CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

PIERRE CAPISTRANO LOPES

título do trabalho

RIO DO SUL

2023

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

PIERRE CAPISTRANO LOPES

TÍTULO DO TRABALHO

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí, como condição parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Orientador: Xxxxxxxxxxx Xxxxxxxxxxxxx

RIO DO SUL

2023

CENTRO UNIVERSITÁRIO PARA O DESENVOLVIMENTO DO ALTO VALE DO ITAJAÍ - UNIDAVI

PIERRE CAPISTRANO LOPES

TÍTULO DO TRABALHO

Trabalho de Conclusão de Curso a ser apresentado ao curso de Sistemas da Informação, da Área das Ciências Naturais, da Computação e das Engenharias, do Centro Universitário para o Desenvolvimento do Alto Vale do Itajaí- UNIDAVI, a ser apreciado pela Banca Examinadora, formada por:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Professor Orientador: Xxxxxxx Xxxxxxxx

Banca Examinadora:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof.

Rio do Sul, xx de mês de 2023.

Epígrafe, Epígrafe, Epígrafe, Epígrafe, Epígrafe, Epígrafe, Epígrafe, Epígrafe, Epígrafe, Epígrafe, Epígrafe, Epígrafe (Autor).

Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória Dedicatória.

AGRADECIMENTOS

Agradecimentos agradecimentos agradecimentos agradecimentos agradecimentos agradecimentos agradecimentos agradecimentos agradecimentos agradecimentos agradecimentos agradecimentos.

RESUMO

Elemento obrigatório, que consiste na apresentação concisa dos pontos relevantes de um texto. O resumo deve dar uma visão rápida e clara do conteúdo e das conclusões do trabalho. Deve conter entre 150 a 500 palavras.

**Palavras-Chave:** palavra 1, palavra 2, palavra 3.

ABSTRACT

Versão do resumo em inglês (também pode ser em espanhol).

**Keywords:** keyword 1, keyword 2, keyword 3.

LISTA DE FIGURAS

Figura 1- Título Figura 21

Figura 2- Título Figura 21

Figura 3- Título Figura 21

Figura 4- Título Figura 21

Figura 5- Título Figura 21

LISTA DE QUADROS

Quadro 1 - Título Quadro 21

Quadro 2 - Título Quadro 21

Quadro 3 - Título Quadro 21

Quadro 4 - Título Quadro 21

Quadro 5 - Título Quadro 21

LISTA DE ABREVIATURAS E SIGLAS

MA Modelo de Abreviatura

**SUMÁRIO**

(a contagem inicia na folha de rosto, mas a paginação inicia apenas na Introdução)

1. INTRODUÇÃO 11

1.1 OBJETIVOS 12

1.2.1 Geral 12

1.2.2 Específicos 12

1.3 JUSTIFICATIVA 12

1. introdução

Nos últimos anos, o Alto Vale do Itajaí tem sido uma região de destaque no setor agrícola, com a agricultura familiar desempenhando um papel fundamental em seu desenvolvimento econômico. Consequentemente, os escritórios de projetos agrícolas têm ganhado cada vez mais importância nesse contexto, à medida que oferecem suporte e serviços essenciais aos agricultores locais. No entanto, o crescimento desses escritórios demanda um gerenciamento eficiente, assim como ocorre em qualquer empresa em expansão. É neste ponto que a tecnologia pode desempenhar um papel crucial, oferecendo soluções para aprimorar e otimizar esse processo de gerenciamento.

Os escritórios de projetos agrícolas desempenham um papel vital ao oferecer serviços diversos, como financiamento agrícola, investimentos, análises de terras, seguro de safra, entre outros, que são essenciais para os agricultores da região. No entanto, a prestação desses serviços requer o gerenciamento eficiente de uma ampla gama de informações. Caso essas informações não sejam devidamente administradas, podem ocorrer perdas de dados, ineficiência nos processos e, em casos extremos, problemas judiciais.

Diante dessa necessidade de gerenciamento mais eficiente, surge um problema evidente: a ausência de sistemas específicos no mercado para o gerenciamento de escritórios de projetos agrícolas. Atualmente, existem apenas sistemas ERP (*Enterprise Resource Planning*) genéricos, que atendem a diferentes tipos de empresas, além de sistemas voltados para o gerenciamento de projetos de forma mais geral. Contudo, um escritório de projetos agrícolas possui requisitos específicos que vão além do escopo dessas soluções existentes.

Logo, há uma demanda latente por um sistema de gerenciamento de escritórios de projetos agrícolas que possua as ferramentas adequadas para gerenciar projetos, suas documentações, atividades e progresso. Além disso, é essencial contar com funcionalidades para controle de clientes, gestão financeira, bem como a geração de gráficos e relatórios contendo indicadores relevantes para o acompanhamento do desempenho desses escritórios.

Diante desse contexto, este trabalho de conclusão de curso tem como objetivo principal o desenvolvimento de um protótipo de sistema para o gerenciamento de escritórios de projetos agrícolas. O sistema proposto visa suprir a lacuna existente no mercado, fornecendo uma solução personalizada e eficiente para atender às necessidades específicas desses escritórios. Por meio deste estudo, pretende-se explorar as funcionalidades essenciais e os benefícios potenciais que um sistema desse tipo pode oferecer, contribuindo assim para o aprimoramento e profissionalização dos escritórios de projetos agrícolas na região do Alto Vale do Itajaí.

**Tema: Protótipo de sistema para gerenciamento de escritórios de projetos agrícolas.**

1.1 PROBLEMA DE PESQUISA

Qual a melhor forma de gerenciar escritórios de projetos agrícolas?

1.2 Objetivos

1.2.1 Geral

* Desenvolver um protótipo de sistema para gerenciamento de escritórios de projetos agrícolas.

1.2.2 Específicos

* Levantar requisitos com escritório modelo;
* Criar regras de negócio;
* Definir tecnologias para desenvolvimento (Linguagens, BD, frameworks);
* Prototipar sistema para gerenciamento de escritórios agrícolas;
* Testar protótipo.

1.3 Justificativa

O Alto vale do Itajaí é uma região onde a agricultura representa grande parte do PIB, principalmente a agricultura familiar, e dessa forma os escritórios de projetos agrícolas crescem cada vez mais, necessitando de um gerenciamento maior como qualquer empresa em crescimento, assim a tecnologia poderá auxiliar esse processo.

Um escritório de projetos agrícolas presta serviços para agricultores que necessitam de financiamento agrícola, investimento, análises de terras, seguro de safra, etc. E todos esses serviços que são fornecidos necessitam de uma gama de informações que, caso não gerenciadas corretamente, podem ocasionar perda de dados, ineficiência dos processos e no pior dos casos problemas judiciais.

A partir dessa necessidade surge um problema, não existe no mercado sistemas específicos para o gerenciamento de escritórios desse tipo, existem somente ERPs (sistemas para gerenciar todos os tipos de empresas) e sistemas para gerenciamento de projetos.

Um sistema de para escritórios de projetos agrícolas necessita de ferramentas para gerenciar os projetos, bem como suas documentações, atividades, andamento, etc. Necessita de um controle de clientes, financeiro, além de gráficos e relatórios com os indicadores dos escritórios.

1.4 CONTEXTUALIZAÇÃO DA EMPRESA

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto. texto texto texto texto texto texto. texto texto texto texto texto texto. texto texto texto texto texto texto. texto texto texto texto texto texto. texto texto texto texto texto texto. texto texto texto texto texto texto. texto texto texto texto texto texto. texto texto texto texto texto texto. texto texto texto texto texto texto. texto texto texto texto texto texto. texto texto texto texto texto texto. texto texto texto texto texto texto. texto texto texto texto texto texto.

2. REFERENCIAL TEÓRICO

2.1 banco de dados

De acordo com Alves (2014), um banco de dados é um conjunto de valores e arquivos que se relacionam entre si e demonstram um cadastro de pessoas, vendas, produtos, agendas, etc. Por exemplo, uma planilha ou uma ficha cadastral podem ser exemplos de cadastros. O banco de dados guarda todos esses cadastros em formato virtual e os disponibiliza para as aplicações consultarem e emitirem relatórios, realizarem vendas, etc.

2.1.1 Banco de dados relacional

De acordo com Alves (2014, p. 20), Banco de dados relacional “se caracteriza pelo fato de organizar os dados em tabelas (ou relações), formadas por linhas e colunas. Assim, essas tabelas são similares a conjuntos de elementos ou objetos, uma vez que relacionam as informações referentes a um mesmo assunto de modo organizado.”

Conforme Silva (2021), bancos de dados relacionais são os mais utilizados no mercado atual, mesmo que tenha surgido a partir de modelos de 1970. Eles são conhecidos por serem sistemas de propósito geral, utilizarem a linguagem SQL (Structured Query Language; ou Linguagem de Consulta Estruturada, em português) para consultar e manipular os dados, além das propriedades ACID para transações, conforme apresentado no Quadro 1.

**Quadro 1 - Propriedades ACID**

|  |  |
| --- | --- |
| **Atomicidade** | As transações são atômicas, executam completamente ou não executam. |
| **Consistência** | As transações criam novos estados válidos, ou, em caso de falha, o banco de dados volta para o último estado válido. |
| **Isolamento** | Garante que uma transação em andamento não será interferida por outras transações. |
| **Durabilidade** | Garante que uma transação em andamento não será interferida por outras transações. |

Fonte: Elaborado a partir de Pritchett (2008)

Ainda de acordo com Silva (2021, p. 14):

Outra característica marcante dos bancos de dados relacionais é a integridade referencial, que garante a acurácia e a consistência dos dados dentro de um relacionamento entre tabelas. Isso é feito por meio de uma chave estrangeira, que faz referência a um valor de uma chave primária em outra tabela, e a integridade referencial garantirá que esse relacionamento é íntegro (o registro referenciado existe)

O Dr. Edgard F. Codd formulou os princípios básicos do sistema de banco de dados relacional em 1968, baseando-se na teoria dos conjuntos e da álgebra relacional. Segundo ele, certos conceitos da matemática podiam ser aplicados ao gerenciamento de bancos de dados, provavelmente por ter sido um brilhante matemático. Segundo CODD (1985), existem um conjunto de doze regras para que um banco de dados relacional fosse admitido como tal:

* Regra de informações;
* Regra de acesso garantido;
* Tratamento de valores nulos;
* Catálogo relacional ativo;
* Inserção, exclusão e alteração em bloco;
* Linguagem de manipulação de dados abrangente;
* Independência física dos dados;
* Independência logica dos dados;
* Regra de atualização de visões;
* Independência de integridade;
* Independência de distribuição;
* Regra não subversiva.

2.1.2 Banco de dados não relacional

Conforme Sadamoto et al. (2012), Bancos de dados não relacionais, também conhecidos como NoSQL, são sistemas de gerenciamento de bancos de dados que não seguem o modelo relacional tradicional. Ao contrário dos bancos de dados relacionais, que armazenam informações em tabelas com linhas e colunas, os bancos de dados não relacionais utilizam uma variedade de modelos de dados, como documentos, grafos, chave-valor e colunas amplas, para armazenar informações de maneira mais flexível e escalável. As principais características dos bancos de dados não relacionais incluem alta escalabilidade horizontal, flexibilidade no esquema de dados, suporte a dados semiestruturados e não estruturados, e alta disponibilidade. Além disso, muitos bancos de dados não relacionais são projetados para funcionar em ambientes distribuídos e podem fornecer recursos de replicação e particionamento para melhorar a capacidade de lidar com grandes volumes de dados.

Bancos de dados não relacionais são adequados para cenários em que a escalabilidade, a flexibilidade e a agilidade são mais importantes do que a consistência transacional ou a conformidade com um modelo de dados estrito, conforme afirmado por Sadamoto et al. (2012). Esses autores ainda destacam que os bancos de dados NoSQL são particularmente adequados para o armazenamento de grandes volumes de dados não estruturados ou semi-estruturados, como documentos, redes sociais, registros de log, entre outros. Além disso, eles podem fornecer recursos de replicação e particionamento para melhorar a capacidade de lidar com grandes volumes de dados e suportar ambientes distribuídos. Por fim, os bancos de dados não relacionais também são úteis em aplicações de análise de dados, processamento de big data, Internet das Coisas (IoT) e aplicações de busca, onde a flexibilidade do esquema de dados é mais importante do que a garantia de integridade referencial.

2.2 programação orientada a objetos

Programação Orientada a Objetos (POO) é um paradigma de programação que se concentra em representar entidades do mundo real como objetos, que possuem atributos e métodos que definem seu comportamento e funcionalidade. O objetivo da POO é facilitar a modelagem e a resolução de problemas complexos, dividindo-os em partes menores e mais gerenciáveis. Nesse paradigma, as entidades são representadas por classes, que são usadas para criar objetos. A POO é baseada em quatro conceitos principais: encapsulamento, herança, polimorfismo e abstração. O encapsulamento é a capacidade de esconder os detalhes internos de um objeto e protegê-lo contra acesso indevido. A herança permite que as classes herdem atributos e métodos de outras classes e estendam seu comportamento. O polimorfismo permite que os objetos sejam tratados de maneiras diferentes, dependendo do contexto em que são usados. A abstração é a capacidade de extrair as características essenciais de um objeto e criar uma classe abstrata que define a interface para essas características. Juntos, esses conceitos permitem que os desenvolvedores criem programas orientados a objetos que são modulares, flexíveis e fáceis de manter (BARNES; KÖLLING, 2012).

Conforme Farinelli (2007), A Programação Orientada a Objetos “consiste em conceber um sistema computacional como um todo orgânico formado por objetos que se relacionam entre si. Esse enfoque pode ser aplicado tanto à análise de sistemas quanto à programação, e essa é uma das principais vantagens da orientação a objetos”, portanto a mesma metodologia serve tanto para a definição lógica do sistema quanto para a sua implementação.

Em análise de sistemas, uma questão antiga é a definição de dados e processos, pois existem metodologias que definem as estruturas de dados primeiros, e após os processos que a utilizam, mas também existem metodologias que fazem o contrário, definindo primeiro os procedimentos que serão automatizados e posteriormente os dados que deverão ser utilizados. Porém como também definido por Farinelli (2007, p. 31):

dados e processos são apenas componentes, e o enfoque está em identificar quais os objetos que interagem entre si no sistema. Os dados são identificados procurando os atributos que definem os objetos, e os procedimentos pelas operações que estes objetos realizam. A interação entre os objetos é definida pelas estruturas e relacionamentos que são identificados. O resultado é que em um modelo orientado a objetos, existe total coerência entre os dados e os processos, mesmo quando há muitas pessoas trabalhando no mesmo sistema.

AAAAAAAAAAAAAAAAAAAAAA NÃO PODE

2.2.1 Linguagens orientadas a objeto

Abaixo está listado as principais linguagens que são utilizadas para Programação Orientada a Objetos:

**Quadro 1 – Linguagens de Programação Orientadas a Objetos**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Linguagem | Herança | Polimorfismo | Encapsulamento | Abstração | Classe Abstrata | Interface | Construtores |
| **C#** | Sim | Sim | Sim | Sim | Sim | Sim | Sim |
| **Delphi** | Sim | Sim | Sim | Sim | Sim | Sim | Sim |
| **JavaScript** | Não | Sim | Parcial | Sim | Não | Sim | Sim |
| **PHP** | Sim | Sim | Sim | Sim | Sim | Sim | Sim |
| **Java** | Sim | Sim | Sim | Sim | Sim | Sim | Sim |

Fonte: Elaborado a partir da Documentação das Linguagens (2023)

C# (pronuncia-se "C sharp") é uma linguagem de programação moderna e orientada a objetos, desenvolvida pela Microsoft como parte da plataforma .NET. O diferencial do C# está na sua performance e eficiência, além da integração com outras tecnologias da Microsoft, como o .NET Framework e o Visual Studio. O C# apresenta uma sintaxe elegante e simples, além de recursos avançados de programação orientada a objetos, como encapsulamento, herança e polimorfismo. Um dos principais problemas enfrentados pelo C# atualmente é a sua limitação em relação à plataforma Windows. Embora a linguagem seja amplamente utilizada no desenvolvimento de aplicações para desktop e mobile na plataforma .NET, sua utilização em outros sistemas operacionais ainda é limitada. Além disso, algumas características do C# podem ser consideradas complexas e difíceis de entender para desenvolvedores iniciantes. (MICROSOFT, 2021).

Delphi é uma linguagem de programação orientada a objetos, originalmente criada pela Borland e atualmente mantida pela Embarcadero Technologies. O diferencial do Delphi está na sua produtividade e simplicidade, além da facilidade de uso e da grande comunidade de desenvolvedores que a utilizam. A linguagem Delphi oferece recursos avançados de programação orientada a objetos, incluindo herança, polimorfismo e interfaces, além de recursos adicionais, como componentes visuais e bibliotecas de terceiros. Um dos principais problemas enfrentados pelo Delphi é a sua limitação em relação à plataforma Windows. Embora a linguagem seja amplamente utilizada no desenvolvimento de aplicações para desktop e mobile, sua utilização em outros sistemas operacionais ainda é limitada. Além disso, a linguagem Delphi apresenta uma curva de aprendizado relativamente longa e pode ser considerada menos popular do que outras linguagens de programação. (EMBARCADERO, 2021).

JavaScript é uma linguagem de programação de script orientada a objetos, amplamente utilizada para desenvolvimento web. O diferencial do JavaScript está na sua capacidade de criar interatividade e dinamismo em páginas web, além da sua flexibilidade e compatibilidade com outros frameworks e bibliotecas. O JavaScript apresenta recursos avançados de programação orientada a objetos, como encapsulamento, herança prototipal e polimorfismo. Um dos principais problemas enfrentados pelo JavaScript atualmente é a segurança. Como a linguagem é executada no lado do cliente, é possível que hackers explorem vulnerabilidades na programação JavaScript para invadir sistemas ou roubar informações confidenciais. Além disso, a grande quantidade de bibliotecas e frameworks disponíveis para o JavaScript pode tornar o desenvolvimento complexo e difícil de gerenciar. (MOZILLA DEVELOPER NETWORK, 2021).

PHP é uma linguagem de programação orientada a objetos, amplamente utilizada para desenvolvimento web do lado do servidor. O diferencial do PHP está na sua facilidade de aprendizado e uso, além da grande quantidade de bibliotecas e extensões disponíveis para sua utilização. O PHP apresenta recursos avançados de programação orientada a objetos, como encapsulamento, herança, polimorfismo e interfaces, além de recursos adicionais, como bibliotecas de terceiros e extensões. Um dos principais problemas enfrentados pelo PHP é a sua segurança. Como a linguagem é amplamente utilizada no desenvolvimento web do lado do servidor, é possível que hackers explorem vulnerabilidades na programação PHP para invadir sistemas ou roubar informações confidenciais. Além disso, a linguagem PHP apresenta uma curva de aprendizado relativamente longa e pode ser considerada menos elegante e sofisticada do que outras linguagens de programação. (PHP, 2021).

Java é uma linguagem de programação orientada a objetos, desenvolvida pela Sun Microsystems e atualmente mantida pela Oracle. O diferencial do Java está na sua portabilidade, segurança e escalabilidade, além da grande quantidade de frameworks e bibliotecas disponíveis para sua utilização. O Java apresenta uma sintaxe simples e clara, além de recursos avançados de programação orientada a objetos, como encapsulamento, herança e polimorfismo. Além disso, a linguagem Java apresenta recursos adicionais, como segurança e portabilidade. Um dos principais problemas enfrentados pelo Java atualmente é a sua performance. Embora a linguagem seja amplamente utilizada em sistemas empresariais, sua performance pode ser considerada inferior a outras linguagens de programação, como C++ e Rust. Além disso, a utilização excessiva de frameworks e bibliotecas pode tornar o desenvolvimento complexo e difícil de gerenciar. (ORACLE, 2021).

2.3 Web services

Web Services são conjuntos de programas que podem ser publicados, buscados e chamados por meio da internet. Esses programas podem realizar um simples processo de troca de mensagens, como também complexas transações comerciais ou industriais, por exemplo, um processo de compra de produtos. Uma vez que um Web Service é publicado em um servidor web, vários programas e até mesmo outros Web Services podem acessá-lo e chamá-lo, tanto para obtenção de dados como para a interação com serviços que uma organização oferece (TAMAE, 2004).

API

HTTP

REST

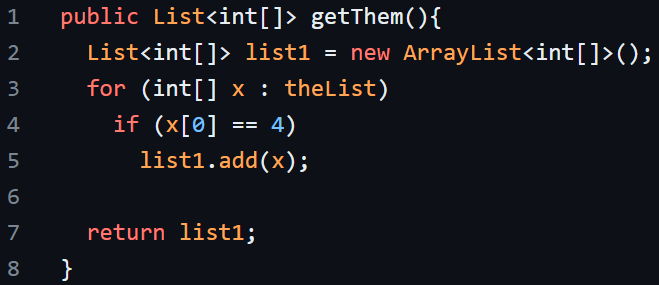
2.4 BOAS PRATICAS DE DESENVOLVIMENTO

Conforme Martin (2009, p. 8), “Um código limpo é simples e direto. Ele é tão bem legível quanto uma prosa bem escrita. Ele jamais torna confuso o objetivo do desenvolvedor, em vez disso, ele está repleto de abstrações claras e linhas de controle objetivas”.

Ainda conforme Martin, um código confuso diminui a produtividade da equipe, cada nova alteração cause problemas em outras duas ou três partes, exigindo remendos, amarrações para que o código não quebre e com o tempo, a bagunça cresce tanto que é impossível solucionar. Com a redução da produtividade, a gerência contrata novos funcionários para aumentar a mão de obra, mas os novos programadores não conhecem o código por completo, não conseguindo diferenciar a mudança que altera o propósito do projeto e a aquela que atrapalha. Criando mais e mais confusões, a produtividade chega cada vez mais perto de zero.

2.4.1 Nomeação

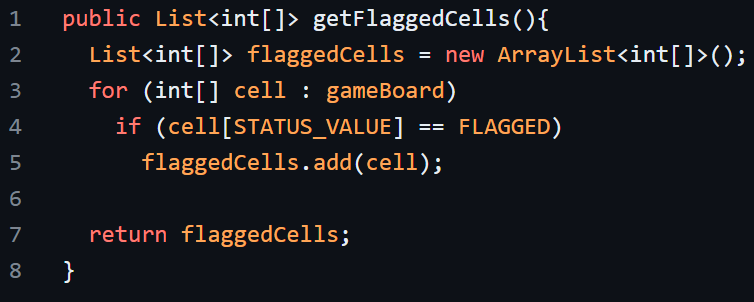
O início de boas práticas de desenvolvimento se dá pela nomeação de variáveis, métodos e classes, pois o nome deve dizer porque existe, o que faz e como é usado. (MARTIN, 2009). A Figura 1 mostra um código que não utiliza uma boa nomeação de suas variáveis, funções, etc.

**Figura 1 – Código sem boas práticas de nomeação**  


**Fonte: Martin (2009, p. 18)**

Esse código é simples, porém seu contexto não está implícito, dificultando o entendimento. A nomeação seguindo as boas práticas garante que em qualquer parte de um código, o seu contexto estará implícito, facilitando a leitura conforme a Figura 2.

**Figura 1 – Código sem boas práticas de nomeação**



**Fonte: Martin (2009, p. 19)**

Além de nomes que dão o contexto do código, é necessário utilizar nomes pronunciáveis. Como uma variável do tipo Date nomeada de “genymdhms” por exemplo, ela pode contextualizar a sua utilização, mas a mesma variável nomeada de “generationTimestamp” facilita muito mais o entendimento, além de melhorar a comunicação da equipe de desenvolvimento. (MARTIN, 2009)

Martin (2009) também salienta a utilização de nomes passíveis de busca, auxiliando na procura de determinadas variáveis, constantes e métodos por todo o projeto, pois utilizar o número 7 como uma constante é comum, mas buscar por todo o código pode demorar por ser um número que pode aparecer em diversos contextos, mas utilizar uma constante com nome MAX\_CLASSES\_PER\_STUDENT torna a pesquisa muito mais rápida.

2.4.2 Funções

A primeira regra para funções é que elas devem ser pequenas. Conforme Martin (2019, p. 34):

Não tenho referências de pesquisas que mostrem que funções muito pequenas são melhores. Só posso dizer que por cerca de quatro décadas tenho criado funções de tamanhos variados. Já escrevi diversos monstros de 3000 linhas; bastantes funções de 100 a 300 linhas; e funções que tinham apenas 20 a 30 linhas. Essa experiência me ensinou que, ao longo de muitas tentativas e erros, as funções devem ser muito pequenas.

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

2.4 DESENVOLVIMENTO AGIL

2.4.1 TESTES AUTOMATIZADOS

2.6. REACTJS

Roldán (2021) afirma que o React é uma biblioteca JavaScript de código aberto e adaptável com o foco na construção de interfaces de usuário complexas a partir de pequenos códigos separados chamados componentes. O React efetivamente torna as aplicações mais flexíveis, fáceis de manter e com um melhor desempenho, dando ao seu fluxo de trabalho um grande impulso, melhorando a velocidade sem afetar a qualidade.

React (2022) declara que o objetivo do ReactJS é o desenvolvimento de UIs interativas de forma fácil e que a estrutura das interfaces se atualize e renderizem na medida que os dados forem sendo alterados de maneira automatizada. Afirmam também que o ReactJS é baseado com componentes encapsulados que podem ser facilmente combinados para interfaces mais complexas e com partes em comum de forma reutilizável.

3. Metodologia DA PESQUISA

O presente trabalho caracteriza-se como pesquisa aplicada e descritiva, pois foi desenvolvido um protótipo da aplicação e também teve como um dos objetivos verificar se o protótipo desenvolvido tem as funcionalidades necessárias para possibilitar o uso comercial.

O trabalho buscou responder o seguinte problema: É possível desenvolver um sistema para otimizar os processos de um escritório de projetos agrícolas?

Após a finalização do levantamento bibliográfico, foi realizado um estudo mais aprofundado sobre o desenvolvimento de aplicações com arquitetura de camadas em conjunto com a arquitetura de microsserviços, utilizando as ferramentas e padrões de desenvolvimentos mais modernos possíveis, visando deixar a aplicação relevante por mais tempo, evitando futuros retrabalhos, migrações para novos padrões e facilitando possíveis adições futuras ao trabalho.

Com base nessa pesquisa foi definido que a aplicação seria feita na IDE Visual Studio, com a linguagem C#, integrando com o banco de dados SQL Server nativamente e design baseado no conjunto de boas práticas definidos pela Microsoft na documentação mais atualizada das tecnologias. Essas escolhas foram feitas por serem sugeridas pela Google no ano de 2020, portanto a manutenção da tecnologia será constante.

4. RESULTADOS DO TC (recomenda-se usar o título do tc aqui)

5. CONCLUSÃO

**REFERÊNCIAS**

TAMAE, André. **Desenvolvimento de Aplicações Web com JSP, Servlets, JavaServer Faces, Hibernate, Ajax e Web Services**. São Paulo: Érica, 2004.

FERREIRA, Arthur Gonçalves**. Interface de programação de aplicações (API) e web services.** São Paulo Platos Soluções Educacionais 2021 1 recurso online ISBN 9786553560338.

SILVA, Luiz F. Calaça et al. **Banco de dados não relacional**. Porto Alegre SAGAH 2021 1 recurso online ISBN 9786556901534.

ALVES, William P. **Banco de Dados**. São Paulo Editora Saraiva, 2014. E-book. ISBN 9788536518961. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788536518961/. Acesso em: 24 mar. 2023.

Pritchett, D. (2008). **BASE: An Acid Alternative**. ACM Queue, 6(3), 48-55. Disponível em: https://dl.acm.org/doi/pdf/10.1145/1394127.1394128. Acesso em: 24 mar. 2023.

DIAS, Ariel da S. **Administração de bancos de dados**. São Paulo Editora Saraiva, 2021. E-book. ISBN 9786589965688. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9786589965688/. Acesso em: 24 mar. 2023.

CODD, Edgar Frank. (1985). **Is Your DBMS Really Relational?** Computerworld, 19(30), 29-33. Disponível em: https://www.analytictech.com/borgatti/whatEver/readingsem/Codd.pdf. Acesso em: 24 mar. 2023.

SADAMOTO, A. M. A. et al. **NoSQL databases: an overview**. In: Proceedings of the 2012 Brazilian Symposium on Computing Systems Engineering (SBESC). IEEE, 2012.

BARNES, D. J.; KÖLLING, M. **Programação Orientada a Objetos com Java**. 3. ed. São Paulo: Pearson, 2012

FARINELLI, Fernanda. **Conceitos básicos de programação orientada a objetos**. Instituto Federal Sudeste de Minas Gerais, 2007.

Microsoft. **C# language reference**. Disponível em: https://docs.microsoft.com/pt-br/dotnet/csharp/. Acesso em: 12 maio 2023.

Embarcadero. **RAD Studio Documentation**. Disponível em: https://docwiki.embarcadero.com/RADStudio/Sydney/en/Main\_Page. Acesso em: 12 maio 2023.

Mozilla. **Javascript**. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript. Acesso em: 12 maio 2023.

PHP. **PHP: Hypertext Preprocessor**. Disponível em: https://www.php.net/manual/pt\_BR/. Acesso em: 12 maio 2023.

Oracle. **Java SE Documentation**. Disponível em: https://docs.oracle.com/en/java/javase/14/. Acesso em: 12 maio 2023.

MARTIN, Robert C. **Código limpo: habilidades práticas do Agile software**. Rio de Janeiro Alta Books, 2009.

DIAS, Ariel da S. **Administração de bancos de dados**. São Paulo Editora Saraiva, 2021. E-book. ISBN 9786589965688. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9786589965688/. Acesso em: 24 mar. 2023.

SILVA, Luiz F. Calaça et al. **Banco de dados não relacional**. Porto Alegre SAGAH 2021 1 recurso online ISBN 9786556901534.

ALVES, William P. **Banco de Dados**. São Paulo Editora Saraiva, 2014. E-book. ISBN 9788536518961. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788536518961/. Acesso em: 24 mar. 2023.

ROLDÁN, Carlos Santana. **React 17:** design patterns and best practices. 3. ed. Birmingham: Packt, 2021. Ebook.

REACT. **Uma biblioteca Javascript para criar interfaces de usuário.** Disponível em: <https://pt-br.react.dev/learn>. Acesso em 22 junho. 2023.

**ANEXOS**