Governance Azure: Solução Integrada de Monitorização e Controlo de Custos

David Tavares Mendonça[1] and F. Jorge Duarte[1]

ISEP – IPP, Porto
1211572@isep.ipp.pt, fjd@isep.ipp.pt

Resumo A crescente complexidade dos ambientes cloud impõe desafios significativos à governança, resultando frequentemente numa fragmentação entre a monitorização operacional e o controlo financeiro. Embora as ferramentas nativas sejam robustas, a dispersão da informação obriga a uma alternância ineficiente entre interfaces, dificultando a correlação entre métricas de desempenho e o seu impacto orçamental. Este artigo propõe uma solução arquitetural integrada para ambientes Microsoft Azure, desenhada para centralizar a observabilidade de recursos e a gestão de custos. Através de uma implementação de referência fundamentada em *Clean Architecture*, demonstra-se como a centralização numa interface de APIs heterogéneas e a abstração de linguagens de consulta complexas (KQL) permitem otimizar a gestão de custos e recursos no Azure. Os resultados mostram redução do esforço na consulta técnica e financeira, através de visualizações interativas e filtros permitindo a utilizadores com diferentes níveis técnicos obterem *insights* acionáveis.

Keywords: Governança · Gestão de custos · Monitorização de métricas · Microsoft Azure · ASP.NET Core · React · KQL · Azure AD · Clean Architecture · Web API

1 Introdução

A adoção generalizada da computação em nuvem transformou radicalmente o paradigma de gestão de infraestruturas de TI (Tecnologias da Informação). Contudo, esta transição trouxe consigo desafios significativos de governança e controlo orçamental. Estudos recentes indicam que a gestão ineficiente de recursos *cloud* é uma das principais causas de desperdício financeiro em grandes organizações, impulsionando a emergência de práticas de *FinOps* para alinhar operações tecnológicas e responsabilidade financeira [1]. A complexidade inerente aos modelos de faturação dinâmica e à elasticidade dos recursos exige, portanto, estratégias robustas de *Cloud Cost Management and Optimization* (CCMO) que transcendam a simples monitorização técnica [2].

Apesar da disponibilidade de ferramentas nativas no ecossistema **Microsoft Azure**, a informação crítica encontra-se frequentemente fragmentada. Esta dispersão obriga os administradores a uma alternância ineficiente entre interfaces e exige um domínio técnico avançado de linguagens de consulta, como **KQL**, que

é a linguagem de consultas predominante em logs, métricas e telemetria, para correlacionar métricas operacionais com o seu impacto financeiro. Enquanto a literatura académica tem feito avanços nos estudos de *Machine Learning* para a deteção automática de anomalias de custos [3] e controlo baseado em políticas [4], persiste uma lacuna na disponibilização de plataformas unificadas que reduzam a carga cognitiva dos operadores humanos na gestão diária.

Para utilizadores não técnicos, a barreira de entrada para compreender a relação entre o desempenho de um recurso e o seu custo permanece elevada. O presente artigo propõe uma solução arquitetural integrada para ambientes **Azure**, desenhada para centralizar a observabilidade e a gestão de custos. O objetivo é demonstrar a relevância da centralização de métricas e custos numa única interface que oferece visualizações interativas e intuitivas, bem como consultas KQL pré definidas e personalizáveis de forma a acelerar o processo de tomada de decisões operacionais e financeiras e acabar com a dificuldade em gerir depósitos de informação em serviços dispersos.

2 Estado da Arte

O presente capítulo enquadra a solução proposta no panorama atual da *governance* em ambientes *cloud*.

2.1 Estudos académicos

A adoção generalizada de serviços *cloud* impulsionou o desenvolvimento de estratégias de *governance* mais sofisticadas [2].

Neste contexto, o conceito de *CloudOps* evoluiu para incorporar práticas de responsabilidade financeira, culminando na metodologia *FinOps*. Esta abordagem cultural e operacional visa alinhar as equipas de engenharia, finanças e gestão em torno da otimização de custos em larga escala [1]. Modelos teóricos como o CARES (*Cost, Automation, Reliability, Elasticity, Security*) sistematizam estes princípios, promovendo uma gestão equilibrada onde o controlo financeiro não compromete a fiabilidade ou a elasticidade dos serviços [1].

Paralelamente à evolução dos modelos de gestão, a investigação tem explorado a aplicação de técnicas de inteligência artificial como catalisadores na monitorização e otimização financeira [3]. Estudos sobre alocação de recursos evidenciam que a distribuição eficiente de recursos deve considerar simultaneamente restrições de custo e eficiência energética, adotando algoritmos de otimização para ajustar dinamicamente a infraestrutura [5].

Especificamente na deteção de padrões de consumo, os estudos académicos destacam o uso de *Machine Learning* para identificar anomalias de custos e prever variações orçamentais. Abordagens como o *AWS PredSpot* demonstram a eficácia de modelos preditivos na antecipação de flutuações de preços em instâncias *spot* [6]. Contudo, embora estes modelos matemáticos sejam robustos na deteção, persiste a necessidade de interfaces de *governance* que traduzam estas previsões em políticas de controlo acionáveis por operadores humanos [4].

Este desfasamento entre a sofisticação algorítmica e a usabilidade operacional motiva a arquitetura centralizada proposta neste artigo.

2.2 Soluções existentes

A gestão eficaz de ambientes *cloud* exige a correlação contínua entre métricas operacionais e o seu impacto financeiro. No ecossistema Microsoft Azure, esta observabilidade é assegurada por serviços nativos robustos, nomeadamente o **Azure Monitor** para telemetria [7], o **Log Analytics** para consulta de registos [8] e o **Cost Management** para controlo orçamental [9]. Embora estas ferramentas ofereçam granularidade técnica elevada, operam frequentemente em depósitos de informação. A literatura identifica esta fragmentação como um obstáculo crítico, obrigando os gestores a alternar entre múltiplos portais e a dominar linguagens de consulta complexas (como KQL) para obter uma visão holística do estado do sistema [2].

Para mitigar esta dispersão, o mercado evoluiu para a oferta de *Cloud Management Platforms* (CMPs) comerciais, tais como o Turbo360 ou o CloudHealth [10] [11]. Estas soluções agregam dados e centralizam a governança, validando a necessidade de interfaces unificadas [1]. Contudo, a adoção destas ferramentas introduz barreiras significativas: além dos custos de licenciamento, apresentam frequentemente curvas de aprendizagem acentuadas e interfaces densas.

Consequentemente, identifica-se a necessidade de soluções que não só integrem dados, mas que também reduzam a dificuldade da sua análise. A solução *Governance Azure* proposta distingue-se precisamente por priorizar a experiência do utilizador aliada com várias formas de visualizar os dados, com diferentes granulariadades facilitando a extração de *insights* relevantes para a gestão.

2.3 Tecnologias utilizadas

A implementação da solução baseia-se em tecnologias que equilibram robustez, escalabilidade e integração com o ecossistema Azure. O *backend* foi desenvolvido em **ASP.NET Core (.NET 9)**, framework *open-source*, que oferece suporte nativo a injeção de dependências e autenticação com **Azure AD** [12]. O sistema de autenticação utiliza **Azure Active Directory (Entra ID)** para garantir controlo de acessos baseado em papéis (RBAC) e segurança na comunicação entre camadas, sendo preferido neste contexto face a alternativas *cloud*. Foram consideradas alternativas a estas tecnologias, mas não foram adotadas por razões de adequação ao contexto.

Para o *backend* foi avaliado **Spring Boot** (**Java**); contudo, **.NET** oferece integração nativa com **Azure** e com as respetivas *APIs*, o que justifica a sua escolha.

Para a camada de apresentação, adotou-se o modelo de *Single Page Application* (SPA). Embora **Angular** e outras alternativas tenham sido consideradas, **React** revelou-se o mais interessante por ser um *framework* revelou-se mais interessante devido à vasta gama de bibliotecas de visualização de dados disponível [13].

Do ponto de vista de identidade, soluções de *IAM* de outros *cloud providers* (como **AWS** ou **Google Cloud**) foram descartadas por quebrarem a coerência tecnológica e por introduzirem fricção na integração com os serviços **Azure**. Assim, a combinação **ASP.NET Core**, **React** e **Azure AD** alinha-se com as necessidades funcionais e não funcionais do projeto.

3 Análise e Desenho da Solução

3.1 Requisitos funcionais e não funcionais

O processo de análise conduziu à definição de requisitos que orientaram o design funcional e técnico do sistema.

Os principais requisitos **funcionais** identificados foram:

- **RF01:** Permitir a autenticação de um utilizador pertencente ao *tenant* da organização na aplicação através do **Azure AD**.
- **RF02:** Permitir a listagem e pesquisa de recursos **Azure** associados à subscrição do utilizador, de forma a gerir os recursos presentes.
- **RF03:** Permitir a consulta de métricas de desempenho de um recurso específico, para acompanhar o seu estado e atividade.
- **RF04:** Permitir a execução de consultas *KQL* (pré-definidas ou personalizadas) sobre *logs* para analisar métricas operacionais, podendo ajustar a *query* às necessidades.
- **RF05:** Permitir a consulta de custos da subscrição com filtros e agrupamentos dinâmicos, facilitando a compreensão do consumo e das tendências financeiras.

Complementarmente, foram definidos requisitos **não funcionais** que garantem a qualidade da solução e simplicidade de utilização:

- **Usabilidade** foco na simplicidade, clareza e fluidez de utilização, sem dependência de conhecimentos técnicos, design visual moderno e coerente.
- **Fiabilidade**- a aplicação deverá gerir de forma controlada, os erros resultantes da falta de dados.
- **Desempenho** utilização de paginação de resultados, de forma a carregar um número limitado de recursos por página para não comprometer a *performance*.
- **Suportabilidade** documentação robusta que visa facilitar a manutenção e evolução do projeto.

3.2 Arquitetura da Solução

A arquitetura da solução foi desenhada sob os princípios da **Clean Architecture**. Esta arquitetura visa assegurar baixo acoplamento, alta coesão e testabilidade. O *Backend* implementa esta arquitetura distribuída por quatro camadas lógicas fundamentais:

1. Camada de apresentação: Contém os *controllers* e a exposição dos *endpoints* REST, servindo como ponto de entrada para o *Frontend*;
2. Camada de aplicação: Responsável pela orquestração de casos de uso e transformação de modelos de dados através de **DTOs**, dissociando as entidades de domínio dos contratos externos;
3. Camada de domínio: O centro da arquitetura, onde residem as regras de negócio e as interfaces de serviço, isoladas de detalhes de implementação externa;
4. Camada de infraestrutura: Implementa as interfaces definidas no *Core*, concretizando as integrações com os serviços Azure.

Foram aplicados princípios **SOLID** e padrões **GRASP** para reforçar a clareza estrutural e a independência entre componentes. Na Fig. 1 é apresentado o diagrama da *vista lógica de nível 2*, que ilustra a arquitetura da solução e a forma como os principais componentes interagem entre si.

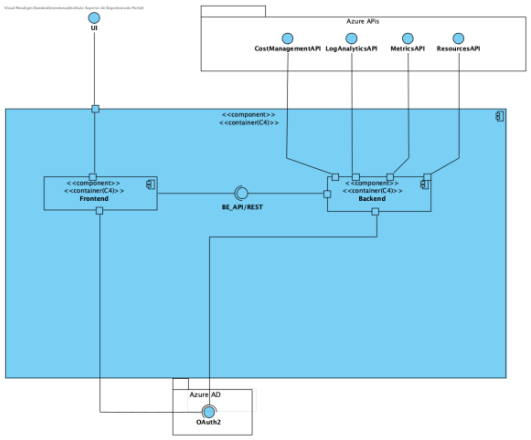

Fig. 1. Diagrama de componentes C4 Vista lógica Nível 2 (UML)

As principais interações dos componentes representados no diagrama acima são: O Frontend (React SPA) autentica-se no Azure AD (OAuth2/MSAL) e, de posse do *token*, invoca a API REST do Backend (ASP.NET Core). O Backend aplica regras de negócio e orquestra chamadas às Azure APIs para ter gestão de recursos e custos, métricas e análise de logs. Serviços transversais (autenticação, *logging*, DI e DTOs) suportam ambos os componentes, garantindo observabilidade e integração consistente.

4 Implementação da Solução

4.1 Segurança: Autenticação e Autorização

A identidade é garantidada através de *OAuth 2.0* com **Azure AD**. No *frontend*, **MSAL.js** obtém e renova *Bearer tokens*, consumido pela WebAPI para invocar as APIs da Azure. No *backend*, o acesso é concedido por políticas baseadas em *roles* do Azure, refletindo diretamente no que o utilizador pode consultar e visualizar.

4.2 Backend e Integrações Azure

A interação com o ecossistema Azure é centralizada na camada de infraestrutura, onde serviços dedicados encapsulam a complexidade da comunicação HTTP e a gestão de autenticação. A arquitetura assegura a cobertura das funcionalidades forneceidas pelas APIs do Azure escolhidas.

A Azure Resource Manager API expõe e gere os recursos; a Azure Monitor API inspeciona-os e produz métricas que suportam alertas operacionais; e a Log Analytics API consulta *workspaces* para obter entradas de log, também ligadas aos mesmos recursos. Em paralelo, a Cost Management API explora custos e origina *cost entries* associados a subscrições, grupos ou etiquetas, permitindo análises por centro de custo.

A Fig. 2 ilustra a implementação deste serviço.

```
    public async Task<string> GetAvailableMetricsAsync(string resourceId)
    {
        var token = await
_authService.GetAccessTokenAsync("https://management.azure.com/.default");
        _httpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", token);

        var url =
$"https://management.azure.com{resourceId}/providers/microsoft.insights/metricDefinitions?
api-version=2018-01-01";

        var response = await _httpClient.GetAsync(url);
        if (!response.IsSuccessStatusCode)
        {
            var error = await response.Content.ReadAsStringAsync();
            throw new Exception($"Erro ao obter métricas disponíveis:
{response.StatusCode} - {error}");
        }

        return await response.Content.ReadAsStringAsync();
    }
```

Fig. 2. Invocação da API do Azure Monitor.

4.3 Frontend e Visualização

O **Frontend** em **React** organiza a navegação por domínios funcionais (recursos, métricas, custos e *logs*) com acesso controlado por rotas protegidas. As interfaces foram desenhadas para facilitar a exploração rápida de dados através de tabelas com pesquisa e gráficos interativos.

A análise financeira é suportada pelo painel de **Sumário de Custos**, onde o utilizador pode aplicar filtros por intervalo temporal, granularidade e agrupamento. Os resultados são renderizados visualmente, permitindo comparar períodos ou observar a distribuição de gastos por categoria através de gráficos circulares, conforme exemplificado na Fig. 3.

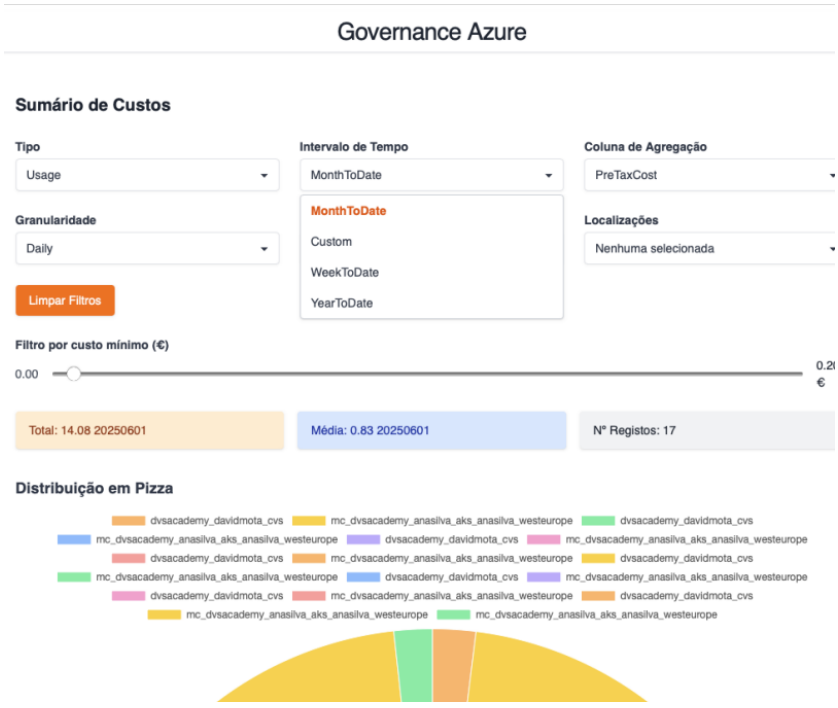

Fig. 3. Visualização interativa da distribuição de custos por recurso.

Para a monitorização operacional, a interface de **Inspeção de Métricas** permite selecionar o recurso, a métrica e a função de agregação desejada. O sistema projeta um gráfico de linhas para análise de tendências, complementado por *KPIs* automáticos (média, máximo e mínimo) e funcionalidade de exportação.

Por fim, a complexidade da consulta de registos é gerida num editor dedicado (Fig. 4), que permite a execução de *queries* KQL e a personalização da consulta pré definida para ajuste às necessidades do utilizador.

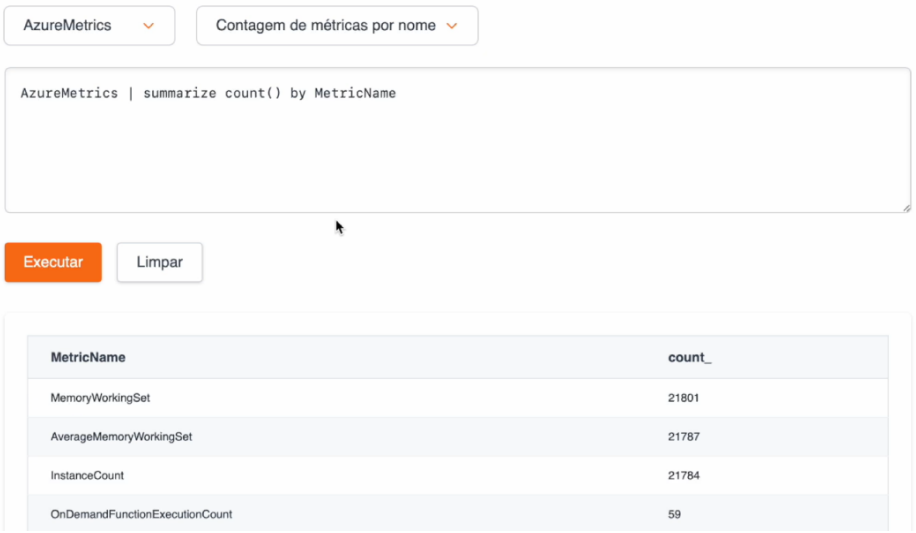

Fig. 4. Interface da camada de abstração de consultas KQL

4.4 Testes da aplicação

Os *endpoints* foram testados com **Postman** e cobertos por testes unitários focados na lógica de serviços. Os restantes cenários foram validados pela UI de forma a verificar se os resultados eram os esperados, assegurando uma experiência consistente sobre dados operacionais e financeiros provenientes das APIs Azure. A robustez da solução foi garantida através da execução de testes unitários, de integração e *end-to-end* (*E2E*). Os testes unitários verificaram regras de negócio e transformação de dados, enquanto os de integração validaram a comunicação entre camadas e o consumo das APIs Azure. Os testes *E2E* reproduziram fluxos reais, desde a autenticação até à visualização de métricas e custos, assegurando o correto funcionamento do sistema como um todo. No total, 112 testes unitários atingiram 77% de cobertura do código, de forma a cobrir os principais fluxos críticos e deixando por testar código que não tem lógica de negócio isolável.

O processo de integração contínua, implementado via GitHub Actions, automatizou a execução dos testes e o *build* do projeto, permitindo deteção precoce de problemas.

4.5 Avaliação e Resultados da Solução

A validação da solução proposta foi conduzida segundo uma metodologia multidimensional, desenhada para aferir a robustez arquitetural e a eficácia operacional da centralização de dados. A estabilidade do sistema foi corroborada pelos testes descritos na secção anterior.

Do ponto de vista da engenharia, a eficácia da adoção da *Clean Architecture* foi notória. Verificou-se que a utilização consistente de objetos de transferência de dados (DTOs) assegurou um desacoplamento efetivo entre o domínio e a infraestrutura, facilitando a manutenção evolutiva sem regressões. Simultaneamente, os testes de segurança ratificaram a integridade do controlo de acessos (RBAC), demonstrando que o mecanismo de autenticação garante o isolamento estrito de *tenants* e preserva a confidencialidade dos dados financeiros durante todo o ciclo de vida das transações HTTP.

Em cenário de produção, a API evidenciou tempos de resposta compatíveis com os requisitos de interatividade, suportando uma navegação fluida e a renderização dinâmica de visualizações complexas. Mais relevante, a centralização de métricas e custos numa interface unificada permite uma grande redução de esforço na correlação entre os indicadores técnicos e impacto orçamental. Assim, os resultados evidenciam a legitimidade e a viabilidade da abordagem proposta.

5 Conclusões

O projeto comprovou a viabilidade de uma solução integrada de governança em **Azure**, focada na monitorização de recursos e no controlo de custos. Os objetivos foram alcançados, incluíram: autenticação via Azure AD; listagem e pesquisa de recursos; consulta de métricas com diferentes agregações; execução de *queries KQL*; e análise de custos com filtros e agrupamentos.

A centralização de métricas, *logs* e custos numa única interface eliminou a necessidade de navegação entre múltiplos serviços, facilitando a leitura conjunta dos indicadores técnicos e do impacto financeiro. A experiência de utilização revelou-se fluida, com visualizações interativas e mecanismos de filtragem intuitivos, mantendo coerência com a identidade da organização.

Em suma, a solução cumpre o propósito definido: simplifica a monitorização e gestão de recursos em Azure, acelera a obtenção de *insights* e apoia a tomada de decisão. Os resultados sustentam a adoção desta abordagem como base sólida para evoluções futuras no contexto de governança em *cloud*.

Identifica-se como limitação funcional a ausência de um sistema automatizado de alertas que notifique os administradores, em tempo real, quando determinados limiares orçamentais são ultrapassados.

Como linhas de investigação futura, perspetiva-se colmatar esta lacuna e integrar modelos de Inteligência Artificial para a deteção automática de anomalias, permitindo evoluir para uma governança pró-ativa e preditiva.

Referências

1. W. A. Al-Sarayreh e M. A. Al-Qudah, "The Role of FinOps in Large-Scale Cloud Cost Optimization," *ResearchGate*, 2024.
2. S. R. Islam, *et al.*, "Cloud Cost Management and Optimization," *ResearchGate*, 2024.
3. J. Smith e A. Doe, "Machine Learning for Intelligent Cloud Cost Anomaly Detection," *ResearchGate*, 2024.
4. K. Patel e R. Singh, "Policy-Based Cloud Cost Control through Intelligent Automation," *ResearchGate*, 2024.
5. S. K. Singh, *et al.*, "Cloud Resource Demand Prediction using Machine Learning in the Context of QoS Parameters," *ResearchGate*, 2021.
6. M. Al-Roomi, *et al.*, "AWS PredSpot: Machine Learning for Predicting the Price of Spot Instances in AWS Cloud," *ResearchGate*, 2022.
7. Microsoft Docs. (2023). *What is Azure Monitor?* Disponível em: https://learn.microsoft.com/en-us/azure/azure-monitor/overview
8. Microsoft Docs. (2023). *What is Log Analytics in Azure?* Disponível em: https://learn.microsoft.com/en-us/azure/azure-monitor/logs/log-analytics-overview
9. Microsoft Docs. (2023). *Azure Cost Management and Billing documentation.* Disponível em: https://learn.microsoft.com/en-us/azure/cost-management-billing/
10. Turbo360. (2025). *Azure Cost Management Tool to Analyze & Optimize Cost Spend.* Disponível em: https://turbo360.com/azure-cost-analysis
11. IBM Cloudability. (2025). *Cloud Cost Management & Optimization Solutions.* Disponível em: https://www.apptio.com/products/cloudability/
12. Daily.dev. (2024). *Backend Frameworks List Choosing the Right One.* Disponível em: https://daily.dev/blog/backend-frameworks-list-choosing-the-right-one
13. Space-O Technologies. (2024). *Top Frontend Frameworks Comparison: Which One to Choose in 2024?* Disponível em: https://www.spaceotechnologies.com/blog/front-end-frameworks-comparison/