**IPV – Instituto Politécnico de Viseu**

**ESTGV – Escola Superior de Tecnologia e Gestão de Viseu**

**Departamento de Informática**

****

**Relatório de Trabalho Prático Final**

**Licenciatura em Engenharia Informática**

**Realizado em**

**Engenharia do Software II**

**Por**

**Lucas Sebastião – pv2**

**João Santos - estgv**

**Daniel Valpereiro – pv22894**

**Miguel Marques – pv23914**

**Orientadores**

**Entidade: Instituto Politécnico de Viseu**

**ESTGV: Carlos Cunha**

**Pedro Martins**

**Viseu, 2024**

Índice

[1 Introdução 1](#_Toc126516724)

[1.1 Estrutura 1](#_Toc126516725)

[2 Atividades desenvolvidas 3](#_Toc126516726)

[2.1 Diagrama de classes 3](#_Toc126516727)

[2.2 Classes 4](#_Toc126516728)

[2.3 Sensor de humidade 6](#_Toc126516729)

[2.4 Lista dos motores mais fracos 6](#_Toc126516730)

[2.5 Função Estou Quente 7](#_Toc126516731)

[2.6 Manutenção 8](#_Toc126516732)

[3 Conclusão 9](#_Toc126516733)

[4 Referências Bibliográfica 11](#_Toc126516734)

# Introdução

Este trabalho aborda dois projetos distintos, cada um focado em diferentes aspetos da arquitetura de software e segurança. O primeiro projeto envolve a criação de uma aplicação de e-learning composta por vários módulos especializados. Utilizando padrões de design de software, como Singleton, Factory, Adapter, Composite, Flyweight, Memento e Decorator. O segundo projeto foca na criação de uma API REST em Node.js, com endpoints para gerenciamento de utilizadores e pedidos. Seguindo as melhores práticas de segurança da OWASP, a API implementa controles rigorosos de autenticação e autorização, como o princípio de Least Privileges, Deny by Default, e ReBAC.

Ambos os projetos têm como objetivo proporcionar uma base sólida para o desenvolvimento de aplicações seguras, eficientes e escaláveis, aplicando conceitos avançados de design de software e práticas de segurança.

# Épico 1

## Criação do Módulo de Configuração Única (singleton)

O padrão Singleton é um padrão de design de software que garante que uma classe tenha apenas uma única instância durante a execução do programa. Ele também fornece um ponto de acesso global a essa instância. A implementação é feita desta maneira:

. **Construtor Privado**: Impede a criação de instâncias da classe fora da sua própria área, garantindo que a única forma de obter uma instância da classe seja através de um método controlado.

. **Instância Estática**: Um membro estático da classe armazena a única instância da classe.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente. **Método de Acesso Público**: Um método público estático (getInstance) fornece acesso à instância única, criando-a se ainda não existir.

**Vantagens**: Controlo sobre o acesso à instância única facilita a gestão de recursos compartilhados. Redução do uso de memória, apenas uma instância é criada e usada durante a execução do programa.

**Desvantagens**: Testabilidade, singletons podem dificultar a escrita de testes unitários, pois a instância única pode manter o estado entre testes. Concorrência, a implementação padrão de Singleton não é thread-safe. Em ambientes multithreaded, é necessário garantir que o acesso à instância seja seguro.

## Criação dos Módulos de Criação de Objetos e Abstração de Plataformas (Factory Method)

O módulo de Criação de Objetos é implementado utilizando o padrão de design Factory, que permite a criação de diferentes tipos de objetos educacionais sem expor a lógica de criação ao cliente. Este padrão facilita a adição de novos tipos de objetos no futuro, mantendo o código flexível e fácil de manter. A implementação do código é feita deste método:

**. Interface de Plataforma**: A interface ObjetoFabrica<T> define o método createObjeto, que recebe um tipo de conteúdo e retorna um objeto desse tipo.

Uma imagem com texto, captura de ecrã, software, ecrã

Descrição gerada automaticamente. **Fabrica de conteúdos:** A classe ConteudoFabrica implementa a interface ObjetoFabrica e define a lógica de criação para diferentes tipos de conteúdos.

**Vantagens**: Encapsulamento da Criação de Objetos, a lógica de criação de objetos é centralizada na fábrica, o que simplifica o código cliente e mantém a lógica de criação separada. Facilidade de extensão, novos tipos de objetos podem ser adicionados sem modificar o código existente. Basta criar novas classes de objetos e atualizar a fábrica.

**Desvantagens:** Complexidade,introduz uma camada extra de abstração, que pode adicionar complexidade ao design do sistema.

# Conclusão

Concluindo, para a realização de uma gestão para uma fabrica é necessário esquematizar o problema, e de seguida construir as classes. Por fim, implementa-se os métodos. Desta forma, fica mais simples realizar o solicitado para este projeto.

Também pode-se mencionar que o projeto foi apenas uma simulação, e que pode ser utilizado para outros tipos de fins do mesmo tipo, e também poderá ser utilizado para outras fabricas para outras empresas.

# Referências Bibliográfica