê nas tabelas (ver Tabela 18).

Tabela 18 – Security Roles

Atores \ Tabelas	**Event**	**Client**	**Task**	**Organizer**	***Security* Role**
Senior Organizer	CRUD	CRUD	CRUD	CRUD	*System Administrator*
Executive Organizer	R	R	CRUD	R	*Executive Organizer*
Junior Organizer	R	R	R	R	*Basic Organizer*

Nota: apesar do *role System Administrator* nos dar permissões para apagar *Organizers* (*Users*) as restrições administrativas da organização (IPG) não o permitem.

A *security role System Administrator* é uma das *security roles* padrão, predefinida e oferece permissões de criação, leitura, escrita e eliminação ao nível da organização em todas as tabelas acima citadas. As *security roles Executive Organizer* e *Basic Organizer*, foram criadas de raiz.

A criação das *security roles* passa por definir quais os privilégios atribuímos a cada tabela e qual o nível de acesso desse privilégio. Á *security role Executive Organizer*, são atribuídas permissões de leitura ao nível da *Business Unit* às tabelas, *Event, Account (Clients)* e *User (Organizers)* e permissões de criação, leitura, escrita e eliminação ao nível da *Business Unit*, á tabela *Task*. Á *security role Basic Organizer*, são atribuídas apenas permissões de leitura ao nível da *Business Unit*. O tema privilégios e níveis de acesso de uma *security role* podem ser consultados em detalhe na secção 2.5.2.2 Privilégios, Níveis de Acesso e Herança das Security Roles.

Apresenta-se a Figura 30 que ilustra o processo de definição da *security role* "*Executive Organizer*". Nessa imagem, podemos identificar os privilégios e níveis de acesso atribuídos às tabelas *Event* e *Task*. É configurando essas permissões nas tabelas pretendidas que se constrói uma security role, ficando esta depois disponível para atribuição a uma *Business Unit*, *Team* ou *User*.

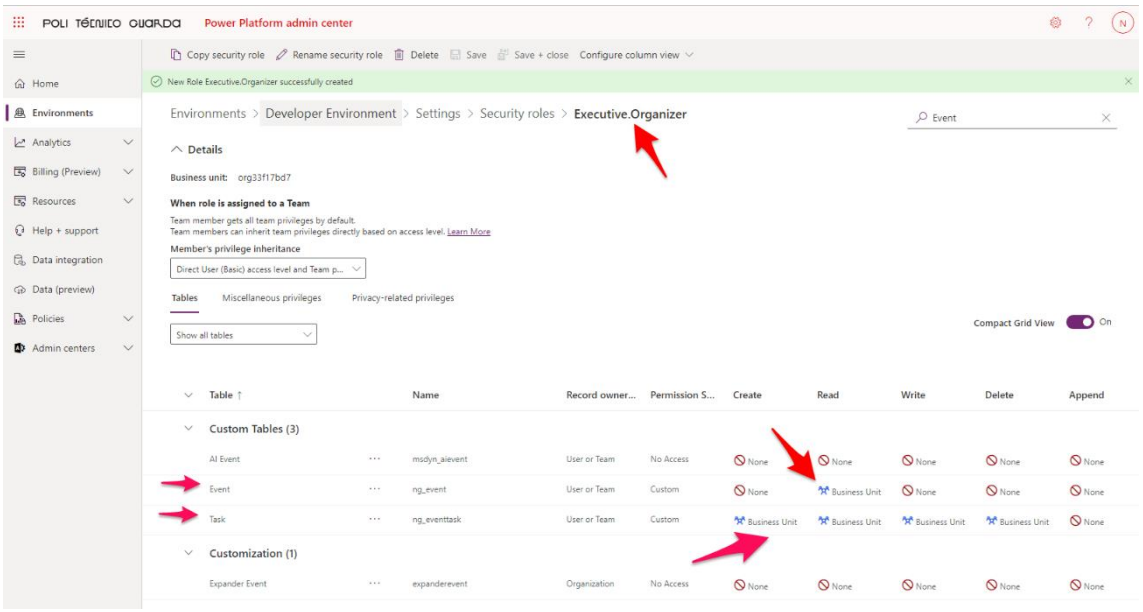


Figura 30 – Security Role Executive.Organizer

A Figura 30 mostra como a *security role Executive Organizer* implementa os privilégios e o nível de acesso deles nas tabelas *Event* e *Task*. Percebe-se que se dá permissão de leitura ao nível da *Business Unit*, á tabela *Event* e permissões de criação, leitura, escrita e eliminação ao nível da *Business Unit*, á tabela *Task*.

Criados as *security roles*, estas ficam disponíveis para serem atribuídas às *Business Units*, *Teams* e *Users*. Á *team Events.Organizers* foi-lhe atribuída a *security role Basic Organizer,* já *aos users* da aplicação foram-lhes atribuídos a *security role* de acordo com a sua função. Assim ao *Senior Organizer* foi-lhe atribuído a *security role System Administrator,* ao *Executive Organizer* foi-lhe atribuído a *security role Executive Organizer* e ao *Junior Organizer* foi-lhe atribuído a *security role Basic Organizer*. Esta estrutura de segurança garante o acesso adequado às informações e funcionalidades da aplicação, de acordo com as responsabilidades e privilégios de cada membro da equipa.

6. Conclusão

O presente relatório teve como objetivo descrever de forma abrangente o trabalho desenvolvido durante o estágio realizado na empresa Noesis Portugal. É gratificante constatar que cada um dos objetivos propostos foi plenamente alcançado.

Durante o período de estágio, foram adquiridos conhecimentos fundamentais sobre o *Dynamics 365*, *Dataverse* e *Power Platform*, tecnologias valiosas no contexto das soluções empresariais. Foram explorados tópicos como grupos de segurança, funções de segurança, unidades de negócio e auditoria. Além disso, foram abordados a criação e gestão de ambientes de trabalho, a implementação de soluções personalizadas, bem como o desenvolvimento e utilização de *plugins* e estudo de módulos no *Dynamics 365*, em particular o módulo de vendas.

Para consolidar e demonstrar esses conhecimentos, foi desenvolvida a aplicação "*Event Manager*" na *Power Platform*, que permitiu exemplificar e aplicar os conceitos aprendidos. A aplicação demonstrou, entre outras coisas, a criação de eventos, registo de clientes, associação de eventos a organizadores e clientes, mais a atribuição de tarefas aos organizadores. Foram definidos diferentes tipos de organizadores, cada um com funções e permissões específicas, o que enriqueceu a compreensão prática dos conceitos subjacentes.

Foram ainda desenvolvidos dois *plugins* na aplicação, os quais possibilitaram a automatização da criação de eventos e tarefas quando se adiciona um cliente, bem como a exclusão de tarefas relacionadas a um evento quando se remove esse evento. Estas implementações demonstraram a capacidade de estender e personalizar as funcionalidades de uma *power app* por meio de *plugins*.

Para futuros projetos, é recomendável o aprofundamento dos conhecimentos em *Dynamics 365*, explorando cenários realistas onde o conhecimento adquirido possa ser aplicado. Essa abordagem permitirá que a organização aproveite ao máximo as poderosas ferramentas do *Dataverse*, *Power Platform* e *Dynamics 365* em soluções empresariais.

É importante ressaltar que este projeto possibilitou a aplicação dos conhecimentos adquiridos ao longo da Licenciatura em Engenharia Informática, bem como a aquisição e aplicação de novos conhecimentos. O estágio curricular foi uma valiosa experiência profissional e pessoal, que contribuiu significativamente para o enriquecimento profissional do finalista. Encerrado o estágio, o finalista recebeu um convite por parte da empresa para a realização de um estágio profissional na Noesis, o que demonstra o reconhecimento do seu trabalho e abre novas oportunidades para o seu desenvolvimento profissional.

Em suma, este relatório cumpre o seu propósito de apresentar de forma clara e detalhada os conhecimentos adquiridos durante o estágio, bem como evidenciar a sua aplicação prática por meio da aplicação desenvolvida. É possível afirmar com confiança, de que as competências adquiridas e a experiência vivenciada, prepararam o finalista para enfrentar desafios futuros no campo das soluções empresariais e contribuir de forma significativa para o sucesso das organizações.

Bibliografia

[1] Microsoft, “Power Automate,” 2023. [Online]. Available: https://powerautomate.microsoft.com/pt-pt/.

[2] Noesis, “About Us: Noesis,” 2023. [Online]. Available: https://www.noesis.pt/.

[3] Microsoft, “What is Power Apps?,” 2023. [Online]. Available: https://learn.microsoft.com/pt-br/power-apps/powerapps-overview.

[4] Microsoft, “What is Microsoft Dataverse?,” 2023. [Online]. Available: https://learn.microsoft.com/en-us/power-apps/maker/data-platform/data-platform-intro.

[5] Microsoft, “Administrar ambientes e recursos do Power Automate,” 2023. [Online]. Available: https://learn.microsoft.com/pt-pt/power-automate/environments-overview-admin.

[6] P. P. Geeks, “Whare Are PowerApps Environment Types?,” 2023. [Online]. Available: https://devoworx.net/what-are-powerapps-environment-types/#whare-are-powerapps-environment-types.

[7] Microsoft, “Solutions overview,” 2023. [Online].

[8] Microsoft, “Security concepts in Microsoft Dataverse,” 2023. [Online]. Available: https://learn.microsoft.com/en-us/power-platform/admin/wp-security-cds#business-units.

[9] Microsoft, “Security roles and privileges,” 2023. [Online]. Available: https://learn.microsoft.com/en-us/power-platform/admin/security-roles-privileges.

[10] Microsoft, “Control user access to environments: security groups and licenses,” 2023. [Online]. Available: https://learn.microsoft.com/en-us/power-platform/admin/control-user-access.

[11] Microsoft, “O que é o Dynamics 365?,” 2023. [Online]. Available: https://dynamics.microsoft.com/pt-br/what-is-dynamics365/.

[12] OpenAI, *Prompt: CRM and Dynamics 365,* 2023.

[13] Microsoft, “Bem-vindo ao Dynamics 365 Sales,” 2023. [Online]. Available: https://learn.microsoft.com/pt-pt/dynamics365/sales/overview.

[14] OpenAI, *Prompt: Plugins in Power Platform,* 2023.

[15] Microsoft, “Event framework,” 2023. [Online]. Available: https://learn.microsoft.com/en-us/power-apps/developer/data-platform/event-framework.

[16] OpenAI, *Prompt: What are assemblies in the context of plugins and Dataverse,* 2023.

[17] I. Sommerville, Software Engineering, Pearson, 2015.

[18] Wikipedia, “Processo de desenvolvimento de software,” 2023. [Online]. Available: https://pt.wikipedia.org/wiki/Processo_de_desenvolvimento_de_software.

[19] InfoEscola, “Info Escola Análise de Requisitos,” 2023. [Online]. Available: https://www.infoescola.com/engenharia-de-software/analise-de-requisitos/. [Acedido em 27 06 2023].

[20] L. M. B. (. Gouveia, “Sistemas de Informação [PDF]. Universidade Fernando Pessoa,” 2023. [Online]. Available: http://homepage.ufp.pt/lmbg/textos/si_texto.pdf.

[21] Creately, “Tutorial do Diagrama de Sequência: Guia completo com exemplos,” 2021. [Online]. Available: https://creately.com/blog/pt/diagrama/tutorial-do-diagrama-de-sequencia/.

[22] OpenAI, *Prompt: "Principais restrições do ambiente Developer",* 2023.

[23] OpenAI, *Prompt: "Environments, Solutions and Publisher",* 2023.

[24] Microsoft, “Gallery control in Power Apps,” 2023. [Online]. Available: https://learn.microsoft.com/en-us/power-apps/maker/canvas-apps/controls/control-gallery.

[25] Microsoft, “Microsoft Power Fx overview,” 2023. [Online]. Available: https://learn.microsoft.com/en-us/power-platform/power-fx/overview.

[26] OpenAI, *Prompt: "Business Units and Teams",* 2023.

[27] Microsoft, “Create or edit business units - Power Platform,” julho 2023. [Online]. Available: https://learn.microsoft.com/en-us/power-platform/admin/create-edit-business-units.

[28] CuboUP, “9 Principais Metodologias de Desenvolvimento de Software.,” 2023. [Online]. Available: https://cuboup.com/conteudo/metodologias-de-desenvolvimento-de-software/.

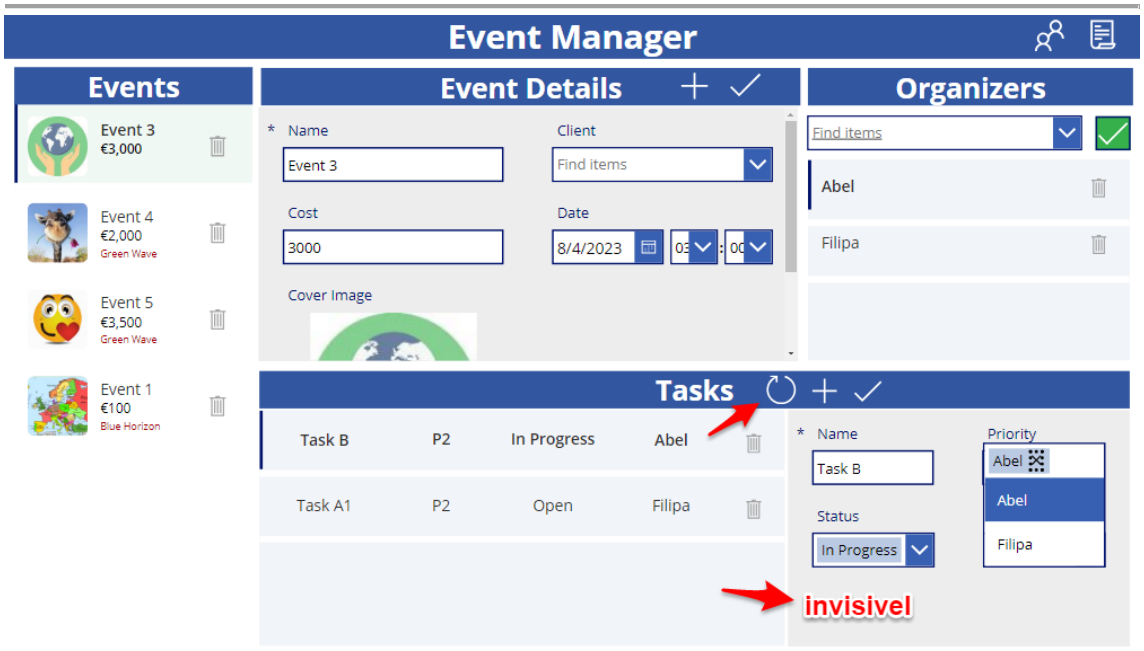
[29] Microsoft, “Power Apps canvas apps documentation,” 2023. [Online]. Available: https://learn.microsoft.com/en-us/power-apps/maker/canvas-apps/.

[30] Microsoft, “What is Microsoft Power Platform,” 2023. [Online]. Available: https://powerplatform.microsoft.com/en-gb/what-is-power-platform/.

Anexos

A 1 - Código e notas pessoais relativo á programação da app Event Manager

SCREEN HOME



FUNÇÕES:

ic_SubmitEvent:
OnSelect = SubmitForm(formEventDetails);
ic_AddEvent:
OnSelect = NewForm(formEventDetails)

ic_AddTask:
OnSelect = NewForm(formTasks)
ic_SubmitTask:
OnSelect = SubmitForm(formTasks)

gallery_Events:
Items = Events
// se item seleccionado, faz cor1, senão branco
TemplateFill = If(ThisItem.IsSelected,RGBA(1, 222, 244, 1),RGBA(0, 0, 0, 0))
Label:
Text = "€" & Text(ThisItem.Cost, "###,###")

ic_deleteEvent:
OnSelect = Remove(Events, ThisItem)

formEventDetails
DataSource = Events
// liga o item do form á galeria
Item = gallery_Events.Selected

formTasks
DataSource = Tasks
// liga o item do form á galeria
Item = gallery_Tasks.Selected
// nota
Error CardStatus! > advanced > unlock > default > ThisItem.'Status (ng_status)'
// original >> novo
CardOrganizer\combo box:
Display\search.fields = ["fullname"] >> ["firstname"]
// antes ia buscar todos os organizers (lookup), agora vai á coleção
OnSelect = false >> ClearCollect(organizersCollection,
gallery_Organizers.AllItems)
Items = Choices([@Tasks].ng_Organizer) >> organizersCollection

gallery_Tasks
Items = gallery_Events.Selected.'Task Statuses'
Label:
Text = ThisItem.Organizer.'First Name'

Events em **gallery_Tasks** é uma coluna lookup que relaciona Tasks com Events.
Em vez de seleccionar o evento na dropDown, mudamos a propriedade **DEfault** do objecto DataCard no U.I. de **ThisItem.Event** para **gallery_Events.Selected** associando assim os
items selecionados na galeria Events com o valor na dropDown. Em seguida mudamos a propriedade visibility do DataCard para falso.

Na relação N - N (Events - Organizers (* Users)
no ic_AddOrganizer , quero associar o Organizer (*User) ao Evento (gallery_Events)

cbx_Organizers
// lista Users
Items = Users
gallery_Organizers
// popula gallery_Organizers com Users relacionados com Eventos
Items = gallery_Events.Selected.Users
ic_RelateOrganizers
// Relate(gallery_Events.Selected.Users, cbx_Organizers.Selected)
OnSelect = ForAll(cbx_Organizers.SelectedItems,
Relate(gallery_Events.Selected.Users, ThisRecord))

ic_UnrelateOrganizer

// NOTA: Não apaga Organizer!

// Remove relação entre o Evento seleccionado (gallery_Events.Selected) e Users (qual User?) - ThisItem

OnSelect = Unrelate(gallery_Events.Selected.Users,ThisItem)

Para garantir que apenas aos organizers relacionados com os eventos possam ser atribuidas Tasks:

formTasks

// antes >> depois

CardOrganizer\combo box:

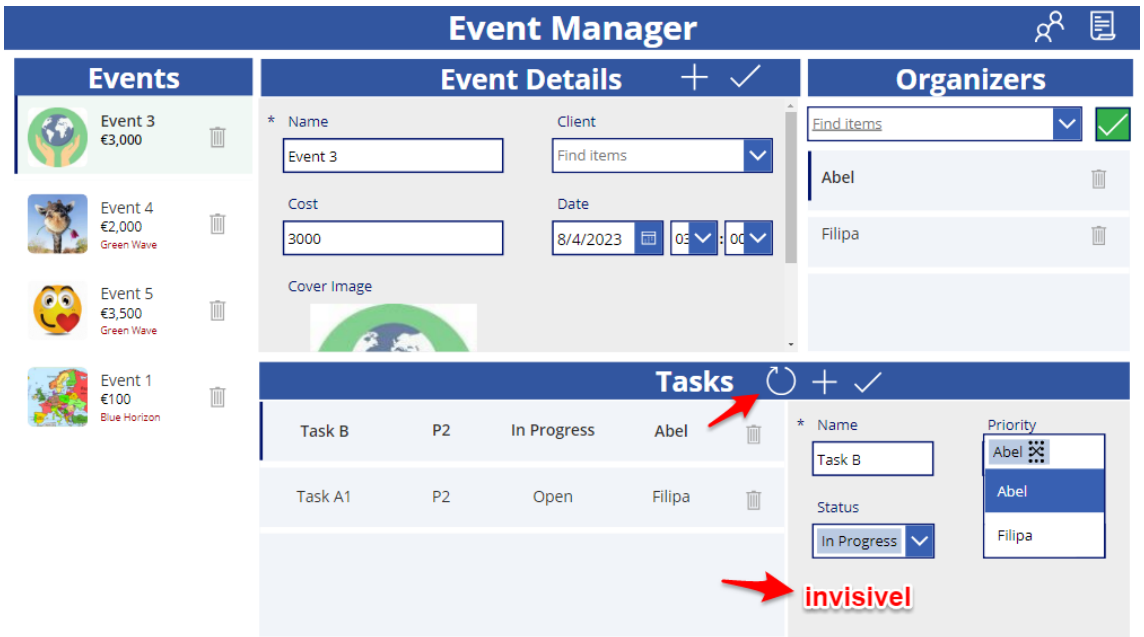
Display\search.fields = ["fullname"] >> ["firstname"]

// antes ia buscar todos os organizers (lookup), agora vai á coleção

OnSelect = false >> ClearCollect(organizersCollection, gallery_Organizers.AllItems)

Items = Choices([@Tasks].ng_Organizer) >> organizersCollection

Uma coleção é um grupo de itens semelhantes, como produtos numa lista de produtos. É usada para armazenar dados que os utilizadores podem gerir na sua aplicação.
A função ClearCollect irá limpar quaisquer dados existentes em organizersCollection e repovoá-la com os registros especificados, gallery_Organizers.AllItems.



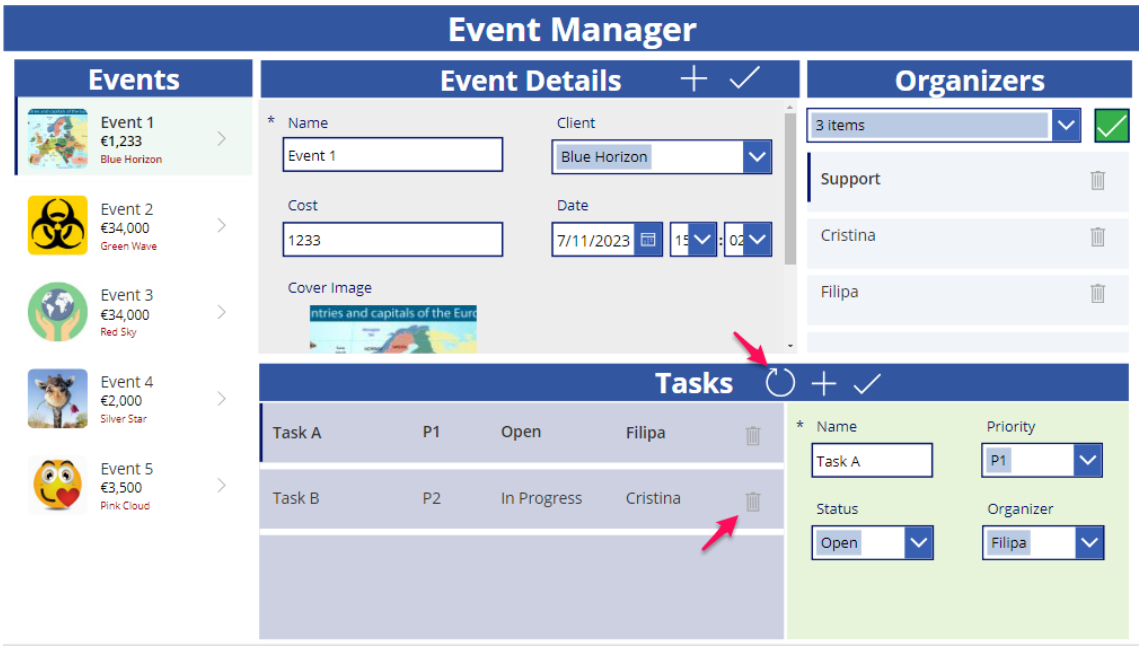
Foi adicionado um delete icon que remove o registo selecionado da Galeria Tasks e um refresh icon para atualizar a Galeria.

ic_RefreshTblTasks

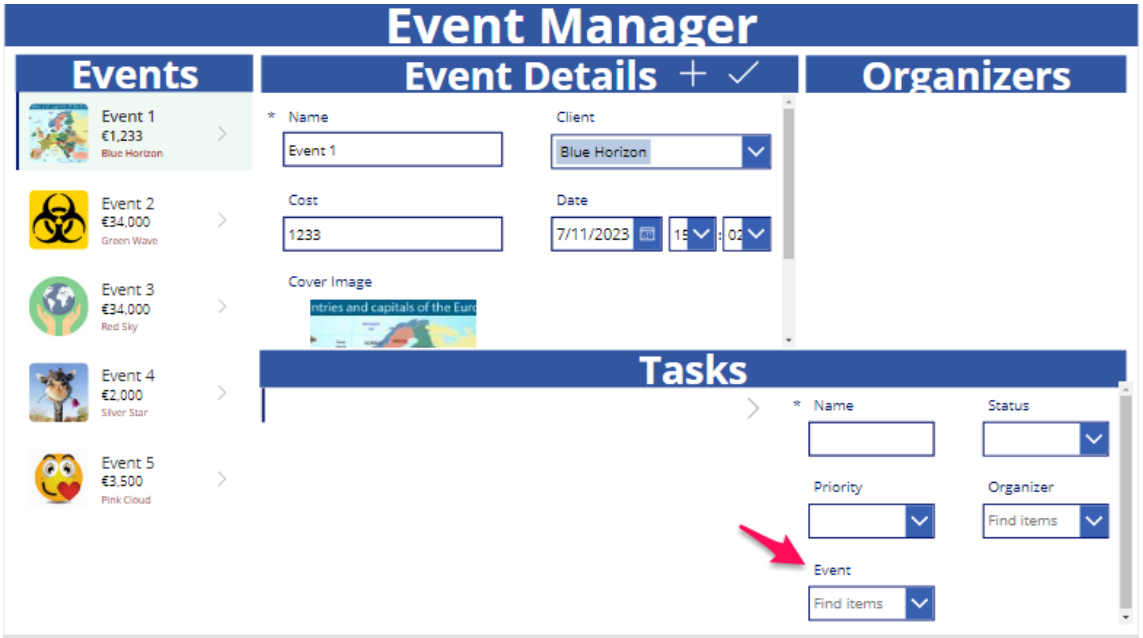
OnSelect = Refresh(Tasks)

ic_DeleteTasks

OnSelect = Remove(Tasks, ThisItem)

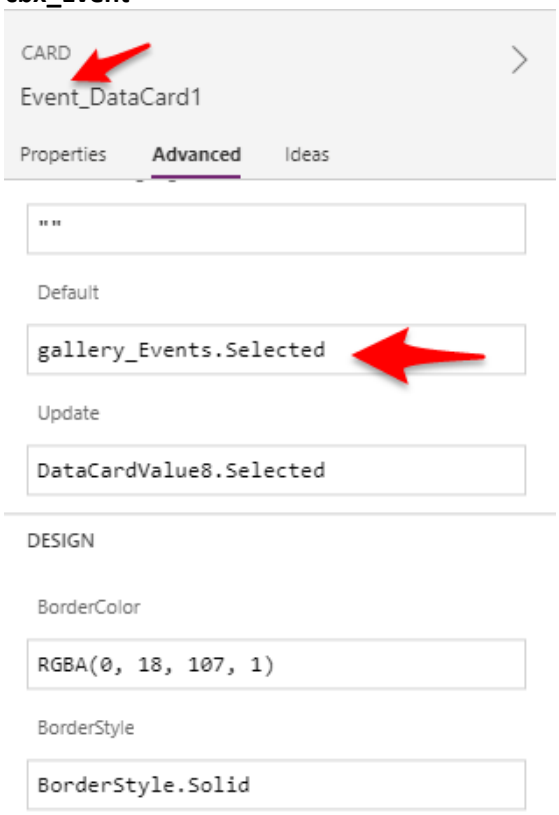


cbxEvents (invisível) em formTasks

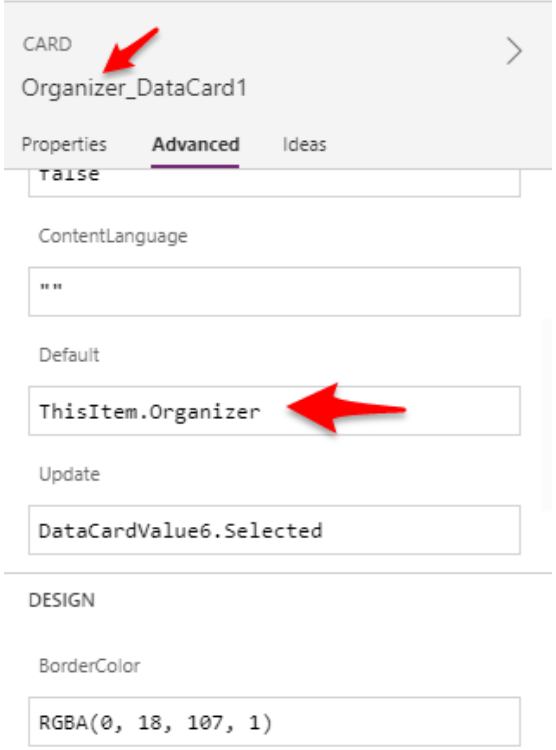


Em vez de seleccionar evento na cbx_Event como fazemos em cbx_Organizer, definimos a propriedade *default* com o valor do item selecionado na gallery_Events e *visible* -> false.

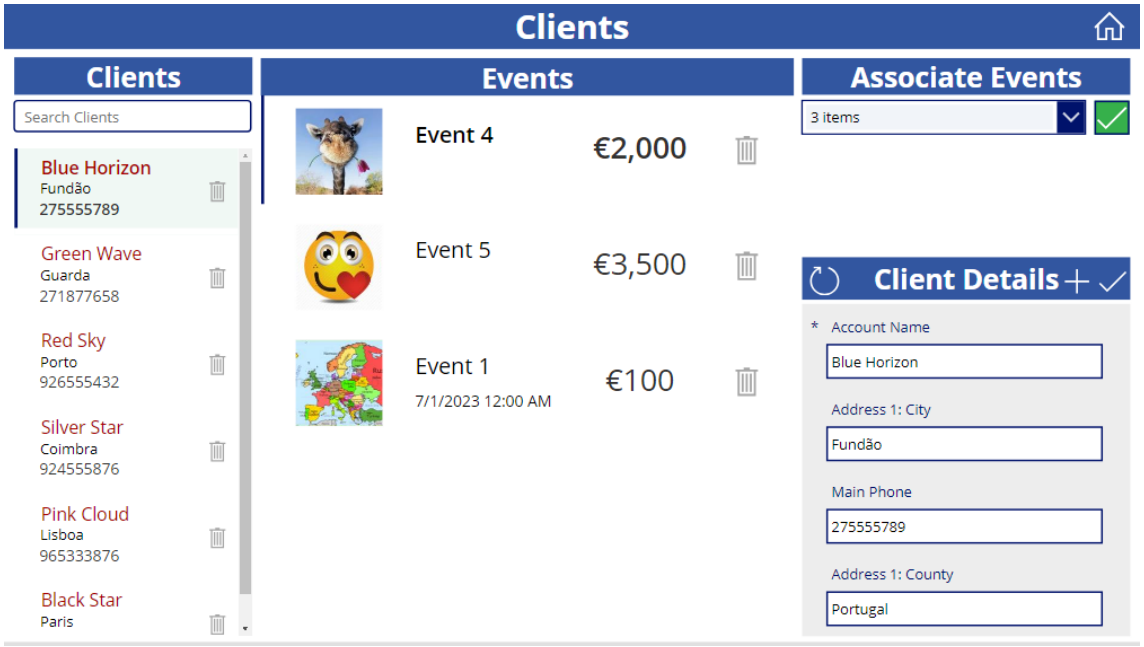
cbx_Event



cbx_Organizer



SCREEN CLIENTS



gallery_Clients
Items = Search(Accounts,textInput_Search.Text, "name", "address1_city")
textInput_Search (sem código)

gallery_Events_2
Items = gallery_Clients.Selected.Events

cbx_Events
Items = Events

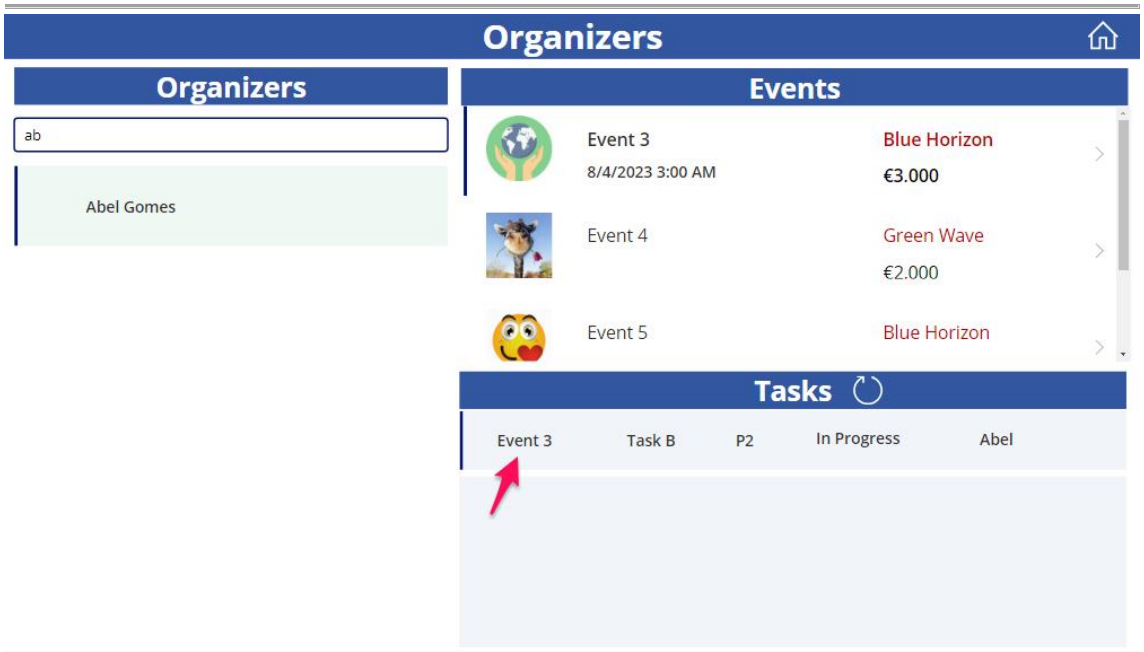
ic_RelateEvents
//Relate(gallery_Clients.Selected.Events,cbx_Events.Selected)
OnSelect = ForAll(cbx_Events.SelectedItems,
Relate(gallery_Clients.Selected.Events,ThisRecord))

ic_UnrelateEvent
//NOTA: Não apaga Evento!
OnSelect: Unrelate(gallery_Clients.Selected.Events,ThisItem)

formClient:
Item = gallery_Clients.Selected
ic_AddClient
OnSelect = NewForm(formClient)
ic_SubmitClient
OnSelect = SubmitForm(formClient)

ic_DeleteClient
OnSelect = Select(Parent); Remove(Accounts,ThisItem)

SCREEN ORGANIZERS

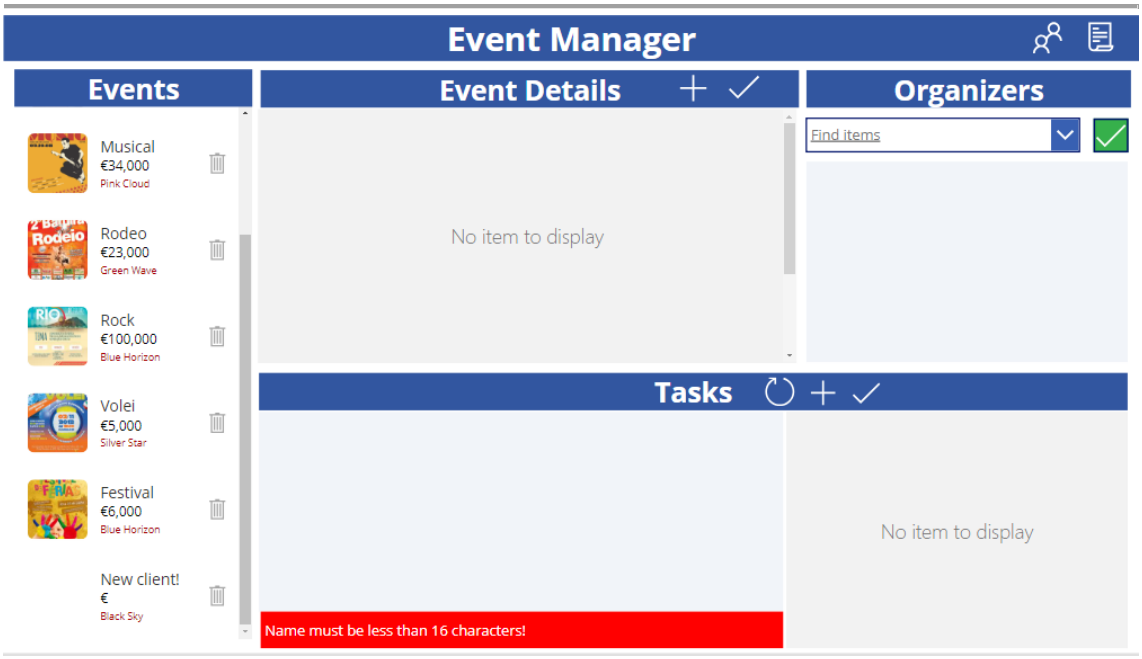


gallery_Organizers_3
Items = Search(Users, textInput_SearchOrganizers.Text, "fullname")

gallery_Events_3
Items = gallery_Organizers_3.Selected.'Events (ng_Event_SystemUser_SystemUser)'

gallery_Tasks_1
Items = gallery_Organizers_3.Selected.'Tasks (ng_ng_eventtask_Organizer_systemuser)'

VALIDAÇÃO DO CAMPO NAME DO FORM_EVENT_DETAILS



ic_SubmitEvent

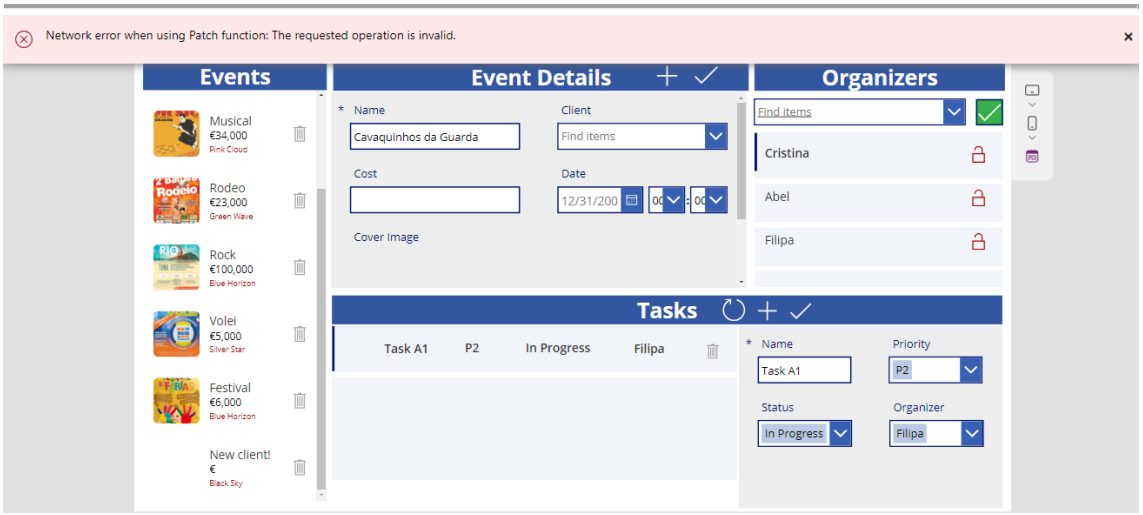
```
OnSelect =
// guarda o valor em nº de carateres, introduzidos no campo Name
Set(
  nameLength,
  Len(formEventDetails.Updates.Name)
);
// se o valor > 16: ErrorMessage, senão ErrorMessage: "" e SubmitForm
If(
  nameLength > 16,
  UpdateContext({ ErrorMessage: "Field must be less than 16 characters!" }),
  UpdateContext({ ErrorMessage: "" }),
  SubmitForm(formEventDetails)
)
```

lbl_ErrorMessage

```
color = Color.White
Fill = If(ErrorMessage <> "", Color.Red, Color.Transparent)
Text = ErrorMessage
```

A função **Set** é utilizada para atribuir um valor a uma variável.
A função **UpdateContext** é utilizada para atualizar os valores das variáveis de contexto dentro de um aplicativo. As variáveis de contexto são específicas do ecrã atual ou do contexto no aplicativo.

MENSAGEM DE ERRO (VALIDAÇÃO DO CAMPO NAME A PARTIR DA RESTRIÇÃO NA TABELA EVENT)



A 2 – Plugins

A 2.1 - DeleteTasksOnEventDeletePlugin

```
using Microsoft.Xrm.Sdk;
using Microsoft.Xrm.Sdk.Messages;
using Microsoft.Xrm.Sdk.Query;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DeleteTasksOnEventDelete
{
    public class DeleteTasksOnEventDeletePlugin : IPlugin
    {
        public void Execute(IServiceProvider serviceProvider)
        {
            // Obter tracing service (não usado!)
            ITracingService tracingService =
(ITracingService)serviceProvider.GetService(typeof(ITracingService));

            // Obter execution context do service provider.
            IPluginExecutionContext context = (IPluginExecutionContext)
                serviceProvider.GetService(typeof(IPluginExecutionContext));

            // Verifica se a entidade de destino é um ng_event
            if (context.InputParameters.Contains("Target") &&
                context.InputParameters["Target"] is EntityReference)
            {
                // Obter a entidade de destino a partir dos parâmetros de
entrada. Uso class EntityReference para obter o entityRef.Id
                // Recebe a referencia do registo de ng_event eliminado
                EntityReference entityRef =
(EntityReference)context.InputParameters["Target"];


                // Verificar se a entidade de destino representa ng_event.
                if (entityRef.LogicalName != "ng_event")
                    return;

                try
                {

                    // Obter a referência do serviço da organização.
                    IOrganizationServiceFactory serviceFactory =
(IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganization
ServiceFactory));
                    IOrganizationService service =
serviceFactory.CreateOrganizationService(context.UserId);

                    // Consulta de todos os registos ng_eventtask associados
ao ng_event que está a ser eliminado
                    QueryExpression query = new
QueryExpression("ng_eventtask");
```

```
                    // só devem ser devolvidos os registos em que o atributo
"ng_event" é igual ao valor de entityRef.Id.
query.Criteria.AddCondition("ng_event", ConditionOperator.Equal,
entityRef.Id);
                    EntityCollection tasks =
service.RetrieveMultiple(query);

                    // throw new InvalidPluginExecutionException("entityRef:
" + entityRef.Id.ToString());

                    // Unrelate e delete de todos os registos ng_eventtask
associados
                    foreach (Entity task in tasks.Entities)
                    {
                        // Unrelate dos registos ng_eventtask associados a
systemuser (Organizers)
                        task.Attributes["ng_organizer"] = null;
                        service.Update(task);

                        // Delete registo ng_eventtask
                        service.Delete("ng_eventtask", task.Id);
                    }
                }
                catch (InvalidPluginExecutionException e)
                {
                    throw new InvalidPluginExecutionException(e.Message);
                    throw new InvalidPluginExecutionException("YOURE OUT!");
                }
            }
        }
    }
}
```

A 2.2 - CreateEventAndTaskOnAccountCreate

```
using Microsoft.Xrm.Sdk;
using Microsoft.Xrm.Sdk.Messages;
using Microsoft.Xrm.Sdk.Query;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


/*
 * Entity - tabela
 * Attribute - coluna
 * Entity e1 - registo da tabela + métodos de Entity
 */
namespace CreateEventAndTaskOnCreatingClient
{
    public class CreateEventAndTaskOnAccountCreate: IPlugin
    {
        public void Execute(IServiceProvider serviceProvider)
        {
            // Obter tracing service
            ITracingService tracingService =
(ITracingService)serviceProvider.GetService(typeof(ITracingService));

            // Obter execution context do service provider.
            IPluginExecutionContext context = (IPluginExecutionContext)
                serviceProvider.GetService(typeof(IPluginExecutionContext));

            // "context.InputParameters["Target"]" referencia a Primary
Entity definida no Plugin Registration Tool
            // A coleção InputParameters contém todos os dados transmitidos
na message request, recebe os dados em runtime
            // "message request" é a operação CRUD
            if (context.InputParameters.Contains("Target") &&
                context.InputParameters["Target"] is Entity)
            {
                // Obter a entidade de destino a partir dos parâmetros de
entrada.
                Entity entity = (Entity)context.InputParameters["Target"];

                // Verificar se a entidade de destino representa uma conta.
                if (entity.LogicalName != "account")
                    return;

                try
                {
                    // Criar uma nova entidade de Evento
                    Entity eventEntity = new Entity("ng_event");
                    eventEntity.Attributes["ng_name"] = "New client!";
                    // lookup account, (1 Event - n Client (account))
                    eventEntity["ng_client"] = new
EntityReference("account", entity.Id);

                    // Criar uma nova entidade Tarefa
                    Entity taskEntity = new Entity("ng_eventtask");
```

```
                    // Obter a referência do serviço da organização.
                    IOrganizationServiceFactory serviceFactory =
(IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganization
ServiceFactory));
                    IOrganizationService service =
serviceFactory.CreateOrganizationService(context.UserId);


                    // Criar o registo Event e obter o seu ID
                    Guid eventId = service.Create(eventEntity);

                    // Atribuir atributos da tarefa: Name, Event
                    taskEntity.Attributes["ng_name"] = "Contact client!";
                    taskEntity["ng_event"] = new EntityReference("ng_event",
eventId);

                    // Consulta do registo de Utilizador com o Email
Principal = 1008043@sal.ipg.pt
                    QueryExpression query = new
QueryExpression("systemuser");
                    query.ColumnSet.AddColumns("internalemailaddress");
                    query.Criteria.AddCondition("internalemailaddress",
ConditionOperator.Equal, "1008043@sal.ipg.pt");
                    EntityCollection users =
service.RetrieveMultiple(query);

                    //
                    if (users.Entities.Count > 0)
                    {
                        // Atribuir atributo da tarefa ao organizador
                        Entity user = users.Entities[0];
                        taskEntity["ng_organizer"] = new
EntityReference("systemuser", user.Id);

                        // Criar uma associação entre n_Event - n_Organiser
ng_Event_SystemUser_SystemUser -> relationship name
                        AssociateRequest associateRequest = new
AssociateRequest()
                        {
                            Target = new EntityReference("ng_event",
eventId),
                            RelatedEntities = new
EntityReferenceCollection() { new EntityReference("systemuser", user.Id) },
                            Relationship = new
Relationship("ng_Event_SystemUser_SystemUser")
                        };

                        // Executar o pedido
                        service.Execute(associateRequest);

                    }

                    // Criar registo de tarefa
                    service.Create(taskEntity);
                }
                catch (InvalidPluginExecutionException e)
                {
                    throw new InvalidPluginExecutionException(e.Message);
                }
            }
```

A 3 – Prompts ao Bing e ChatGPT

01--

. Can a canvas power app remove records from a dataverse table

. Can you give me an example with function Remove, like remove a record from contact table

. After publishing my canvas power app, where and how can I find it

. With wich users can i share my app

. If I share my app with a user that has no permissions to add records to the table contact, but my app has a button that adds those records to that table, can that user add that record by clicking that button.

02--

. Is it common to build powerapps that just use data from dataverse, instead of building apps that make crud operations in dataverse

. What is Power Query

. Is power query a microsoft product directed to act in dataverse

. Can power query make crud operations on dataverse tables

. Can power query make crud operations on dataverse tables

. Where can I find power query to operate on dataverse tables

03--

. Does an edit form from a canvas app has the ability to submit data in the data source table

that is connected with

. Does it need a button to submit the data

. wich function do you use in that sbmit button

04--

. whats the difference between managed solutions and not managed in power apps

05--

. in power apps canvas app when a combo box has a data source table example User,

. wich column of data is going to list, what is the criteria

06--

. in power apps canvas app, what the propertie item represent

>> propertie item: representa o registo. Por exemplo, num Formulário, você pode usar a propriedade Item para especificar qual registo da base de dados deve ser exibido no formulário.

07--

I see a lot of familiar names on the standard User table, are those members of my organization, or not at all related

>> It is possible that the familiar names you see on the standard User table are members of your organization.

08--

>> After publishing a Canvas App, you can access it in several ways. You can run the app on a mobile device by downloading the Power Apps mobile app from the App Store or Google Play Store and signing in with your credentials. You can also run the app in a web browser by going to the Power Apps website and selecting Apps from the left pane. From there, you can select the app you want to run. Additionally, if the app has been shared with you, you may receive an email with a link to the app. You do not have to access make.powerapps.com each time you want to use the app.

. what are standalone apps

>> A standalone app is an application that can be installed and run independently on a device without the need for an additional platform or container app. In the context of Power Apps, a standalone app refers to a Canvas App that has been published as a standalone, native mobile app. This means that the app can be downloaded and installed directly from the Apple or Google app store and run independently on a mobile device without the need for the Power Apps mobile app.

09--

I´ve just finished my canvas app MY_APP, there I use a table User with several users, like A, B, C, now I want to give different roles to wich one, A can do CRUD operations on all tables B can only write on tableX and C can only read, wich steps i have to take to assign these different roles to A, B, C. Is it a good idea to create a business unit

10--

Do different solutions in the same environment share the same dataverse instance

11---

May different Solutions have different users, for example a Solution X has A B C Users and APP_1, Solution Y has A S D Users and APP_2, User A can access both apps, but user B can only access APP_1

12--

The security roles are inherited from top to bottom, never the opposite, a user inherited the security roles from B.U. and team, priviledges are added to the user depending of his access levels

13--

what are team types give me a example of wich one and main differences between them

>> According to web search results, a team can be one of the following types: Owner, Access, Azure AD Security group, or Azure AD Office group [2]. An **Owner** team can own records and have security roles assigned to it. An **Access** team does not own records but can be granted access to records owned by others. An **Azure AD Security group** team is similar to an owner team in that it can own records and have security roles assigned to it. The members of this type of team are dynamically derived based on their Azure AD group membership [2]. An **Azure AD Office group** team is similar to an Azure AD Security group team but corresponds directly to the Azure AD Office group type [2]. Is there anything else you would like to know?

14--

what is the team administrator how does he differenciate himself from others

>> In general, a team administrator may be responsible for managing the team's settings, membership, and security roles

15--

. Is it possible to create a plugin triggered in a canvas app

. Are not plugins triggered when something changes in the Dataverse table

. knowing that if i create a plugin triggered by a change in a table, if a canvas app changes that table would not the plugin be triggered

16--

Pre-operation Plugin, these plugins execute after the main system operation and within the

database transaction. Explain this to me like I was 7 years old, with an example.

17--

. Editing column table in Dataverse what is the difference between Required: Business required and Business recommended

>> Business Required means that the column is mandatory...

. Business Recommended means that it is recommended that the column has a value, but it is still optional,

. has a warning it shows a blue cross.

18--

how do i refresh a gallery in PowerApps canvas app give me a code example for each case.

19--

. what is the recommended class library C# (.NET Framework) for creating a Dataverse plugin 2023

. are there any issues using the 4.7.2 version, should i downgrade it

>> It is recommended to use .NET Framework 4.6.2 when creating a Dataverse plugin

. how do i downgrade to 4.6.2

20--

.What is a control in dataverse and explain what is timeline and how is it created

.To add a timeline control, it must be located on a form within a table. Give me an example

21--

.create new activity records directly from the timeline, give me an example and mention wich fields of wich tables are envolved

.For example, if you are creating an appointment, you would need to fill in fields such as subject,

.start time, end time, location, etc. Give me a more elaborate example, highlite fields and tables

22--

.context.InputParameters.Contains("Target") receives all data of a table?

23--

.give me the sequence of steps of what a plugin does, let say a plugin create preoperation on the entity account

.give me a simple example of a postoperation

.In Task.Attributes["ownerid"] = new EntityReference("systemuser", context.UserId);

.EntityReference refers to the relationship with another Entity

24--

Can i instanciate any entity just by doing var Task = new Entity("logical table name");

25--

review for me the arguments of EntityReference method

26--

eventEntity ("ng_event") is related with User ("systemuser ") in a many to many relationship.

Now i want to ensure that each time i create a new eventEntity ("ng_event") the record

is related to User ("systemuser ")

27--

Remove(Events, ThisItem); from canvas app can trigger a delete message plugin

28--

.Is it possible to copy a canvas app from one solution to another solution

for making sure that the app works properly do i need to mimic the source database in the new solution

.and if i just export the app in to another solution but in the same environment do i still need to replicate

.the schema of the source database in the new solution

29--

.o que é a prototipagem na engenharia de software, é ela um método especifico de metodologia de desenvolvimento de software, pode ela ser considerada parte da metodologia agil

30--

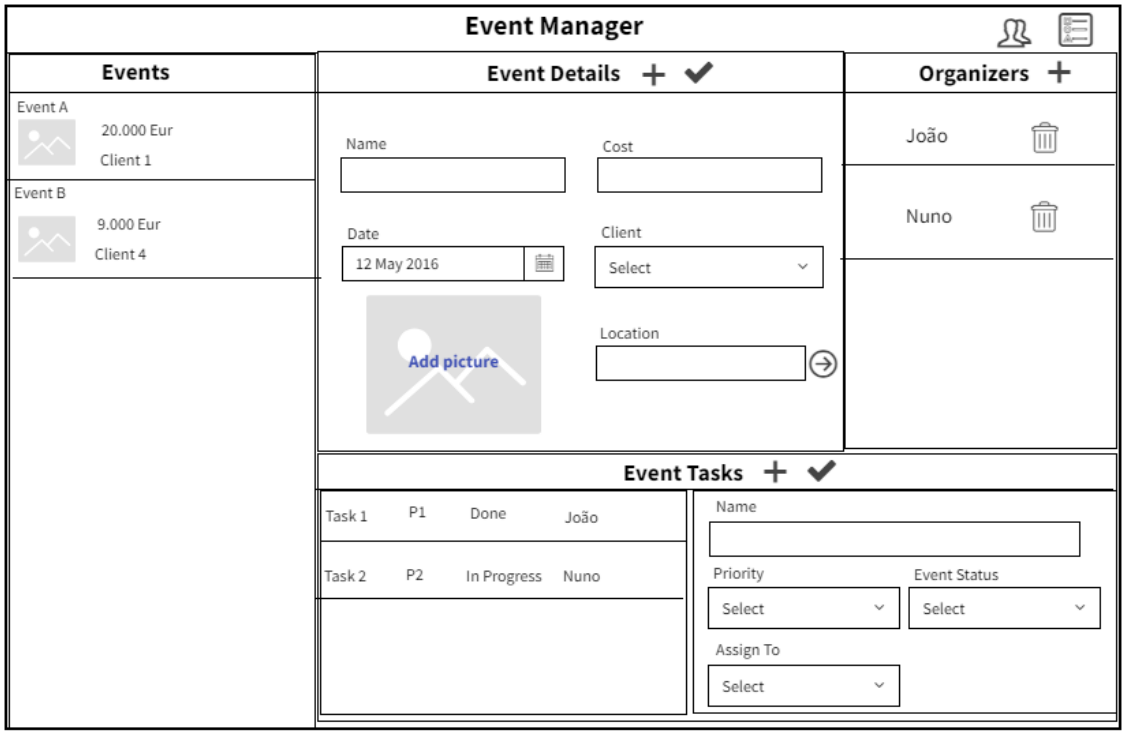
.are you familiar with security groups

.so how can i determine for example user Nuno Galinho of environment "Developer Environment" can see the delete button of Item Events

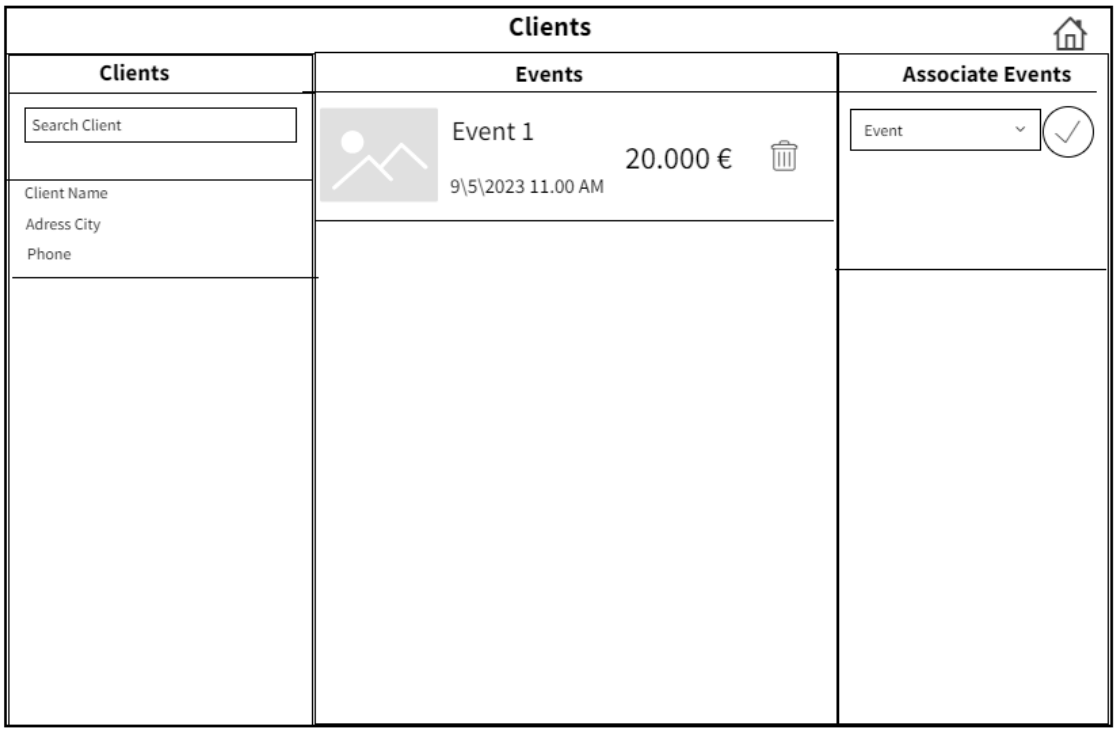
.from a gallery of a canvas app but user Cristina Borges of of same environment can not, is that not defined in the security group

A 4 – Protótipos dos ecrãs: Home, Clientes, Organizadores

Ecrã Home



Ecrã Clientes



Ecrã Organizadores