INSTITUTO FEDERAL GOIANO - CAMPUS MORRINHOS
CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

Eduardo dos Santos Pires

RELATÓRIO TÉCNICO: Uso de padrões de projeto e Transport Layer Security (TLS) no contexto do Framework Web Symfony

MORRINHOS - GO
2025

Eduardo dos Santos Pires

RELATÓRIO TÉCNICO: Uso de padrões de projeto e Transport Layer Security (TLS) no contexto do Framework Web Symfony

Trabalho de Conclusão de Curso ao Instituto Federal Goiano de Ciências e Tecnologias como requisito parcial para obtenção do título de tecnólogo em sistemas para a internet.

Área de concentração: **Desenvolvimento de Sistemas.**

Orientador: **Rodrigo Elias Francisco**

MORRINHOS - GO

2025

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas – SIBI/IF Goiano Campus Morrinhos

P667r Pires, Eduardo dos Santos.
Relatório Técnico: Uso de padrões de projeto e Transport Layer Security(TLS) no contexto do Framework Web Symfony. / Eduardo dos Santos Pires. – Morrinhos, GO: IF Goiano, 2025.
29 f. : il. color.

Orientador: Dr. Rodrigo Elias Francisco.

Trabalho de conclusão de curso (graduação) – Instituto Federal Goiano Campus Morrinhos, Tecnologia em Sistemas para Internet, 2025.

1. API REST. 2. PHP. 3. Padrões de Projeto. I. Francisco, Rodrigo Elias. II. Instituto Federal Goiano. III. Título.

CDU 004.415.2

Fonte: Elaborado pela Bibliotecária-documentalista Morgana Guimarães, CRB1/2837

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO

PARA DISPONIBILIZAR PRODUÇÕES TÉCNICO-CIENTÍFICAS NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Com base no disposto na Lei Federal nº 9.610, de 19 de fevereiro de 1998, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano a disponibilizar gratuitamente o documento em formato digital no Repositório Institucional do IF Goiano (RIIF Goiano), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

IDENTIFICAÇÃO DA PRODUÇÃO TÉCNICO-CIENTÍFICA

- ☐ Tese (doutorado)
- ☐ Dissertação (mestrado)
- ☐ Monografia (especialização)
- ☑ TCC (graduação)
- ☐ Artigo científico
- ☐ Capítulo de livro
- ☐ Livro
- ☐ Trabalho apresentado em evento
- ☐ Produto técnico e educacional - Tipo:

Nome completo do autor:
Eduardo dos Santos Pires

Matrícula:
2021104211710066

Título do trabalho:
RELATÓRIO TÉCNICO: Uso de padrões de projeto e Transport Layer Security (TLS) no contexto do Framework Web Symfony

RESTRIÇÕES DE ACESSO AO DOCUMENTO

Documento confidencial: ☑ Não ☐ Sim, justifique:

Informe a data que poderá ser disponibilizado no RIIF Goiano: 19 / 12 / 2025

O documento está sujeito a registro de patente? ☐ Sim ☑ Não

O documento pode vir a ser publicado como livro? ☐ Sim ☑ Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O(a) referido(a) autor(a) declara:

• Que o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;

• Que obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autoria, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;

• Que cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

Morrinhos — Local
19 / 12 / 2025 — Data

gov.br Documento assinado digitalmente
EDUARDO DOS SANTOS PIRES
Data: 18/12/2025 22:39:42-0300
Verifique em https://validar.iti.gov.br

Assinatura do autor e/ou detentor dos direitos autorais

Ciente e de acordo:

gov.br Documento assinado digitalmente
RODRIGO ELIAS FRANCISCO
Data: 19/12/2025 08:14:10-0300
Verifique em https://validar.iti.gov.br

entador(a)

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO

Ata nº 32/2025 - CCEPTNM-MO/CEPTNM-MO/DE-MO/CMPMHOS/IFGOIANO

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

ATA DE APRESENTAÇÃO DE TRABALHO DE CURSO

Aos **9** dias do mês de **dezembro** de **2025**, às **19:00** horas, foi realizada, remotamente via Google Meet, a apresentação pública do trabalho de curso do discente **Eduardo dos Santos Pires** intitulado **RELATÓRIO TÉCNICO: Uso de padrões de projeto e Transport Layer Security (TLS) no contexto do Framework Web Symfony**, como requisito necessário para a conclusão do curso. A Banca Examinadora, constituída pelos professores: **Rodrigo Elias Francisco** – orientador, **José Pereira Alves**, **Alexandre Carvalho Silva**. Após a análise, emitiram o seguinte resultado:
(x) Aprovado
( ) Aprovado com ressalva

(A Banca Examinadora deve definir as exigências a serem cumpridas pelo aluno na revisão, ficando o orientador responsável pela verificação do cumprimento das mesmas.)

Observações:_____
_____
( ) Reprovado com o seguinte parecer: _____
_____

Morrinhos, 9 de dezembro de 2025

Por ser verdade firmamos a presente:

(Assinado Eletronicamente)
Rodrigo Elias Francisco (Presidente da banca)

(Assinado Eletronicamente)
Alexandre Carvalho Silva (Membro)

(Assinado Eletronicamente)
José Pereira Alves (Membro)

Documento assinado eletronicamente por:

- **Rodrigo Elias Francisco**, **PROFESSOR ENS BASICO TECN TECNOLOGICO** , em 09/12/2025 20:39:34.
- **Alexandre Carvalho Silva**, **PROFESSOR ENS BASICO TECN TECNOLOGICO** , em 09/12/2025 20:47:36.
- **Jose Pereira Alves**, **PROFESSOR ENS BASICO TECN TECNOLOGICO** , em 09/12/2025 21:15:17.

Este documento foi emitido pelo SUAP em 09/12/2025. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 772844
Código de Autenticação: 4c1495d304

INSTITUTO FEDERAL GOIANO
Campus Morrinhos
Rodovia BR-153, Km 633, Zona Rural, SN, Zona Rural, MORRINHOS / GO, CEP 75650-000
(64) 3413-7900

RESUMO

O presente trabalho tem como objetivo demonstrar a importância dos padrões de projeto de software e da arquitetura bem definida na criação de sistemas profissionais e escaláveis. Para ilustrar esses conceitos foi desenvolvida uma API Rest para controle de gastos, utilizando o framework *Symfony* PHP.

A escolha de um modelo de negócio, não muito denso, de controle de gastos, permite uma análise detalhada da aplicação prática dos padrões arquitetônicos de software. Utilizando o sistema gerenciador de banco de dados MySQL, embora a flexibilidade da biblioteca ORM Doctrine permita a substituição por outros sistemas com facilidade. Além disso, foi abordada uma explicação da camada de segurança web implementada no projeto através do TLS.

Por meio de uma abordagem prática, que é o foco do presente curso, e detalhada, demonstrou-se como a aplicação correta de conceitos da área de desenvolvimento e tecnologia da informação podem impactar positivamente o desenvolvimento de software.

Palavras-chave: API REST, PHP, Padrões de Projeto

ABSTRACT

This work aims to demonstrate the importance of software design patterns and a well-defined architecture in the creation of professional and scalable systems. To illustrate these concepts, a REST API for expense control was developed using the Symfony PHP framework.

Choosing a business model that isn't too dense for cost control allows for a detailed analysis of the practical application of software architectural patterns. The MySQL database management system was used, although the flexibility of the Doctrine ORM library allows for easy replacement with other systems. Furthermore, an explanation of the web security layer implemented in the project using TLS was provided.

Through a practical and detailed approach, which is the focus of this course, it was demonstrated how the correct application of concepts from the area of development and information technology can positively impact software development.

Keywords: API REST, PHP, Design Patterns.

SUMÁRIO

1 INTRODUÇÃO

A célebre frase “não existe software perfeito” ressoa como um lembrete atemporal das limitações inerentes ao trabalho humano, especialmente em uma área tão desafiadora como o desenvolvimento de software. Como destacam Pressman e Maxim (2016), mesmo os sistemas mais bem projetados e testados ainda estarão sujeitos a defeitos. Essa constatação, longe de ser um obstáculo, impulsiona a busca incessante pela melhoria contínua, mesmo sabendo que a perfeição é uma meta inatingível.

Reconhecendo que a perfeição é inalcançável, a engenharia de software se apoia em métodos que buscam reduzir falhas e elevar a confiabilidade. Como destacam Gamma et al. (1995), os padrões de projeto oferecem soluções comprovadas para problemas recorrentes no desenvolvimento, servindo de referência robusta há décadas. No entanto, embora haja vasta literatura sobre o tema, muitos profissionais iniciantes encontram dificuldades em compreender sua aplicação prática no desenvolvimento de sistemas.

El Boussaidi e Mili (2011) destacam que o uso adequado dos padrões exige três condições principais: compreendê-los, avaliar sua relevância frente ao problema de design e aplicá-los de forma correta — etapas que não são triviais. Nesse contexto, o problema central que este trabalho busca responder é sobre como demonstrar, de forma prática, a relevância da aplicação estruturada de padrões de projeto e frameworks modernos no desenvolvimento de um sistema de controle de saldo.

Assim, este trabalho demonstra como a aplicação estruturada desses padrões, integrada ao uso de frameworks modernos, possibilita criar sistemas sustentáveis e capazes de evoluir diante das constantes demandas do mercado.

1.1. JUSTIFICATIVA

O avanço constante das tecnologias digitais e a crescente complexidade dos sistemas de informação exigem soluções cada vez mais organizadas, seguras e

escaláveis. Nesse cenário, torna-se relevante demonstrar a aplicação prática de padrões de projeto e frameworks, não apenas como recurso didático, mas como estratégia de aproximação entre teoria e prática no desenvolvimento de software. Assim, este trabalho busca justificar sua importância ao apresentar uma proposta que alia conceitos consolidados da engenharia de software com a implementação de um sistema funcional, voltado ao controle de receitas e despesas, que pode ser aplicado em contextos reais de indivíduos ou pequenas empresas.

1.3. OBJETIVOS

O objetivo principal deste trabalho é demonstrar, por meio de um caso prático, como a aplicação estruturada de padrões de projeto e tecnologias modernas contribui para o desenvolvimento de sistemas mais organizados, escaláveis e seguros. Para isso, adotou-se a construção de uma API REST utilizando o framework Symfony, integrando práticas arquitetônicas e mecanismos de segurança baseados no protocolo TLS.

Para atingir esse objetivo central, estabelecem-se como objetivos específicos:

- Aplicar padrões de projeto no contexto do framework Symfony, evidenciando como esses padrões fortalecem a arquitetura e contribuem para a modularidade e extensibilidade do sistema.
- Implementar uma *Application Programing Interface* (API) de controle de saldo contendo recursos de autenticação, autorização e operações relacionadas às movimentações financeiras.
- Configurar e demonstrar o uso de práticas de segurança de redes no ambiente de desenvolvimento, reforçando a importância de estabelecer canais seguros de comunicação entre cliente e servidor.
- Desenvolver um *front-end* com o *framework Vue.js* para demonstrar o fluxo de interação com a API, apresentando as operações principais e o consumo dos recursos implementados.

- Evidenciar o processo de integração entre API, front-end e agente inteligente, demonstrando como diferentes camadas de uma aplicação moderna podem operar de maneira coesa.

1.4. METODOLOGIA

Para atingir os objetivos propostos, adotou-se uma metodologia prática e aplicada, dividida em etapas:

1) Inicialmente, realizou-se uma revisão bibliográfica abordando padrões de projeto, arquitetura de software e protocolos de segurança, com foco na contextualização e fundamentação teórica. Foram analisadas obras clássicas da engenharia de software, documentação oficial do Symfony e publicações relevantes sobre TLS e APIs modernas.
2) Preparou-se a infraestrutura local para suportar a execução da aplicação. Utilizou-se o pacote XAMPP para o gerenciamento do banco de dados MySQL. O gerenciamento de dependências do *back-end* foi realizado através do *Composer*, e a inicialização do servidor local foi feita via *Symfony Command Line Interface* (CLI). Todo o código-fonte foi versionado utilizando o sistema *Git*, com repositório hospedado na plataforma *GitHub*, garantindo o controle de alterações e histórico do projeto.
3) Foi feita a construção da lógica de negócio utilizando o framework *Symfony*. A modelagem de dados foi realizada através do *Object-Relational Mapping* (ORM) *Doctrine*. O foco central consistiu na refatoração e implementação dos Padrões de Projeto selecionados para resolver problemas de acoplamento e escalabilidade. Paralelamente, implementaram-se os mecanismos de segurança, incluindo a geração de certificados auto-assinados para habilitar o protocolo TLS.
4) Também construiu-se uma interface de usuário utilizando o framework *Vue.js* na versão 3, com a ferramenta de construção *Vite* e a biblioteca de componentes visuais *Vuetify*. Integrado a esta interface, desenvolveu-se um

agente assistente baseado em Inteligência Artificial, servindo como prova de conceito da interoperabilidade do sistema.

5) Por fim, a partir dessas implementações descritas, foram gerados diagramas, e análises que permitem avaliar os benefícios decorrentes da aplicação das práticas estudadas, relacionando teoria e prática em um único estudo de caso.

1.5. ORGANIZAÇÃO DO TRABALHO

Este relatório está organizado da seguinte forma: a Seção 2 apresenta os conceitos fundamentais; a Seção 3 descreve a proposta de desenvolvimento e implementação; e, por fim, a Seção 4 traz as considerações finais.

2 CONCEITOS FUNDAMENTAIS

O desenvolvimento de software moderno se apoia em uma base teórica e prática composta por conceitos que orientam a criação de sistemas robustos, escaláveis e de fácil manutenção. Entre esses conceitos, destacam-se os padrões de projeto, a arquitetura de software, os frameworks e as APIs, todos fundamentais para estruturar soluções que respondam de forma eficiente às demandas atuais. A seguir, serão explorados esses elementos e suas contribuições para o processo de construção de sistemas.

2.1 PADRÕES DE PROJETO

Os padrões de projeto são soluções reutilizáveis para problemas recorrentes no design de software, utilizadas há décadas para melhorar a manutenibilidade, escalabilidade e comunicação entre desenvolvedores. Desde a publicação original de *Design Patterns* (1995), tornou-se consenso que tais padrões ajudam a escolher alternativas de design que tornam o sistema reutilizável e evitam aquelas que comprometem a reutilização. Essa durabilidade comprova sua relevância, mantendo-se atual mesmo em contextos tecnológicos modernos.

2.2 ARQUITETURA DE SOFTWARE

A arquitetura de software define o arcabouço estrutural de um sistema: seus componentes, propriedades externas e relações entre eles (SOMMERVILE, 2019). Essa definição formal fundamenta o entendimento de que a arquitetura é o elemento-chave que orienta decisões de projeto, afeta atributos de qualidade (como desempenho, confiabilidade e escalabilidade) e permite que sistemas evoluam com segurança ao longo do tempo.

2.3 FRAMEWORK (*SYMFONY*)

O Symfony destaca-se no ecossistema PHP por oferecer um conjunto de componentes reutilizáveis e uma arquitetura fundamentada em boas práticas de engenharia de software. Sua estrutura orientada a serviços, aliada ao uso extensivo de padrões de projeto como a Injeção de Dependência, favorece a modularidade, o baixo acoplamento e a escalabilidade das aplicações. Dessa forma, o Symfony exemplifica como frameworks modernos podem traduzir conceitos teóricos de arquitetura de software em soluções práticas, robustas e aplicáveis no mercado (SYMFONY, 2024).

2.4 API

Uma Interface de Programação de Aplicações (API) é um conjunto de regras que possibilita a comunicação entre diferentes softwares, permitindo a integração de sistemas de forma padronizada e segura. No contexto atual, as APIs assumem papel central no desenvolvimento de arquiteturas escaláveis, como por exemplo micro-serviços, que priorizam o design modular e reutilizável. De acordo com a IBM (2024), uma API pode ser compreendida como um conjunto de regras e protocolos que possibilita a comunicação e a troca de informações entre diferentes aplicações de software, promovendo a integração e o compartilhamento de recursos de maneira padronizada.

2.5 TLS

O *Transport Layer Security* (TLS) é um protocolo criptográfico amplamente utilizado para assegurar confidencialidade, integridade e autenticação na comunicação entre sistemas. Por meio de certificados digitais e criptografia assimétrica, eles estabelecem camadas adicionais de segurança aos canais de redes de computadores, protegendo informações sensíveis contra interceptações e alterações maliciosas. Embora tenha substituído o *Secure Socket Layer* (SSL), o termo SSL ainda é usado de forma genérica para designar conexões seguras, pois ambos possuem o mesmo propósito dentro da aplicação em redes. Conforme Rescorla (2018), protocolos como o TLS são fundamentais para garantir comunicações confiáveis em ambientes de rede inseguros, justificando sua aplicação prática no uso do Symfony para geração de certificados e configuração de HTTPS.

3 SISTEMA PROPOSTO

A proposta deste trabalho consiste no desenvolvimento de uma API REST voltada ao controle de saldo, com funcionalidades de registro de receitas e despesas, além de mecanismos de autenticação e autorização para assegurar a integridade das informações. A solução foi construída utilizando o framework Symfony, que fornece um arcabouço robusto para a criação de aplicações web profissionais, em conjunto com demais bibliotecas do mesmo ecossistema.

O sistema foi desenvolvido adotando padrões de projeto como forma de estruturar o código de maneira clara e reutilizável, favorecendo a manutenção e a escalabilidade. Adicionalmente, integra recursos de segurança por meio do uso de certificados digitais SSL/TLS, garantindo que as comunicações ocorram em um canal criptografado. A combinação desses elementos permite alinhar teoria e prática, demonstrando a relevância de boas práticas de engenharia de software na construção de aplicações modernas.

3.1 PADRÕES DE PROJETOS UTILIZADOS

O desenvolvimento de sistemas de software robustos exige soluções que não apenas atendam às necessidades funcionais, mas que também promovam clareza, flexibilidade e possibilidade de evolução futura. Nesse sentido, “a aplicação estruturada de padrões de projeto promove modularidade, favorece a manutenibilidade e a reutilização do código, além de estabelecer uma linguagem compartilhada entre desenvolvedores” (LOGESWARAMOORTHY, 2020).

Na implementação proposta, foram empregados diferentes padrões de projeto, cada um com objetivos específicos e aplicados em pontos estratégicos do sistema. Entre eles, destacam-se padrões criacionais, estruturais e comportamentais, como Singleton, Factory, Repository, Strategy, Template Method e Observer. Cada um desses padrões será descrito a seguir, acompanhado de

exemplos de aplicação prática no sistema e de representações gráficas que auxiliam na compreensão de sua estrutura e funcionamento.

3.1.1 REPOSITORY

Segundo Fowler (2003), o padrão Repository atua como uma coleção intermediária entre a lógica de domínio e a camada de persistência, promovendo o isolamento do acesso a dados. No projeto, essa abordagem é implementada por meio do Doctrine Repository, que centraliza as consultas e operações de banco de dados.

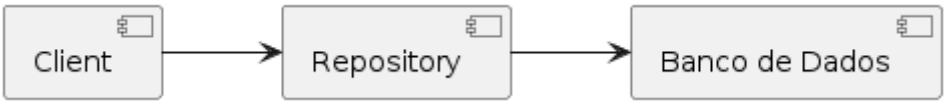

Figura 1 – Diagrama conceitual do *Repository*

Esse padrão traz clareza ao separar as regras de negócio dos detalhes da persistência, o que facilita a manutenção e amplia a testabilidade do sistema. Na Figura 1 vemos como esse padrão abstrai o acesso à fonte de dados do sistema, tornando o sistema mais desacoplado e manutenível.

3.1.2 DEPENDENCY INJECTION (DI)

A *Dependency Injection*, ou Injeção de Dependências, é um padrão fundamental utilizado para promover o baixo acoplamento entre componentes no desenvolvimento de software. Segundo Fowler (2004), esse padrão delega a responsabilidade de criação e fornecimento de dependências a um terceiro elemento — geralmente denominado *container* ou *injector* —, permitindo que as classes consumam recursos sem criar instâncias diretamente.

A Figura 2 apresenta um diagrama genérico que ilustra o funcionamento conceitual da *Injeção de Dependências*. Nessa representação, o componente *Injector* é responsável por criar e fornecer instâncias das dependências necessárias

aos componentes *Client*, *ServiceA* e *ServiceB*, conforme suas interfaces. Dessa forma, o *Client* não precisa conhecer as implementações concretas de suas dependências, apenas suas abstrações, promovendo modularidade e testabilidade.

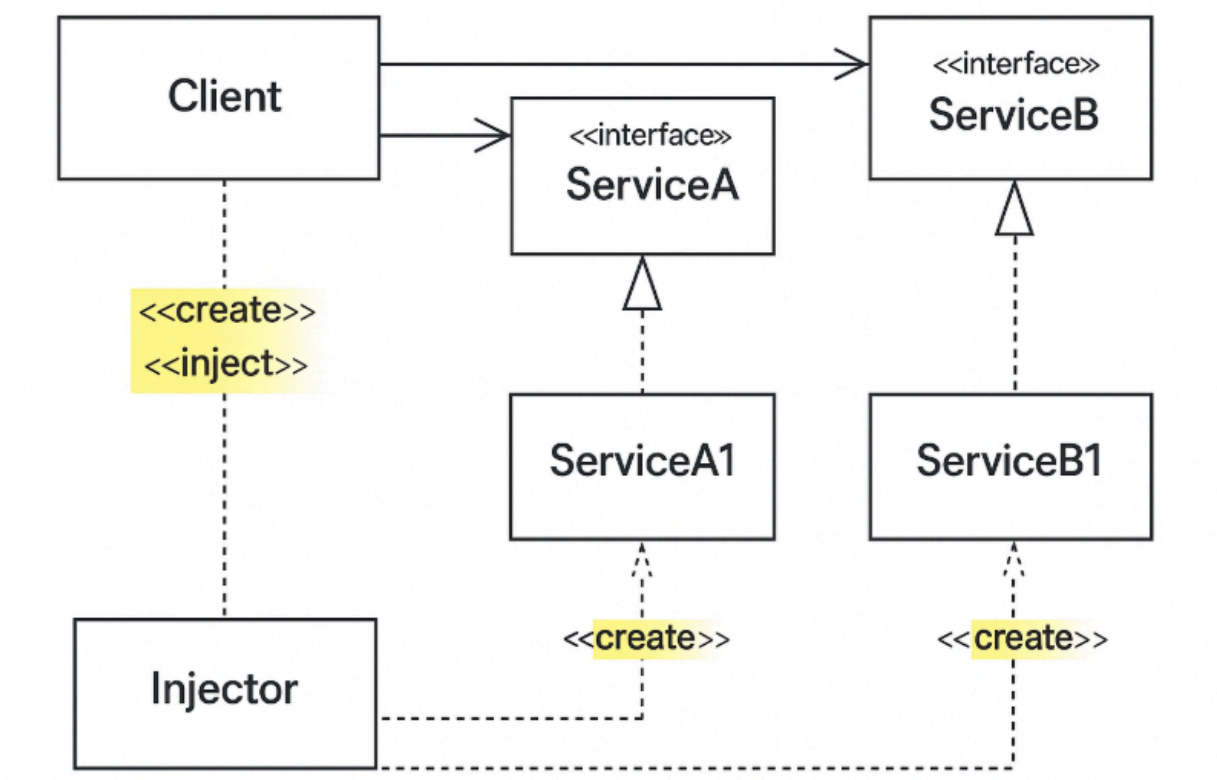

Figura 2 – Diagrama conceitual da Injeção de Dependências

No contexto do *Symfony Framework*, o princípio é o mesmo, mas a implementação ocorre de forma automatizada. O *Symfony* utiliza um arquivo de configuração (*services.yaml*) para declarar os serviços e suas dependências. Durante o processo de inicialização, o *framework* lê essas definições, registra cada serviço dentro de um container, e injeta automaticamente as dependências nas classes que as solicitam. (SYMFONY, 2025).

```php
public function __construct(
    SerializerInterface $serializer,
    ServiceLocatorInterface $locator
) {
    parent::__construct($serializer);
    $this->serviceLocator = $locator;
}
```

Figura 3 – Construtor de um serviço que depende de abstrações

Essa abordagem elimina a necessidade de um *injector* explícito, pois o contêiner interno do *Symfony* atua como tal, gerenciando o ciclo de vida dos objetos e resolvendo suas dependências com base no tipo declarado no construtor das classes. O trecho de código da Figura 3 ilustra esse funcionamento, mostrando uma classe que recebe via injeção automática os serviços *SerializerInterface* e *ServiceLocatorInterface*.

3.1.3 SERVICE LOCATOR

O Service Locator é descrito por Fowler (2004) como um padrão que centraliza a resolução de dependências em um único ponto, permitindo a obtenção dinâmica de serviços. No projeto, ele é empregado por meio de um serviço personalizado, responsável por localizar e entregar instâncias de serviços conforme a necessidade.

```php
public function getServiceInstance(EnumServiceType $serviceType, string $entity): IEntityService
{
    $service = $this->getService($serviceType, $entity);
    try {
        /** @var IEntityService $instance */
        $instance = $this->container->get($service);
        return $instance;
    } catch (ServiceNotFoundException $e) {
        throw new RuntimeException("O serviço '$service' não é acessível pelo container");
    }
}
```

Figura 4 – Método do *Service Locator*

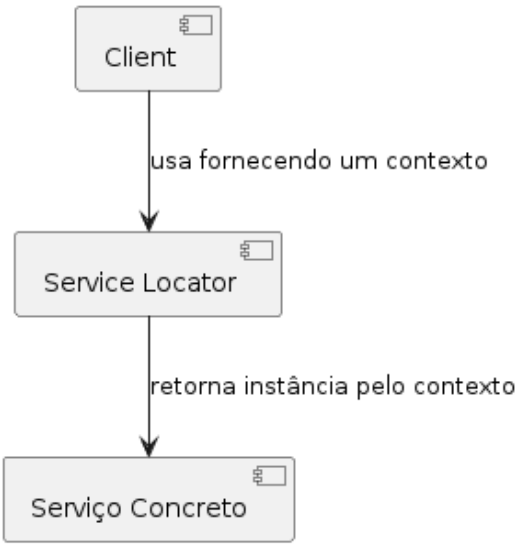

Figura 5 – Diagrama genérico do *Service Locator*

O padrão Service Locator, aplicado no Symfony, é apresentado pelo uso de um service locator, ou seja, um serviço que contém e resolve outros serviços sob demanda. Conforme a documentação oficial, podemos definir um localizador de serviços, e injetar esse localizador em serviços que necessitem escolher dinamicamente entre vários outros serviços. (SYMFONY, 2025).

Esse recurso oferece flexibilidade, mas exige uso criterioso para não gerar dependências ocultas. No sistema, ele permite abstrair responsabilidades e facilita a manutenção de múltiplos serviços relacionados, além de, no contexto do projeto, ter sido implementado de forma a integrar-se ao principal mecanismo de gerenciamento de serviços, o *design pattern* anterior, a fim de facilitar a resolução de instâncias sem provocar acoplamento.

3.1.4 STRATEGY

O padrão Strategy, conforme descrito por *Gamma et al.* (1995), define uma família de algoritmos, encapsulando cada um deles de forma que possam ser intercambiáveis em tempo de execução. Essa abordagem permite variar o comportamento de uma funcionalidade sem modificar sua estrutura interna, promovendo alta flexibilidade.

No projeto desenvolvido, o padrão Strategy foi aplicado na classe *EntityArgResolver*, que implementa o conceito de *Argument Resolver* do framework Symfony. De acordo com a documentação oficial do Symfony (SYMFONY, 2025), um *Argument Resolver* é responsável por determinar como os parâmetros de um método do *controller* devem ser construídos a partir de uma requisição HTTP.

No caso específico deste projeto, o *EntityArgResolver* utiliza uma estratégia dinâmica para decidir como o argumento será resolvido, de acordo com o modo definido pelo atributo *EntityArgProvider* presente no controlador.

Três estratégias distintas são possíveis dentro do contexto do projeto, sendo:

- *Desserialize* converte o corpo da requisição JSON em uma instância da entidade correspondente, utilizando o serviço de serialização do Symfony;
- *Query* obtém o identificador da entidade na requisição e realiza a consulta ao banco de dados, retornando o objeto completo;
- *Merge* combina as duas estratégias anteriores: busca o objeto no banco de dados e mescla seus dados com os valores enviados no corpo da requisição.

```
yield match ($argProvider->getMode()) {
    EnumArgProviderMode::Deserialize => $this->deserialize($request, $class),
    EnumArgProviderMode::Query       => $this->query($request, $class),
    EnumArgProviderMode::Merge       => $this->merge($request, $class),
};
```

Figura 6 – Resolução de entidades na requisição ao *controller*

Como mostrado no trecho de código da Figura 6, essa estrutura reflete o conceito fundamental do padrão Strategy, onde o contexto *EntityArgResolver* delega o processamento para diferentes estratégias de resolução, que são selecionadas em tempo de execução com base no cenário da requisição.

A flexibilidade dessa abordagem permite que novas estratégias sejam adicionadas futuramente sem modificar a lógica central, garantindo maior extensibilidade e mantendo o baixo acoplamento entre as camadas de controle e domínio. A Figura 7 apresenta uma visão genérica da estrutura do padrão *Strategy*.

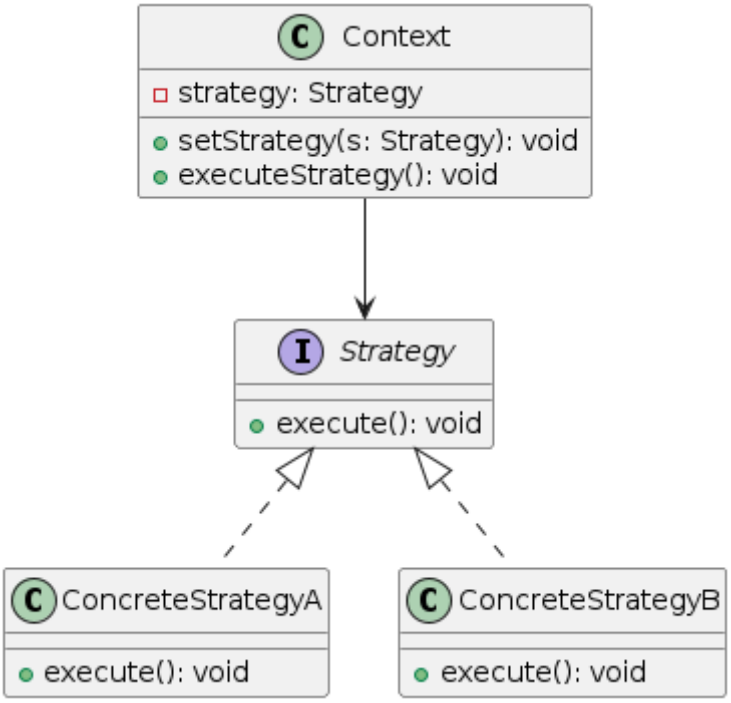

Figura 7 – Abordagem em classes do padrão *strategy*

3.1.5 OBSERVER

O padrão Observer, descrito por Gamma et al. (1995), define uma relação de dependência um-para-muitos entre objetos, de forma que quando um objeto muda de estado, todos os seus dependentes são notificados e atualizados automaticamente. Esse padrão é amplamente utilizado em arquiteturas orientadas a eventos, pois favorece o baixo acoplamento e a extensibilidade do sistema.

No projeto, o padrão é aplicado por meio dos *event subscribers* do Symfony, que permitem reagir a eventos do ciclo de vida da aplicação. No contexto do trabalho, um exemplo é o *KernelControllerListener*, que se inscreve para escutar eventos disparados pelo *Kernel* do *framework*, possibilitando que diferentes componentes respondam a ações sem depender diretamente do controlador. Nas Figuras 8 e 9 vemos um diagrama genérico de como o Symfony estrutura esse padrão, voltado para a parte de emissão de eventos, e um trecho de código da classe *KernelControllerListener*. (SYMFONY, 2025).

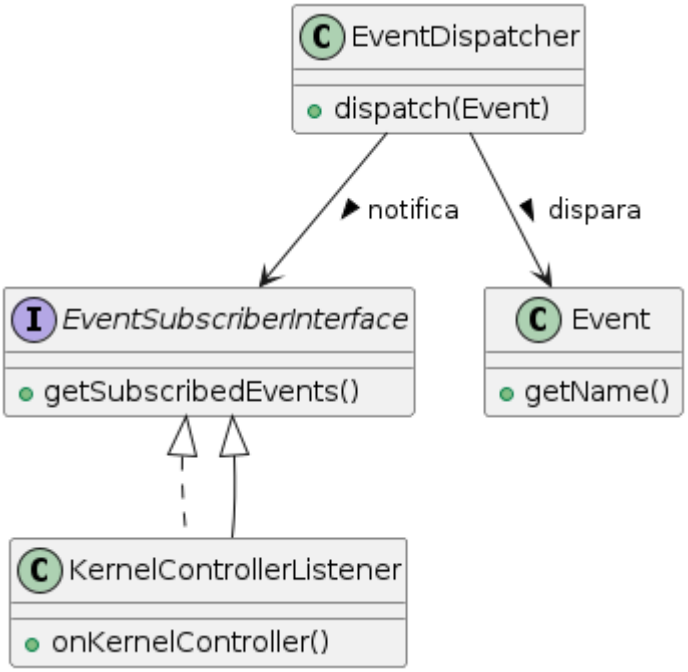

Figura 8 – *Observer* no *Symfony*

```php
public static function getSubscribedEvents(): array
{
    return [
        KernelEvents::CONTROLLER => 'onKernelController'
    ];
}
```

Figura 9 – Método obrigatório no *Symfony* para classes desse *pattern*.

Dentro do contexto da aplicação, a classe *KernelControllerListener* diz para o *framework* que possui um método *onKernelController* que é executado para eventos do tipo *controller*. Durante esse processo é verificada a permissão de determinado usuário, quando um controlador é resolvido para uma URL da requisição HTTP atual. Assim, essa abordagem garante que novas funcionalidades possam ser adicionadas ao sistema com base na ocasião de eventos, reforçando o princípio de aberto/fechado e promovendo maior modularidade na aplicação.

3.2 INTEGRANDO COM TLS

A segurança da comunicação entre cliente e servidor é um requisito essencial para aplicações modernas, sobretudo quando envolvem dados sensíveis como credenciais e informações financeiras. Nesse contexto, a integração do sistema proposto com o protocolo TLS desempenha papel central na proteção das transações realizadas.

Segundo Rescorla (2018), o funcionamento do TLS baseia-se em um processo chamado Handshake, uma sequência de etapas na qual cliente e servidor negociam parâmetros de segurança — daí o nome, que em inglês significa “aperto de mão”. O cliente solicita a conexão segura, o servidor responde com seu certificado digital emitido por uma autoridade certificadora, demonstrando a autenticidade da sua identidade, e ambos acordam o algoritmo de criptografia e as chaves de sessão a serem utilizadas para proteger todo o tráfego subsequente. Dessa forma, mesmo em redes públicas, a comunicação permanece confiável e protegida.

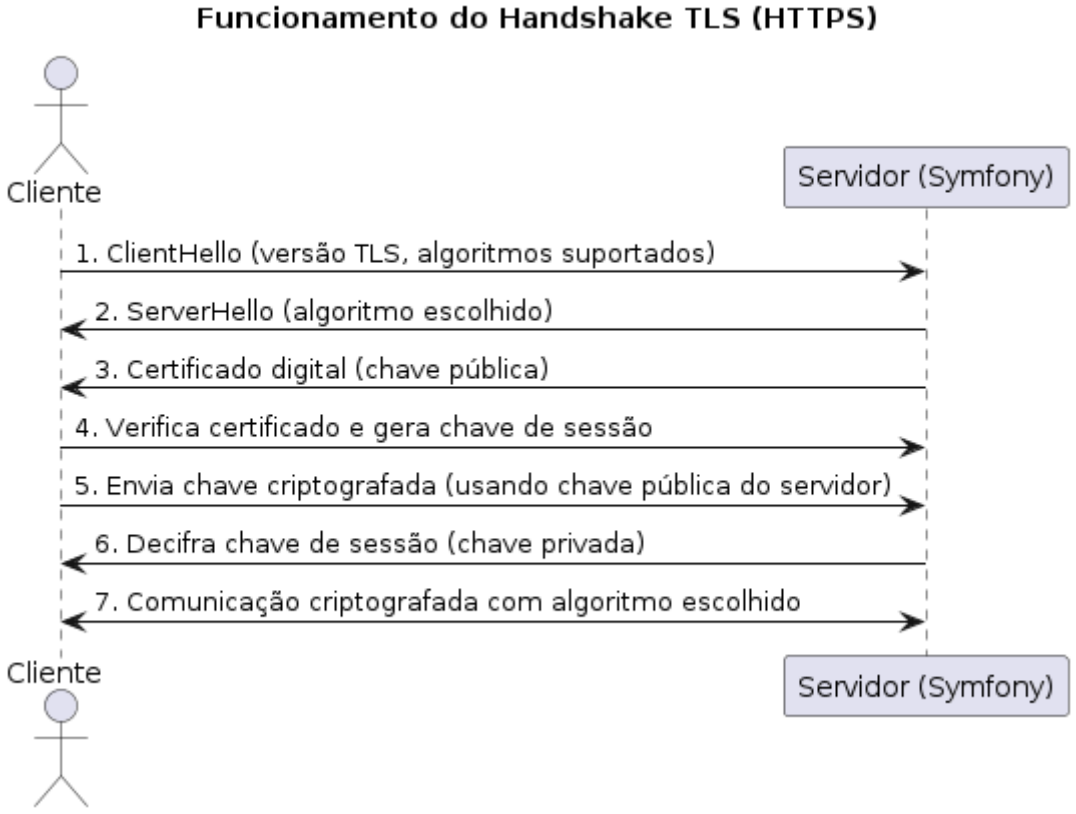

Figura 10 – Funcionamento do *Handshake* TLS (HTTPS). Baseado em Rescorla (2018)

A Figura 10 ilustra o funcionamento simplificado do Handshake TLS, destacando as etapas de negociação entre cliente e servidor durante o estabelecimento de uma sessão segura. Por meio desse processo, as partes envolvidas validam a autenticidade dos certificados, geram as chaves de sessão e definem o algoritmo criptográfico que será utilizado para cifrar os dados transmitidos.

Por sua vez, o uso do *Hypertext Transfer Protocol Secure* (HTTPS) consiste na implementação do *Hypertext Transfer Protocol* (HTTP) sobre o TLS, resultando em uma camada adicional de segurança que protege o tráfego entre o cliente e o servidor. Conforme documentação oficial da Mozilla (2023), o HTTPS proporciona três garantias fundamentais: a autenticidade do servidor por meio de certificados digitais válidos; a integridade dos dados, assegurando que as mensagens não sejam alteradas durante o trânsito; e a confidencialidade, ao cifrar todo o conteúdo trafegado na sessão.

No contexto do *Symfony*, esse recurso pode ser explorado de maneira prática, visto que o framework oferece suporte nativo à configuração de certificados digitais para execução do servidor em HTTPS. Nas figuras 11 e 12 vemos como com um comando de terminal, é possível instalar automaticamente um certificado de desenvolvimento e executar a API em um ambiente seguro, simulando condições próximas a um cenário de produção. Essa funcionalidade integra a preocupação com a segurança diretamente ao ciclo de desenvolvimento, evidenciando como frameworks modernos não apenas aceleram a construção de sistemas, mas também incorporam mecanismos que fortalecem a confiabilidade das aplicações. (SYMFONY, 2025).

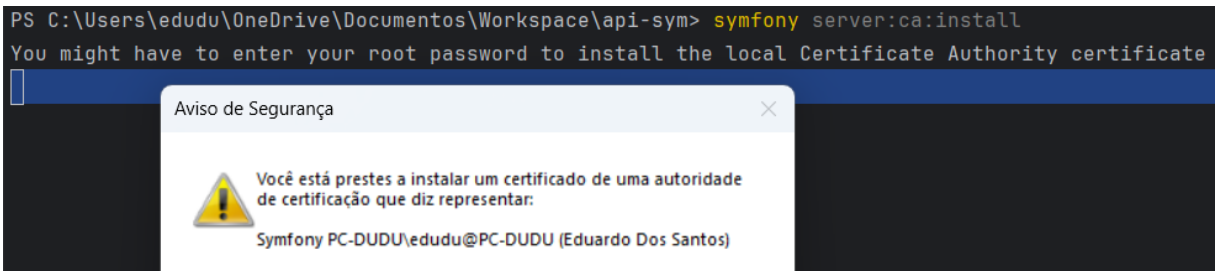

Figura 11 – Comando para instalação do certificado TLS

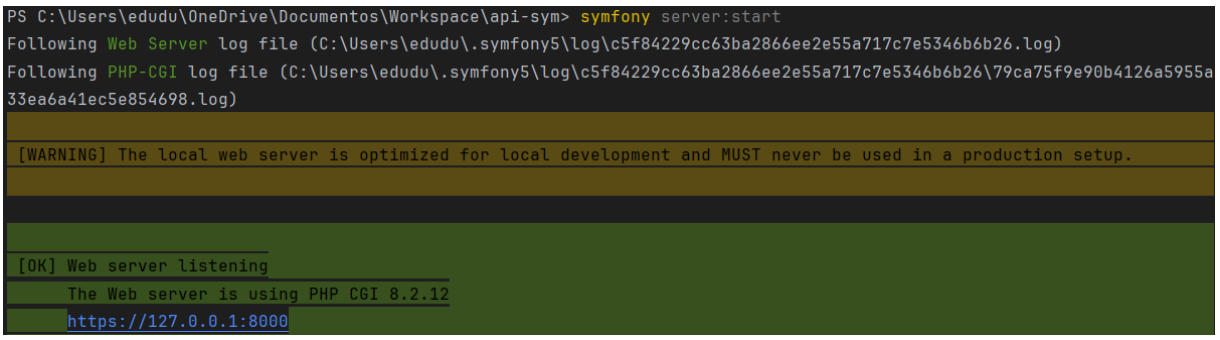

Figura 12 – Servidor rodando localmente com HTTPS

4 CONCLUSÃO E CONSIDERAÇÕES FINAIS

O presente relatório técnico teve como objetivo demonstrar, de forma prática e fundamentada, como a aplicação estruturada de padrões de projeto e o uso do protocolo Transport Layer Security (TLS) contribuem para o desenvolvimento de sistemas web modernos, seguros e escaláveis. Por meio do desenvolvimento de uma API REST utilizando o framework Symfony, foi possível exemplificar a integração entre conceitos teóricos da engenharia de software e sua aplicação efetiva em um ambiente real de desenvolvimento.

A implementação dos padrões de projeto como *Repository*, *Dependency Injection*, *Service Locator*, *Strategy*, *Observer* e outros, evidenciou a importância de uma arquitetura bem definida e de um código modular e reutilizável. Cada padrão aplicado contribuiu para aspectos específicos da qualidade do software, como desacoplamento, testabilidade, clareza estrutural e facilidade de manutenção, demonstrando que o uso consciente dessas práticas impacta diretamente a eficiência e a escalabilidade da aplicação.

Da mesma forma, a integração do sistema com o protocolo TLS mostrou-se essencial para garantir a segurança da comunicação entre cliente e servidor, assegurando a confidencialidade, integridade e autenticidade dos dados transmitidos. A configuração prática do HTTPS no ambiente de desenvolvimento do Symfony reforçou a viabilidade de implementar padrões de segurança desde as fases iniciais do ciclo de vida do software, alinhando teoria e prática de forma eficiente.

Conclui-se, portanto, que o uso combinado de padrões de projeto e mecanismos de segurança em rede permite alcançar resultados expressivos na criação de sistemas robustos e profissionais, atendendo aos requisitos técnicos e de qualidade esperados na indústria de desenvolvimento web. A aplicação dos princípios estudados neste trabalho demonstra que, mesmo em projetos de pequena escala, é possível empregar soluções arquiteturais maduras que aumentam a confiabilidade e a escalabilidade do produto final.

Como trabalhos futuros, sugere-se a ampliação da API para incluir mecanismos de monitoramento de desempenho, testes automatizados mais abrangentes, e a possível integração com serviços baseados em inteligência artificial, de modo a otimizar processos de análise de dados e recomendação. Essas melhorias podem estender ainda mais o potencial técnico do sistema e reforçar a relevância prática das boas práticas de engenharia de software no contexto de aplicações web modernas.

REFERÊNCIAS

EL BOUSSAIDI, Ghizlane; MILI, Hafedh. *Understanding design patterns — what is the problem? Software: Practice & Experience*, v. 42, n. 12, p. 1495-1529, 2011. DOI: https://doi.org/10.1002/spe.1145.

FOWLER, Martin. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003.

FOWLER, Martin. *Inversion of Control Containers and the Dependency Injection Pattern*. 2004. Disponível em: https://martinfowler.com/articles/injection.html. Acesso em: 25 set. 2025.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley, 1995.

IBM. *O que é uma API (Interface de Programação de Aplicações?)* 2024. Disponível em: https://www.ibm.com/think/topics/api. Acesso em: 8 set. 2025.

LOGESWARAMOORTHY, Tharshayini. *How Design Patterns improve software quality and maintainability. Medium*, 2020. Disponível em: https://medium.com/@tharsha9956/design-patterns-949e23b23f0d. Acesso em: 25 set. 2025.

MOZILLA. *Introduction to HTTPS. Mozilla Developer Network (MDN)*, 2023. Disponível em: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview#https. Acesso em: 14 out. 2025.

PRESSMAN, Roger S.; MAXIM, Bruce R. *Engenharia de software: uma abordagem profissional.* 8. ed. Porto Alegre: AMGH, 2016.

RESCORLA, Eric. *The Transport Layer Security (TLS) Protocol Version 1.3.* RFC 8446, 2018. Disponível em: https://www.rfc-editor.org/rfc/rfc8446. Acesso em: 22 set. 2025.

SOMMERVILLE, Ian. *Engenharia de Software.* 10. ed. São Paulo: Pearson, 2019.

SYMFONY. *Symfony Documentation.* 2024. Disponível em: https://symfony.com/doc. Acesso em: 3 out. 2025.

SYMFONY. *Service Container (Dependency Injection). Symfony Docs*, 2025. Disponível em: https://symfony.com/doc/current/service_container.html. Acesso em: 30 out. 2025.

SYMFONY. *Service Subscribers & Locators. Symfony Docs*, 2025. Disponível em: https://symfony.com/doc/6.3/service_container/service_subscribers_locators.html. Acesso em: 30 out. 2025.

SYMFONY. *Extending Action Argument Resolving. Symfony Docs*, 2025. Disponível em: https://symfony.com/doc/7.1/controller/value_resolver.html. Acesso em: 30 out. 2025.

SYMFONY. *Events and Event Listeners. Symfony Docs*, 2025. Disponível em: https://symfony.com/doc/current/event_dispatcher.html. Acesso em: 30 out. 2025.

SYMFONY. *Symfony Local Web Server – Enabling TLS. Symfony Docs*, 2025. Disponível em: https://symfony.com/doc/current/setup/symfony_cli.html#enabling-https-tls. Acesso em: 30 out. 2025.