**MIDDLEWARE PARA INTEGRAÇÃO DE SISTEMAS WEB COM SERVIÇOS EM NUVEM**

FLOR, Caio Eduardo Vieira[[1]](#footnote-0)

SANTOS, Lucas Soares dos[[2]](#footnote-1)

NEGRETTO, Diego Henrique[[3]](#footnote-2)

Centro Universitário Hermínio Ometto – FHO, Araras – SP, Brasil

**Resumo**

*Com o constante desenvolvimento da tecnologia, torna-se necessário o uso de recursos que possam auxiliar os softwares, melhorando seu desempenho, escalabilidade e usabilidade. Entre os recursos cada vez mais presentes e essenciais nesse cenário estão os serviços em nuvem, que permitem o armazenamento de dados em um ambiente virtualizado. Para integrar um sistema em nuvem, enfrentamos um grande desafio devido à diversidade de APIs e métodos de autenticação disponíveis no mercado.  
Neste projeto, será desenvolvido um Middleware para simplificar o processo de integração com múltiplos serviços em nuvem, fornecendo uma interface unificada para os desenvolvedores. A solução será implementada em PHP, utilizando Composer para o gerenciamento de dependências e aplicando padrões de projeto como Facade, Strategy e Factory. A arquitetura do projeto é baseada em Service Layer e Domain Model, promovendo modularidade e reutilização.  
Como resultado, espera-se reduzir o esforço necessário para integrações com serviços como AWS S3, FTP, SFTP e futuramente Google Drive, além de facilitar a manutenção e extensão de sistemas web que dependem dessas tecnologias.*

*Palavras chave: Biblioteca, Computação em Nuvem, Incorporação de Sistemas, Conectividade digital, Plataformas online.*

**1 Introdução**

**1.1 Contextualização**

Com o avanço constante da transformação digital, as organizações têm adotado cada vez mais os serviços em nuvem como base para suas operações, buscando maior escalabilidade**,** disponibilidade global e redução de custos operacionais. Provedores como *Amazon Web Services* (AWS), *Google Cloud Platform* (GCP) e *Microsoft Azure* tornaram-se líderes no mercado, oferecendo soluções robustas que permitem a execução de aplicações em escala global (ARAUJO, 2021).  
No entanto, à medida que esses serviços se tornam mais populares, os desenvolvedores enfrentam o desafio da integração: cada provedor utiliza APIs, modelos de autenticação, formatos de requisição e estruturas de diretórios distintos. Essa variedade torna a integração direta com múltiplos serviços um processo complexo, propenso a erros e com alto custo de manutenção.  
Em um cenário onde empresas precisam adaptar seus sistemas a diferentes soluções de armazenamento e computação em nuvem, muitas vezes exigidas por seus próprios clientes, surge a necessidade de ferramentas que abstraiam essa complexidade, facilitando a interoperabilidade entre plataformas.  
Nesse contexto, o uso de bibliotecas de *software* reutilizáveis representa uma solução estratégica. Bibliotecas são conjuntos de códigos previamente desenvolvidos e testados, que encapsulam funcionalidades comuns e oferecem interfaces padronizadas para uso em múltiplos projetos. Segundo Barreto (2007), a reutilização de componentes promove o aumento da produtividade, melhora a qualidade dos processos e facilita a definição de soluções por meio da composição modular de partes reutilizáveis. Ainda segundo o autor, essas técnicas, tradicionalmente aplicadas no desenvolvimento de *software*, também podem ser adaptadas à definição de processos, permitindo maior flexibilidade, reaproveitamento de conhecimento e redução de esforço no desenvolvimento.   
Estudos também indicam que bibliotecas bem projetadas podem acelerar o desenvolvimento, facilitar a manutenção e reduzir a densidade de erros (MALDONADO, 2002; Desai et al., 2021).  
Este projeto se propõe a desenvolver um *Middleware* com o objetivo de padronizar a integração com múltiplos serviços em nuvem, fornecendo uma única interface de acesso para operações como *upload*, *download* e gerenciamento de arquivos. A proposta é reduzir a complexidade técnica e promover maior eficiência no desenvolvimento de aplicações *web*, independentemente do provedor em nuvem utilizado.

**1.2 Tema de Pesquisa**

Este trabalho tem como tema a criação de um *Middleware*, desenvolvido na forma de uma biblioteca, com o objetivo de facilitar a integração de sistemas *web* com diferentes serviços em nuvem.

**1.3 Motivações e Justificativas**

A crescente demanda por soluções personalizadas, escaláveis e interoperáveis tem impulsionado o uso de serviços em nuvem nas mais diversas aplicações *web*. No entanto, essa diversidade de plataformas e modelos de integração impõe desafios significativos aos desenvolvedores, que frequentemente precisam implementar soluções específicas para cada provedor.  
Diante desse cenário, torna-se necessário o desenvolvimento de ferramentas que simplifiquem e unifiquem a comunicação entre sistemas *web* e diferentes serviços em nuvem. A biblioteca proposta neste projeto busca atender a essa necessidade, oferecendo uma interface única e reutilizável, capaz de abstrair as particularidades de cada serviço.  
Com isso, pretende-se reduzir o esforço de desenvolvimento, os custos de manutenção e o tempo necessário para implantação de novas funcionalidades, além de promover maior padronização e praticidade na evolução das aplicações.

**1.4 Objetivos**

O objetivo geral deste trabalho é desenvolver um *Middleware* em PHP, implementado como uma biblioteca reutilizável, que facilite a integração de sistemas *web* com diferentes serviços em nuvem.

**1.4.1 Objetivos específicos**

* Investigar o conceito e os usos de bibliotecas no contexto da programação;
* Identificar quais serviços em nuvem poderão ser integrados à biblioteca;
* Realizar o levantamento de requisitos da biblioteca;
* Desenvolver estratégias específicas para comunicação com cada serviço em nuvem;
* Implementar uma **fachada (*Facade*)** que abstraia as operações internas da biblioteca;
* Publicar a biblioteca no gerenciador de dependências ***Composer***, da linguagem PHP;
* Desenvolver um sistema *web* que integre e demonstre a aplicação prática da biblioteca.

**2 Revisão Bibliográfica**

A presente sessão propõe-se a abordar de maneira abrangente os materiais, tecnologias e definições pertinentes à consecução do projeto em análise.

**2.1 Fundamentação Teórica e Técnica**

**2.1.1 Biblioteca de software**

Bibliotecas de *software* são conjuntos de códigos previamente desenvolvidos que encapsulam funcionalidades comuns e podem ser reutilizados em diferentes projetos. Seu uso permite reduzir o tempo de desenvolvimento, aumentar a padronização, diminuir erros e facilitar a manutenção de sistemas (MALDONADO, 2002).

**2.1.2 Computação em Nuvem**

A computação em nuvem é um modelo de fornecimento de recursos computacionais - como armazenamento, processamento e rede - de forma sob demanda, com cobrança baseada no uso, escalabilidade automática e alta disponibilidade. Segundo Carvalho (2020), esse modelo oferece uma economia significativa e maior flexibilidade para empresas de todos os portes.

Cada serviço possui sua própria API, métodos de autenticação e estrutura de organização de dados, o que dificulta a integração com sistemas que precisam ser compatíveis com múltiplos provedores.

O estudo de Desai et al. (2021) reforça esse desafio, ao apontar que a diversidade de interfaces e requisitos técnicos entre os provedores em nuvem podem dificultar a automação e manutenção de sistemas integrados.

**2.1.3 *Middleware* e Abstração de Integrações**

*Middleware* é um tipo de *software* intermediário que atua como uma camada de comunicação entre sistemas distintos, permitindo que aplicações interajam mesmo utilizando tecnologias, protocolos e arquiteturas diferentes. Ele encapsula a lógica necessária para conectar serviços e componentes, oferecendo uma interface unificada e padronizada ao desenvolvedor (LEITHARDT, 2015).

O *middleware* também pode promover flexibilidade e adaptabilidade, ao permitir que diferentes ferramentas e serviços sejam integrados dinamicamente conforme as necessidades específicas da aplicação, o que favorece sua reutilização (BARROS, 2011).

No projeto em questão, a biblioteca proposta atua como *middleware* especializado em serviços de armazenamento em nuvem. Seu papel é abstrair a complexidade das APIs de serviços como AWS S3, *Google Drive*, FTP e SFTP, centralizando as integrações em uma única estrutura reutilizável e extensível.

Esse conceito está alinhado com o princípio de abstração da orientação a objetos, no qual detalhes internos são ocultados por trás de interfaces simplificadas. Para isso, o projeto também adota o padrão de projeto *Facade*, permitindo ao desenvolvedor interagir com qualquer serviço em nuvem suportado a partir de uma única interface central, sem a necessidade de compreender ou adaptar-se às especificidades de cada provedor.

**2.1.4 Tecnologias Utilizadas**

**PHP** - O PHP foi a linguagem escolhida para o desenvolvimento da biblioteca e do sistema demonstrativo. Trata-se de uma linguagem de código aberto, amplamente utilizada no desenvolvimento de aplicações *web* e compatível com diversos *frameworks* e bibliotecas de terceiros (CONVERSE, 2003).

**Composer** - O *Composer* é o gerenciador de dependências padrão do PHP. Ele será utilizado para disponibilizar a biblioteca ao público, facilitando sua instalação e integração em projetos diversos (BENTO, 2021).

**JavaScript** -A linguagem escolhida para o *front-end* do sistema é o *JavaScript*, devido à sua capacidade de modificar dinamicamente o conteúdo das páginas *web*, proporcionando uma experiência interativa e fluida ao usuário (PRESCOTT, 2016). O *JavaScript* será utilizado principalmente para implementar funcionalidades como o *upload* de arquivos e a exclusão de arquivos sem a necessidade de recarregar a página.

**Bootstrap** -Para estilizar o sistema *web*, será utilizado o *Bootstrap*, um *framework* *front-end open source* amplamente adotado para a criação de interfaces responsivas e modernas (TWITTER, 2022). O *Bootstrap* facilitará a construção de páginas que se adaptam a diferentes tamanhos de tela, garantindo que o sistema seja acessível em dispositivos móveis e *desktops*.

**APIs e Protocolos de Integração** - A biblioteca proposta será compatível com diferentes serviços de armazenamento em nuvem, cada um com características técnicas específicas. A seguir, são descritos os principais serviços e protocolos a serem integrados:

* FTP (*File Transfer Protocol*):

Protocolo clássico para transferência de arquivos em redes TCP/IP. Apesar de sua simplicidade e ampla adoção, o FTP não oferece mecanismos de segurança, como a criptografia. No projeto, o suporte ao FTP será útil para integração com sistemas legados ou servidores que ainda adotam esse protocolo como padrão.

* SFTP (SSH *File Transfer Protocol*):

Evolução segura do FTP, o SFTP utiliza o protocolo SSH (*Secure Shell*) para garantir criptografia na transmissão de dados e maior integridade. Sua compatibilidade com o FTP facilita migrações e oferece maior segurança para rotinas automatizadas de *backup* e sincronização, principalmente em sistemas que exigem confidencialidade nas operações de *upload* e *download*.

* Amazon S3 (*Simple Storage Service*):

Serviço de armazenamento da AWS que permite armazenar e recuperar qualquer volume de dados a qualquer momento, com alta durabilidade e escalabilidade. A integração será realizada por meio do SDK oficial da AWS, o que permitirá à biblioteca executar operações como *upload*, listagem e exclusão de arquivos de forma padronizada. O uso do S3 é ideal para aplicações que demandam alta disponibilidade e integração com outros serviços da AWS.

* Google Drive:

Plataforma de armazenamento e colaboração do *Google* baseada em nuvem. A integração será feita via API RESTful oficial, utilizando o protocolo OAuth 2.0 para autenticação segura. A biblioteca permitirá o gerenciamento de arquivos e pastas, controle de permissões e automatização de ações como *upload*, *download* e compartilhamento de documentos. O uso do *Google Drive* é especialmente relevante para ambientes corporativos e colaborativos, nos quais a centralização e o controle de acesso são prioritários.

**2.2 Trabalhos Relacionados**

Estudos recentes exploram soluções de integração entre sistemas e serviços em nuvem, apontando desafios comuns, como compatibilidade entre APIs e segurança de dados. O estudo de Desai et al. (2021), por exemplo, explora as funcionalidades do Microsoft Azure para a implantação de aplicações, abordando aspectos como escalabilidade, segurança e automação no desenvolvimento e implantação de serviços em nuvem.

O estudo de Sousa et al. (2010) oferece uma boa perspectiva sobre as arquiteturas e tecnologias de computação em nuvem, além dos desafios específicos da integração de serviços. Esse estudo é relevante para o desenvolvimento da biblioteca proposta neste projeto, que busca simplificar a integração com múltiplos serviços em nuvem, fornecendo uma interface unificada para os desenvolvedores.

Com base no projeto de Bhattacharjee et al. (2019), que aponta que a interação entre sistemas e serviços em nuvem exige conhecimento aprofundado e uma grande curva de aprendizado para cada serviço específico, foi proposta a ideia de um *framework* com comandos abstratos e interativos. O objetivo foi facilitar o uso e tornar os serviços em nuvem mais eficientes no cotidiano.

**3 Metodologia**

Este trabalho se baseia na metodologia de pesquisa aplicada e tecnológica, com o propósito de desenvolver uma biblioteca de *software* (*Middleware*) em PHP, voltada à padronização da integração entre sistemas *web* e serviços de armazenamento em nuvem. A biblioteca, nomeada *Cloudz*, já está funcional para integração com AWS S3, FTP e SFTP, restando apenas a finalização do módulo de *Google Drive*. O desenvolvimento de um sistema *web* demonstrativo será realizado na última etapa do projeto, com o objetivo de validar a aplicação prática da biblioteca.  
A arquitetura do projeto está estruturada nos padrões *Service Layer* e *Domain Model*, garantindo modularidade, reutilização e clareza nas responsabilidades.

**3.1 Ferramentas e Organização do Projeto**

Para o planejamento e acompanhamento das tarefas do projeto, foram adotadas as seguintes ferramentas:

* Trello**:** organização de tarefas e fluxos utilizando metodologia *Kanban*, com separação de etapas como “A Fazer”, “Em Desenvolvimento” e “Finalizado”.
* GitHub**:** Versionamento de código, controle de *branches*, documentação com README.md, *changelogs* e publicação do pacote via *Composer*.
* Composer**:** gerenciador de dependências utilizado para instalar bibliotecas auxiliares e preparar a distribuição da *Cloudz*.
* PHP, *Bootstrap*, MySQL e ferramentas de testes foram utilizadas como base tecnológica.

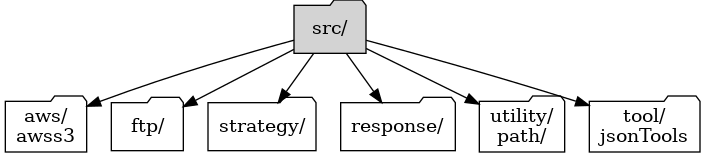
**3.2 Pesquisa**

A pesquisa teve caráter bibliográfico, exploratório e aplicado, com foco no estudo dos benefícios de centralizar um conjunto de códigos pré-escritos em uma biblioteca reutilizável. O objetivo é facilitar a integração de sistemas *web* com um ou mais serviços em nuvem, promovendo redução de esforço de manutenção, padronização de código e facilidade para novas integrações. Para embasar tecnicamente o projeto, foram estudados conceitos como reutilização de componentes, *middleware*, camadas de arquitetura e padrões de projeto (*Strategy*, *Facade*, *Factory*), além das tecnologias envolvidas no desenvolvimento.

**3.3 Desenvolvimento da Biblioteca**

O projeto foi estruturado modularmente sob o diretório src/, com divisão por responsabilidade. A figura abaixo representa a organização dos principais pacotes:

Figura 1 - Diagrama de Pacotes da Biblioteca CloudZ.

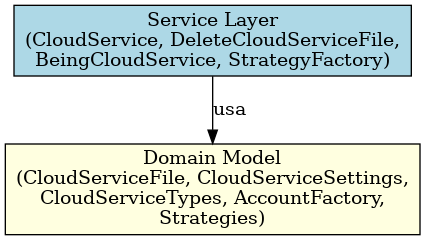


Fonte: Elaborado pelos autores.

**3.3.1 Arquitetura**

A biblioteca foi construída com base na arquitetura *Service Layer* + *Domain Model*, refletida no seguinte diagrama de camadas (Figura 2):

Figura 2 - Diagrama de Camadas da Biblioteca CloudZ.



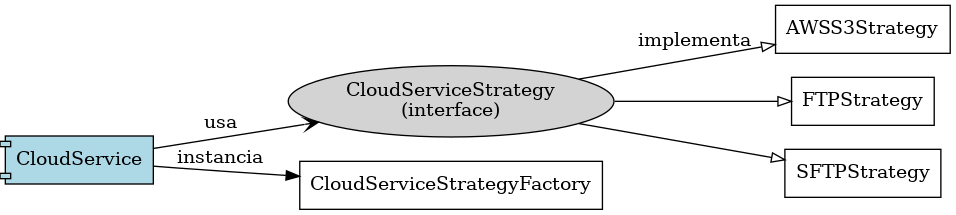
Fonte: Elaborado pelos autores.

A *Service Layer* é responsável por orquestrar as operações e interações com estratégias de armazenamento, enquanto a *Domain Model* encapsula objetos do domínio, como arquivos, configurações, estratégias e contas.

**3.3.2 Estratégias e Fábricas**

As integrações com os serviços em nuvem são definidas por meio de estratégias específicas que herdam da interface *CloudServiceStrategy*. A seleção da estratégia correta é realizada pela *CloudServiceStrategyFactory*, conforme ilustrado na Figura 3:

Figura 3 - Diagrama de Classes das Estratégias da CloudZ.



Fonte: Elaborado pelos autores.

**3.4 Validação**

A biblioteca será testada em um ambiente real de produção, por meio da integração com um sistema ERP *web* utilizado por uma empresa de tecnologia da informação situada na cidade de Araras, interior de São Paulo. A biblioteca será aplicada em operações reais do sistema, como *upload* e exclusão de arquivos diretamente em serviços em nuvem, atendendo demandas cotidianas do ambiente de trabalho.

Essa integração permitirá observar o comportamento da biblioteca em condições reais de uso, avaliando seu desempenho, robustez e confiabilidade. A empresa parceira fornecerá *feedbacks* contínuos durante os testes, contribuindo para o refinamento da biblioteca. Além disso, testes manuais com *scripts* PHP foram utilizados durante o desenvolvimento para simular erros, validar respostas das estratégias e assegurar o funcionamento da arquitetura modular implementada.

**4 Resultados**

O projeto encontra-se em estágio avançado, com a biblioteca *Cloudz* parcialmente concluída e já funcionando em diversos cenários. As funcionalidades implementadas foram validadas por meio de testes manuais e também por integração com um sistema ERP em ambiente de produção.

**4.1 Funcionalidades Implementadas**

* Integração completa com os serviços AWS S3, FTP e SFTP, com suporte a *upload*, exclusão e listagem de arquivos.
* Implementação do padrão de projeto *Facade*, que permite ao desenvolvedor interagir com qualquer serviço por meio de uma única interface.
* Modularização com uso de *Service Layer* e *Domain Model*, conforme representado nos diagramas apresentados na metodologia.
* Utilização de estratégias distintas (AWSS3Strategy, FTPStrategy, SFTPStrategy) escolhidas dinamicamente por meio da fábrica CloudServiceStrategyFactory.

**4.2 Testes Realizados**

Os testes envolveram:

* *Upload* e exclusão de arquivos utilizando diferentes serviços;
* Simulação de falhas de autenticação e conectividade;
* Validação das respostas da biblioteca utilizando as classes *CloudServiceResponse.*

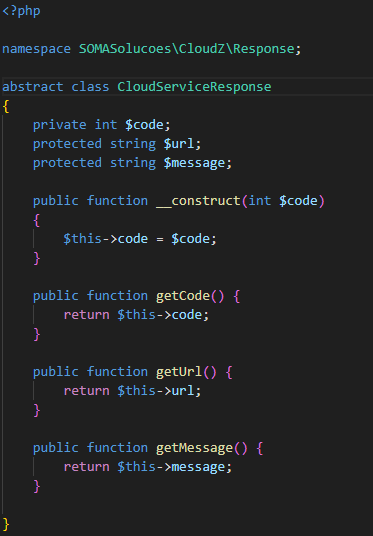
A biblioteca será integrada a um ERP *Web* de uma empresa de tecnologia da cidade de Araras/SP, permitindo seu uso em situações reais de operação. Durante esse processo, foram observadas:

* Redução no tempo de desenvolvimento de integrações;
* Facilidade na manutenção de código;
* *Feedbacks* positivo da equipe técnica da empresa parceira.

**4.3 Evidências Visuais**

As seguintes evidências visuais foram geradas para ilustrar o funcionamento da biblioteca *Cloudz* e suas principais funcionalidades já implementadas.

Figura 4 - Classe de retorno de sucesso.

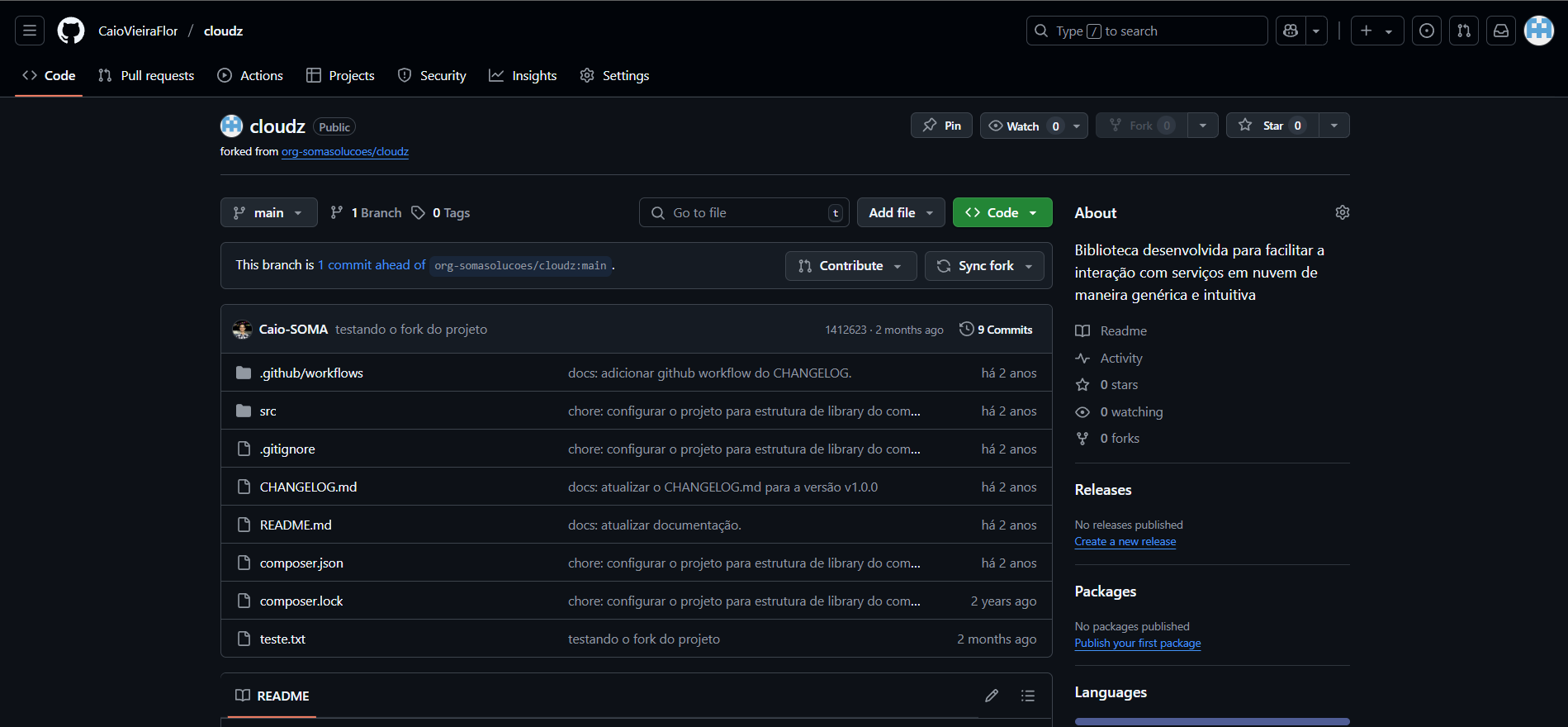


Fonte: Elaborado pelos autores.

Figura 4 - Classe base de resposta da biblioteca *Cloudz*.

A imagem apresenta a classe abstrata *CloudServiceResponse*, responsável por definir a estrutura comum de resposta utilizada pelas estratégias da biblioteca *Cloudz*. Essa classe serve como superclasse para implementações específicas como *CloudServiceResponseSuccess* e *CloudServiceResponseError*, padronizando os atributos *code*, *url* e *message*, além dos métodos de acesso correspondentes.

Figura 5 - Repositório da biblioteca no GitHub

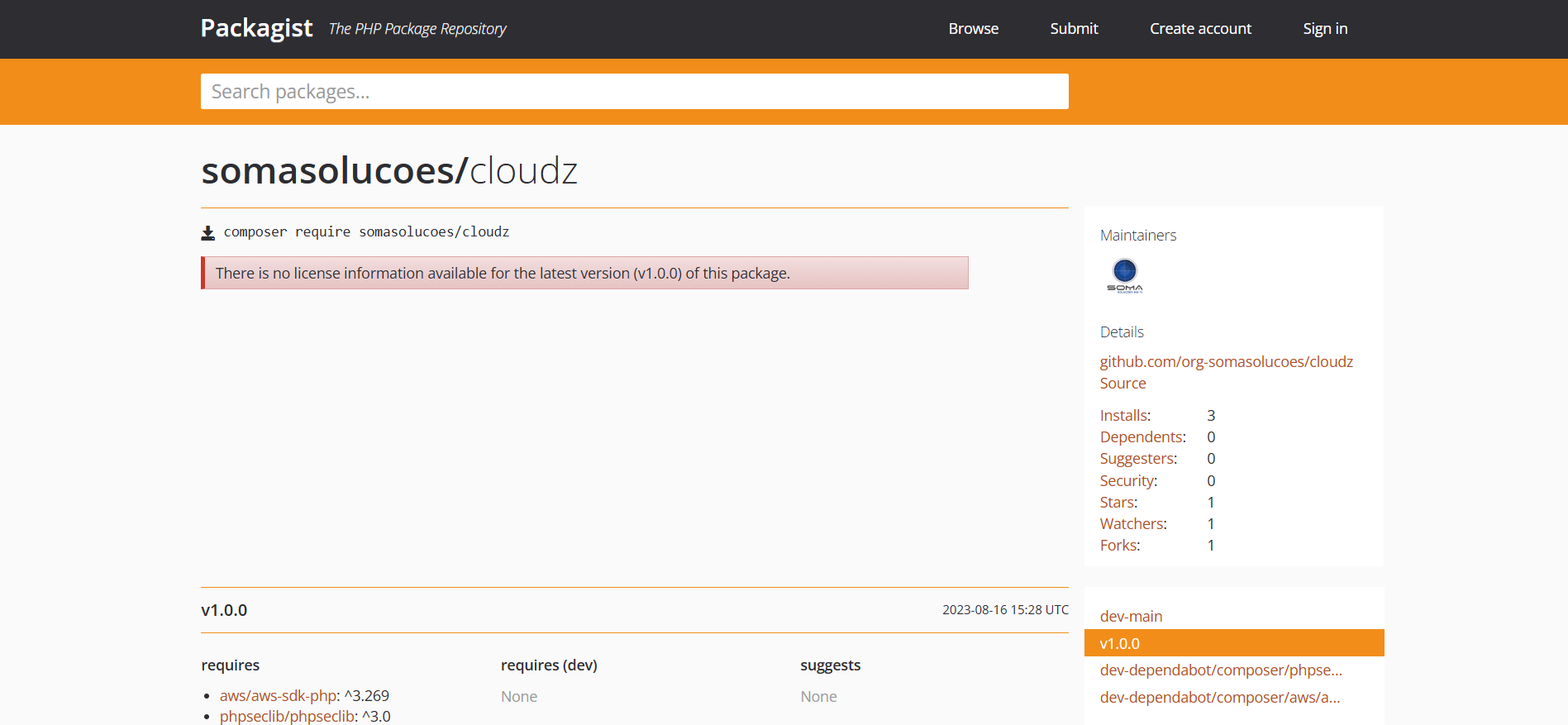


Fonte: Elaborado pelos autores.

Figura 5 - Repositório da biblioteca *Cloudz* no *GitHub* com *README*.

A imagem apresenta o repositório público da biblioteca hospedado no *GitHub*. É possível visualizar o arquivo *README*, que contém instruções de instalação, exemplos de uso e estrutura de diretórios, o que demonstra a preocupação com a documentação e distribuição via *Composer*.

Figura 6 - Primeira versão disponível da biblioteca



Fonte: Elaborado pelos autores.

A imagem mostra a publicação da biblioteca *Cloudz* no repositório oficial *Packagist*, com a versão 1.0.0 disponível para instalação. Nela é possível visualizar o comando de instalação via *Composer*, as dependências utilizadas (como o SDK da AWS e a biblioteca *phpseclib*), e os metadados do pacote. Essa publicação demonstra que a biblioteca está pronta para ser integrada a outros projetos PHP de forma padronizada e automatizada.

**5 Considerações Finais**

O projeto desenvolvido teve como objetivo a criação de uma biblioteca *middleware* em PHP, voltada à integração de sistemas *web* com diferentes serviços de armazenamento em nuvem. Através da aplicação de padrões de projeto como *Facade*, *Strategy* e *Factory*, e da arquitetura baseada em *Service Layer* e *Domain Model*, foi possível desenvolver uma solução modular, reutilizável e escalável.  
Entre os objetivos específicos propostos, destacam-se:

* A implementação das integrações com AWS S3, FTP e SFTP, que foram concluídas com sucesso;
* A padronização do uso de estratégias de serviço por meio de uma interface central (*facade*);
* A estruturação da biblioteca com foco em extensibilidade e desacoplamento.

A biblioteca *Cloudz* encontra-se funcional e em estágio avançado. A próxima etapa do projeto inclui a integração da biblioteca a um sistema ERP *web* utilizado por uma empresa de tecnologia da informação. Essa validação prática em ambiente de produção permitirá observar o desempenho da solução em cenários reais de uso e coletar *feedbacks* para futuros aprimoramentos.  
Apesar dos avanços, algumas funcionalidades ainda estão em desenvolvimento, como a integração com o *Google Drive* utilizando OAuth 2.0. O sistema *web* demonstrativo também será finalizado na próxima fase, permitindo uma apresentação mais amigável das capacidades da biblioteca.  
Como próximos passos, destaca-se a intenção de expandir a compatibilidade com novos serviços em nuvem e realizar melhorias na interface e experiência do sistema demonstrativo.

Conclui-se, portanto, que os resultados alcançados até o momento comprovam a viabilidade e utilidade da proposta, oferecendo uma ferramenta que pode ser amplamente utilizada por desenvolvedores que necessitam de integração com múltiplos serviços em nuvem em seus sistemas *web*.

**Referências Bibliográficas**

ARAUJO, Herbert Vargas de; FLORENTINO, Paulo Henrique A. **Análise de serviços de armazenamento por objetos em provedores de nuvens públicas.** 2021. 46 f. Trabalho de conclusão de curso (Licenciatura em Computação) - Universidade de Brasília, Brasília, 2021. Disponível em: <https://bdm.unb.br/handle/10483/31190>. Acesso em: 10 set. 2024.

AZURE (Brasil) (org.). **Benefícios da migração na nuvem**. 2024. Disponível em: [https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/benefits-of-clo](https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/benefits-of-cloud-migration/) [ud-migration/](https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/benefits-of-cloud-migration/). Acesso em: 09 set. 2024.

BARRETO, Ahilton; MURTA, Leonardo; ROCHA, Ana Regina. Uma Abordagem para Definição de Processos de Software Baseada em Reutilização. In: **ProQuality (UFLA) III Workshop de Implementadores MPS. BR**. 2007. p. 33-39. Disponível em: <https://leomurta.github.io/papers/barreto2007.pdf>. Acesso em: 13 mai. 2025.

BARROS, Heitor José dos Santos et al. Um Middleware adaptável para descoberta, composição e invocação automática de Serviços Web Semânticos. 2011. Disponível em: <https://www.repositorio.ufal.br/handle/riufal/845>. Acesso em: 13 mai. 2025.

BENTO, Evaldo Junior. **Desenvolvimento web com PHP e MySQL**. Editora Casa do Código, 2021. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=xG2CCwAAQBAJ&oi=fnd&pg=PT3&dq=composer+PHP&ots=___mZJEeOw&sig=th1hOQjJXS017BaznkbS_49bmnM#v=onepage&q=composer%20PHP&f=false>. Acesso em: 19 mai. 2025.

BHATTACHARJEE, Anirban; BARVE, Yogesh; GOKHALE, Aniruddha. **CloudCamp: Automating cloud services deployment & managent.** 2019. Vanderbilt University, Department of Electrical Engineering and Computer Science. Nashville, Tennessee, USA. 2019. Disponível em: <https://ieeexplore.ieee.org/abstract/document/8456424>. Acesso em: 08 nov. 2024.

CONVERSE, Tim; PARK, Joyce. **PHP: a bíblia**. Gulf Professional Publishing, 2003. 710 f. Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Rio de Janeiro, 2003. Disponível em:

[https://scholar.google.com.br/scholar?hl=pt-BR&as\_sdt=0%2C5&q=CONVERSE%2](https://scholar.google.com.br/scholar?hl=pt-BR&as_sdt=0%2C5&q=CONVERSE%2C%2BTim%3B%2BPARK%2C%2BJoyce.%2BPHP%3A%2Ba%2Bb%C3%ADblia.%2BGulf%2BProfessional%2BPublishing%2C%2B2003.&btnG) [C+Tim%3B+PARK%2C+Joyce.+PHP%3A+a+b%C3%ADblia.+Gulf+Professional+Pu](https://scholar.google.com.br/scholar?hl=pt-BR&as_sdt=0%2C5&q=CONVERSE%2C%2BTim%3B%2BPARK%2C%2BJoyce.%2BPHP%3A%2Ba%2Bb%C3%ADblia.%2BGulf%2BProfessional%2BPublishing%2C%2B2003.&btnG) [blishing%2C+2003.&btnG=](https://scholar.google.com.br/scholar?hl=pt-BR&as_sdt=0%2C5&q=CONVERSE%2C%2BTim%3B%2BPARK%2C%2BJoyce.%2BPHP%3A%2Ba%2Bb%C3%ADblia.%2BGulf%2BProfessional%2BPublishing%2C%2B2003.&btnG). Acesso em: 20 set. 2024.

LEITHARDT, Valderi Reis Quietinho. UbiPri: middleware para controle e gerenciamento de privacidade em ambientes ubíquos. 2015. Disponível em: <https://lume.ufrgs.br/handle/10183/147774>. Acesso em: 19 mai. 2025.

MALDONADO, José Carlos. **Padrões e frameworks de software.** 2002. 35 f. Notas Didáticas, Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, ICMC/USP, São Paulo, 2002. Disponível em: <https://sites.icmc.usp.br/rtvb/apostila.pdf>. Acesso em: 10 set. 2024.

PRESCOTT, Preston. **Programação em JavaScript**. 2016. 73 f. Babelcube Inc. 2016. Disponível em: [https://books.google.com.br/books?id=J9mGCwAAQBAJ&lpg=PT4&ots=aK3sxDYbYr&dq=PRESCOTT%2C%20Preston.%20Programa%C3%A7%C3%A3o%20em%20JavaScript.%202016.%20%20Babelcube%20Inc.%2C%202016.&lr&hl=pt-BR&pg=PT](https://books.google.com.br/books?id=J9mGCwAAQBAJ&lpg=PT4&ots=aK3sxDYbYr&dq=PRESCOTT%2C%20Preston.%20Programa%C3%A7%C3%A3o%20em%20JavaScript.%202016.%20%20Babelcube%20Inc.%2C%202016.&lr&hl=pt-BR&pg=PT4%23v%3Donepage&q&f=false) [4#v=onepage&q&f=false](https://books.google.com.br/books?id=J9mGCwAAQBAJ&lpg=PT4&ots=aK3sxDYbYr&dq=PRESCOTT%2C%20Preston.%20Programa%C3%A7%C3%A3o%20em%20JavaScript.%202016.%20%20Babelcube%20Inc.%2C%202016.&lr&hl=pt-BR&pg=PT4%23v%3Donepage&q&f=false). Acesso em: 01 out. 2024.

SILVA, Matheus Nícolas da; CARVALHO, Marcus. **Análise de Mecanismos de Serverless Computing em Ambientes de Nuvens Computacionais**. 2018. 15 f. TCC (Graduação) - Curso de Sistemas de Informação, Universidade Federal da Paraíba (Ufpb) Campus, Rio Tinto, 2018. Disponível em: <https://repositorio.ufpb.br/jspui/bitstream/123456789/17169/1/MNS08102018.pdf>. Acesso em: 09 set. 2024.

V.P, Desai; K.S, Oza; P.P, Shinde; P.G, Naik. **Microsoft Azure: Cloud platform for application service deployment**. 2021. 5 f. Department of Computer Science,

Shivaji University, Kolhapur, India. 2021. Disponível em: [https://www.researchgate.net/publication/356556260\_Microsoft\_Azure\_Cloud\_Platfor](https://www.researchgate.net/publication/356556260_Microsoft_Azure_Cloud_Platform_for_Application_Service_Deployment) [m\_for\_Application\_Service\_Deployment](https://www.researchgate.net/publication/356556260_Microsoft_Azure_Cloud_Platform_for_Application_Service_Deployment). Acesso em: 11 nov. 2024.

1. FHO|UNIARARAS. Aluno do Curso de Sistemas de Informação, Caio Eduardo Vieira Flor, caio.flor@alunos.uniararas.br [↑](#footnote-ref-0)
2. FHO|UNIARARAS. Aluno do Curso de Sistemas de Informação, Lucas Soares dos Santos, lucassoares12016@alunos.fho.edu.br [↑](#footnote-ref-1)
3. FHO|UNIARARAS. Professor do Curso de Sistemas de Informação, Diego Henrique Negretto, diegonegretto@fho.edu.br [↑](#footnote-ref-2)