**PROGAMA ORIENTADO AO OBEJETOS II**

1. **Introdução**

Dentro da matéria Programação Orientada a Objetos II, ela nos desafia a montar um projeto com o objetivo de desenvolver um sistema de desktop robusto, utilizando a linguagem C# e no ambiente de desenvolvimento (Visual Studio 2022), com a persistência de dados gerenciada pelo banco de dados SQL Server. Para garantir a organização e a manutenibilidade do código, a arquitetura da aplicação seguirá o padrão Model-View-Controller (MVC), separando claramente as responsabilidades entre a interface do usuário, as regras de negócio e o acesso aos dados. A comunicação com o back-end será realizada através de uma API dedicada, cuja documentação e testes serão facilitados pela ferramenta Swagger, garantindo a escalabilidade para futuras integrações. Adicionalmente, o sistema contará com a integração de um componente de Inteligência Artificial (IA), que atuará como uma ajuda aos técnicos que a ferramenta auxiliara nas resoluções dos problemas.

1. **Desktop**

Conceito de Aplicações Desktop.

Aplicações desktop, também chamadas aplicações de ambiente de trabalho, são software feito para instalar e rodar no sistema operativo do computador. Diferentemente das aplicações web, que rodam nos navegadores, as aplicações desktop usam os recursos locais do hardware processador, memória, e sistema de arquivos, diretamente por exemplo Fowler, 2003.

A ligação principal entre a aplicação desktop e o sistema operativo é crucial para o seu funcionamento. Sem dúvida, o sistema operativo fornece serviços usados pelas aplicações para interagir com o hardware, escondendo a complexidade dos componentes físicos Stallings, 2017. Ele age como uma camada de abstração e gestor de recursos, garantindo que os pedidos do software sejam atendidos de forma organizada e segura.

Como Tanenbaum e Bos destacaram lá em 2016, o sistema operativo age tipo a fundação pros programas rodarem, permitindo que o software funcione bem.

No mundo da engenharia de software, criar aplicativos para computadores exige muita atenção com a qualidade, já que eles têm acesso direto ao hardware do usuário. Pressman, em 2021, diz que o software é tipo um produto que precisa ser planejado direitinho pra ficar bom.

A Interface Gráfica do Usuário, tem um papel muito importante nesse caso. A facilidade de uso e a clareza na interação são o que fazem um aplicativo bom, por isso, um design focado no usuário é superimportante para a eficiência e intuição, como o Norman disse em 2013.

O sucesso de plataformas como o Windows já foi estudado por quem entende de interfaces complexas, mostrando o poder de uma benfeita, como o Petzold mostrou em 1998.

Regras internacionais definem as características de qualidade, tipo funcionalidade, confiança, usabilidade e desempenho, ajudando na criação de softwares complicados. Á norma ISOIEC 25010, criada pela ABNT em 2011, por exemplo.

**REFERÊNCIAS**

ANICHE, Mauricio. **Test-Driven Development: Teste e Design no Mundo Real com. NET**. Editora Casa do Código, 2014.

ARIAS, Ángel. **Aprende a Programar ASP. NET y C#: 2ª Edición**. IT campus academy, 2015.

DE DADOS, Componentes do Modelo. Projeto de Banco de Dados. 2000.

DEITEL, Harvey; DEITEL, Paul; DEITEL, Abbey. **Android: como programar**. Bookman Editora, 2015.

DIMES, Troy. **Programação em C# para iniciantes**. Babelcube Inc., 2016.

EVANS, Eric. **Domain-driven design: atacando as complexidades no coração do software**. Alta Books, 2009.

Faceli, Katti, et al. "Inteligência artificial: uma abordagem de aprendizado de máquina." (2021).

FOWLER, Martin. **Padrões de arquitetura de aplicações corporativas**. Bookman, 2009.

FOWLER, Susan J. **Microsserviços prontos para a produção: Construindo sistemas padronizados em uma organização de engenharia de software**. Novatec Editora, 2019.

GAMMA, Erich et al. Design patterns: Abstraction and reuse of object-oriented design. In: **European conference on object-oriented programming**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993. p. 406-431.

GAMMA, Erich. **Padrões de projetos: soluções reutilizáveis**. Bookman editora, 2009.

GERON, Aurelien. **Mãos à obra: aprendizado de maquina com Scikit-Learn, Keras & Tensorflow**. Alta Books, 2021.

GOODFELLOW, Ian et al. **Deep learning**. Cambridge: MIT press, 2016.

HEJLSBERG, Anders; WILTAMUTH, Scott; GOLDE, Peter. **C# language specification**. Addison-Wesley Longman Publishing Co., Inc., 2003.

JUNIOR, Edemilton Alcides Galindo; ROCHA, Romeu Dias; DE SOUZA MACIEL, Ronierison. Desenvolvimento de api rest com spring boot. 2021.

MARTIN, Robert C. **Arquitetura Limpa: o guia do artesão para estrutura e design de software**. Alta Books Editora, 2019.

MARTIN, Robert C. **Clean Code-Refactoring, Patterns, Testen und Techniken für sauberen Code: Deutsche Ausgabe**. MITP-Verlags GmbH & Co. KG, 2013.

MICHAELIS, Mark. **Essential C# 12.0**. Addison-Wesley Professional, 2023.

NEWMAN, Sam. **Criando Microsserviços–2a Edição: Projetando sistemas com componentes menores e mais especializados**. Novatec Editora, 2022.

RUSSELL, Stuart; NORVIG, Peter. Inteligência Artificial: uma abordagem moderna. **Campus,**, 2004.

TAULLI, Tom. **Introdução à inteligência artificial: uma abordagem não técnica**. Novatec Editora, 2020.