**FUNDAÇÃO OSWALDO ARANHA**

**CENTRO UNIVERSITÁRIO DE VOLTA REDONDA**

**CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO**

**TRABALHO DE PROGRAMAÇÃO ORIENTADO A OBJETOS II**

**PROJETO EVENTFLOW**

**Pedro Amando Gandos Citelli-** **0009-0003-4285-0142**

**Pedro Lucas Reis de Oliveira Sousa - 0009-0001-7369-1768**

**Pablo Irineu de Souza - 0009-0004-1545-7879**

**Paulo Ryan Garani Salgado - 0009-0004-5521-4134**

**Victor Hugo Varelo Hartung - 0009-0009-5273-7526**

**SISTEMA DE GERENCIAMENTO DE EVENTOS: Uma Aplicação de Fluxo Continuo e Eficiente sobre Eventos**

**VOLTA REDONDA**

**2024**

**FUNDAÇÃO OSWALDO ARANHA**

**CENTRO UNIVERSITÁRIO DE VOLTA REDONDA**

**CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO**

**TRABALHO DE PROGRAMAÇÃO ORIENTADA A OBJETOS II**

**PROJETO EVENTFLOW**

**SISTEMA DE GERENCIAMENTO DE EVENTOS: Uma Aplicação de Fluxo Continuo e Eficiente sobre Eventos**

Trabalho apresentado na disciplina de Programação Orientada a Objetos II do Curso de Sistemas de Informação do Centro Universitário de Volta Redonda.

Alunos:

Pedro Amando Gandos Citelli

Pedro Lucas Reis de Oliveira Sousa

Pablo Irineu de Souza

Paulo Ryan Garani Salgado

Victor Hugo Varelo Hartung

Orientador:

Prof. Rosenclever Gazoni

**VOLTA REDONDA**

**2024**

# RESUMO

Este trabalho tem como objetivo desenvolver um sistema de gerenciamento de eventos, otimizando a organização e a eficiência operacional através de funcionalidades que atendam às necessidades de organizadores e participantes. O projeto será estruturado em torno de quatro entidades principais: Evento, Participante, Inscrição e Organizador, permitindo o gerenciamento completo de eventos, desde a criação e inscrição de participantes até a consulta automática de endereços utilizando a API ViaCep. O sistema visa alinhar as soluções propostas às necessidades reais do negócio e dos usuários.

Palavras-Chave: Gerenciamento de eventos; Organização; Eficiência; ViaCep.

**SUMÁRIO**

[RESUMO 3](#_Toc182225677)

[1.0 INTRODUÇÃO 8](#_Toc182225678)

[2.0 ESTRUTURA E PLANEJAMENTO 11](#_Toc182225679)

[3.0 REFERENCIAL TEÓRICO 13](#_Toc182225680)

[3.1 Apresentando a Programação Orientada a Objetos 13](#_Toc182225681)

[3.1.1 Descrição Do Código 17](#_Toc182225682)

[3.2 Contextualizando a Modelagem UML 18](#_Toc182225683)

[3.3 Compreendendo o Conceito e a Aplicação de APIs 20](#_Toc182225684)

[4.0 Análise de Requisitos 21](#_Toc182225685)

[4.1 Levantamento e Especificação 21](#_Toc182225686)

[4.2 Identificação de Entidades 21](#_Toc182225687)

[4.3 Relacionamento entre Entidades 22](#_Toc182225688)

[5.0 Metodologia 23](#_Toc182225689)

[5.1 Abordagem de Desenvolvimento 23](#_Toc182225690)

[5.2 Etapas do Desenvolvimento 23](#_Toc182225691)

[5.3 Ferramentas e Tecnologias Utilizadas 23](#_Toc182225692)

[5.4 Justificativa da Metodologia 23](#_Toc182225693)

[6.0 Aplicação 24](#_Toc182225694)

[6.1 Estrutura do Código 24](#_Toc182225695)

[6.2 Demonstração de Uso 28](#_Toc182225696)

[7. CONSIDERAÇÕES FINAIS 29](#_Toc182225697)

[8. NOTAS DE PARTICIPAÇÃO DOS COMPONENTES DA EQUIPE 30](#_Toc182225698)

**LISTA DE FIGURAS**

[Figura 1 – Logo EventFlow](#_Toc42806369) 8

[Figura 2 - Código de Encapsulamento 1](#_Toc42806370)4

[Figura 3 - Código de Herança 15](#_Toc42806371)

[Figura 4 - Código de Polimorfismo e Abstração 15](#_Toc42806372)

[Figura 5 - Código Principal 1](#_Toc42806373)6

Figura 6 - Entidade Inscrição 24

Figura 7 - Relacionamento entre entidades 25

Figura 8 - Integração da API 26

Figura 9 - Estrutura Main 27

Figura 10 - Vantagens da manutenção voltada para disponibilidade e confiabilidade 27

**LISTA DE DIAGRAMAS**

Diagrama 1 – Classe UML 19

**LISTA DE SIGLAS**

|  |  |
| --- | --- |
| PjBL | *Project Based Learning* (Aprendizado Baseado em Projetos). |
| UML | *Unified Modeling Language* (Linguagem de Modelagem Unificada). |
| UniFOA | Centro Universitário de Volta Redonda. |

# 

# 1.0 INTRODUÇÃO

A rede de eventos “Event Flow” é uma empresa de organização de eventos com sede na cidade de Volta Redonda - RJ. Ela atende uma ampla gama de clientes, oferecendo serviços que abrangem desde a idealização e planejamento até a execução e acompanhamento de eventos. A eficiência logística e a personalização das experiências são pilares fundamentais para garantir a satisfação de seus clientes e participantes.

O sistema de gestão da Event Flow foi desenvolvido com o intuito de otimizar o gerenciamento de eventos, permitindo maior controle sobre inscrições, consulta de endereços para eventos por meio da API ViaCep e comunicação com os participantes. Essas funcionalidades visam proporcionar uma experiência mais integrada e organizada, alinhando as operações internas às necessidades de organizadores e usuários.

Na execução de seus projetos, a Event Flow adota metodologias inovadoras, como o Aprendizado Baseado em Projetos (PjBL), que busca integrar soluções tecnológicas às demandas reais do mercado. Estudos mostram que metodologias como o PjBL contribuem significativamente para a inovação e eficiência, permitindo à empresa consolidar-se como referência no setor de eventos.

Logotipo, nome da empresa

Descrição gerada automaticamente

Figura-1 – Logo EventFlow

Fonte: Elaborado pelos autores

**Objetivo Geral**

Tornar eficaz o gerenciamento de eventos, proporcionando uma experiência integrada e organizada para organizadores e participantes, com foco em eficiência operacional e personalização.

* 1. **Objetivos Específicos**

Visando atender ao objetivo geral, o projeto foca nas seguintes funcionalidades essenciais do sistema de gerenciamento de eventos da Event Flow:

* Desenvolver um Sistema de Gerenciamento de Eventos que integre informações de inscrições, participantes e organizadores;
* Implementar controle automatizado para monitorar inscrições e status de pagamento;
* Fornecer dados de localização em tempo real através de integração com API ViaCep;
* Facilitar a comunicação entre organizadores e participantes para melhorar a experiência dos usuários;
* Permitir a gestão centralizada de múltiplos eventos, otimizando a logística e o planejamento.
  1. **Motivação do Projeto**

Desenvolver conhecimentos na disciplina de Programação Orientada a Objetos II, aplicando conceitos avançados de POO no desenvolvimento de um sistema funcional e eficiente. O objetivo é propor uma solução tecnológica que atenda às necessidades específicas do mercado de eventos, utilizando boas práticas de programação para criar um sistema robusto, escalável e alinhado às expectativas dos usuários e organizadores.

* 1. **Resultados Esperados**

Melhoria nos processos de organização e gestão de eventos, com redução de falhas operacionais e aumento da eficiência e satisfação dos participantes. Controle eficaz da localização e das inscrições, garantindo uma experiência mais profissional e personalizada, além de fortalecer a posição da Event Flow como líder no mercado de eventos.

# 2.0 ESTRUTURA E PLANEJAMENTO

Para a construção do Sistema de Gerenciamento de Eventos da Event Flow, foi adotada uma abordagem estruturada com base nos conceitos de Engenharia de Software e Programação Orientada a Objetos II. O projeto utiliza a *UML* (*Unified Modeling Language*) como ferramenta principal para modelagem do sistema, com foco no Diagrama de Classes para representar a estrutura e os relacionamentos entre as entidades principais do sistema.

O desenvolvimento foi dividido em etapas, descritas a seguir:

**Referencial Teórico**

* Apresentando a Programação Orientada a Objetos: introdução aos conceitos fundamentais e sua importância para o projeto;
* Contextualizando a Modelagem UML: explicação sobre a aplicação da UML no desenvolvimento de sistemas;
* Compreendendo o Conceito e a Aplicação de APIs: definição e utilidade de APIs no contexto do sistema, destacando a integração para consultas de endereços via API ViaCep.

|  |
| --- |
| **Análise de Requisitos** |
| * Levantamento e Especificação: detalhamento dos requisitos funcionais e não funcionais do sistema para garantir que atenda às necessidades dos usuários; * Identificação de Entidades: definição das principais classes do sistema, incluindo seus atributos e métodos; * Relacionamentos entre Entidades: modelagem inicial das associações, garantindo a consistência e a funcionalidade do sistema. |
|  |

|  |
| --- |
| **Metodologia** |
| * Baseada nos conceitos de Programação Orientada a Objetos e modelagem com UML. * Dividida em etapas: análise de requisitos, modelagem, implementação e testes. * Utilização de ferramentas como IDEs, bibliotecas e a API ViaCep para consulta de endereços. * Foco na criação de um sistema funcional e bem estruturado, com validação contínua em cada etapa do desenvolvimento. |
| **Aplicação** |
| * Sistema projetado para gestão de eventos e inscrições. * Funcionalidades incluem controle de eventos, participantes e pagamentos. * Integração com a API ViaCep para consultas automáticas de endereços de eventos. * Benefícios incluem automatização, redução de falhas e melhor experiência para usuários e organizadores. |

# 3.0 REFERENCIAL TEÓRICO

## 3.1 Apresentando a Programação Orientada a Objetos

Segundo Robert C. Martin (2017), a Programação Orientada a Objetos (POO) é um paradigma que organiza o software em torno de dados, ou objetos, e não de funções e lógica. Essa abordagem facilita a manutenção e a escalabilidade dos sistemas, promovendo uma estrutura modular e reutilizável.

Os principais pilares da POO são:

* **Encapsulamento:** Consiste em ocultar os detalhes internos de um objeto, expondo apenas o que é necessário através de interfaces públicas. Isso protege os dados e reduz a complexidade do sistema.
* **Herança:** Permite que uma classe derive características (atributos e métodos) de outra, promovendo a reutilização de código e a criação de hierarquias que refletem relações do mundo real.
* **Polimorfismo:** Habilidade de diferentes classes responderem de maneira específica a uma mesma mensagem ou método, permitindo que objetos de diferentes tipos sejam tratados de forma uniforme.
* **Abstração:** Processo de simplificar a complexidade ao ocultar detalhes desnecessários e destacar características essenciais, facilitando o entendimento e a modelagem do sistema.

Texto

Descrição gerada automaticamenteAs figuras abaixo ilustram a aplicação prática dos conceitos fundamentais da Programação Orientada a Objetos, abordando encapsulamento, herança, polimorfismo e abstração. Esses pilares são representados em um exemplo de código simplificado, desenvolvido em C#, que demonstra como os princípios teóricos se traduzem em soluções reais no desenvolvimento de software.

Figura-2 - Código de Encapsulamento

Fonte: Fornecida pelos Autores.

Texto

Descrição gerada automaticamente

Figura-3 - Código de Herança

Fonte: Fornecida pelos Autores.

Texto

Descrição gerada automaticamente

Figura-4 - Código de Polimorfismo e Abstração

Fonte: Fornecida pelos Autores.

Texto

Descrição gerada automaticamente

Figura-5 - Código de Principal

Fonte: Fornecida pelos Autores.

## 3.1.1 Descrição Do Código

## 

O código apresentado ilustra a aplicação prática dos conceitos fundamentais da Programação Orientada a Objetos (POO), divididos entre encapsulamento, herança e polimorfismo.

Os principais pilares da POO são:

* **Encapsulamento:** Consiste em ocultar os detalhes internos de um objeto, expondo apenas o que é necessário através de interfaces públicas. Isso protege os dados e reduz a complexidade do sistema.
* **Encapsulamento**: A classe Evento utiliza atributos privados (nome e local), acessíveis apenas por meio de propriedades públicas (Nome e Local). Isso garante que a manipulação dos dados seja controlada, seguindo o princípio de proteção de informações.
* **Herança**: A classe EventoOnline herda a classe Evento, reutilizando atributos e métodos. Além disso, sobrescreve o método ExibirDetalhes para adicionar informações específicas de eventos online.
* **Polimorfismo**: É demonstrado com a classe abstrata Notificacao, que é implementada pelas classes EmailNotificacao e SMSNotificacao. Ambas redefinem o método Enviar, permitindo comportamentos específicos para cada tipo de notificação.
* **Execução no Código Principal**: No método Main, são criados objetos das classes Evento e EventoOnline, utilizando seus métodos para exibir detalhes. Também é demonstrado o uso do polimorfismo para enviar notificações por email e SMS.

## 3.2 Contextualizando a Modelagem UML

## 

A UML (Unified Modeling Language) é uma linguagem de modelagem amplamente utilizada na Engenharia de Software para representar graficamente a estrutura e o comportamento de sistemas. De acordo com Booch, Rumbaugh e Jacobson (2005), a UML é “uma linguagem padrão para especificação, visualização, construção e documentação de sistemas complexos”.

No projeto de Sistema de Gerenciamento de Eventos, utilizamos o **Diagrama de Classes**, uma das principais ferramentas da UML. Ele foi empregado para modelar as entidades principais do sistema e seus relacionamentos, fornecendo uma visão clara e detalhada da arquitetura do software.

A imagem a seguir ilustra o Diagrama de Classes desenvolvido para o projeto, destacando as principais entidades e seus relacionamentos, proporcionando uma visão clara e estruturada da arquitetura do sistema.

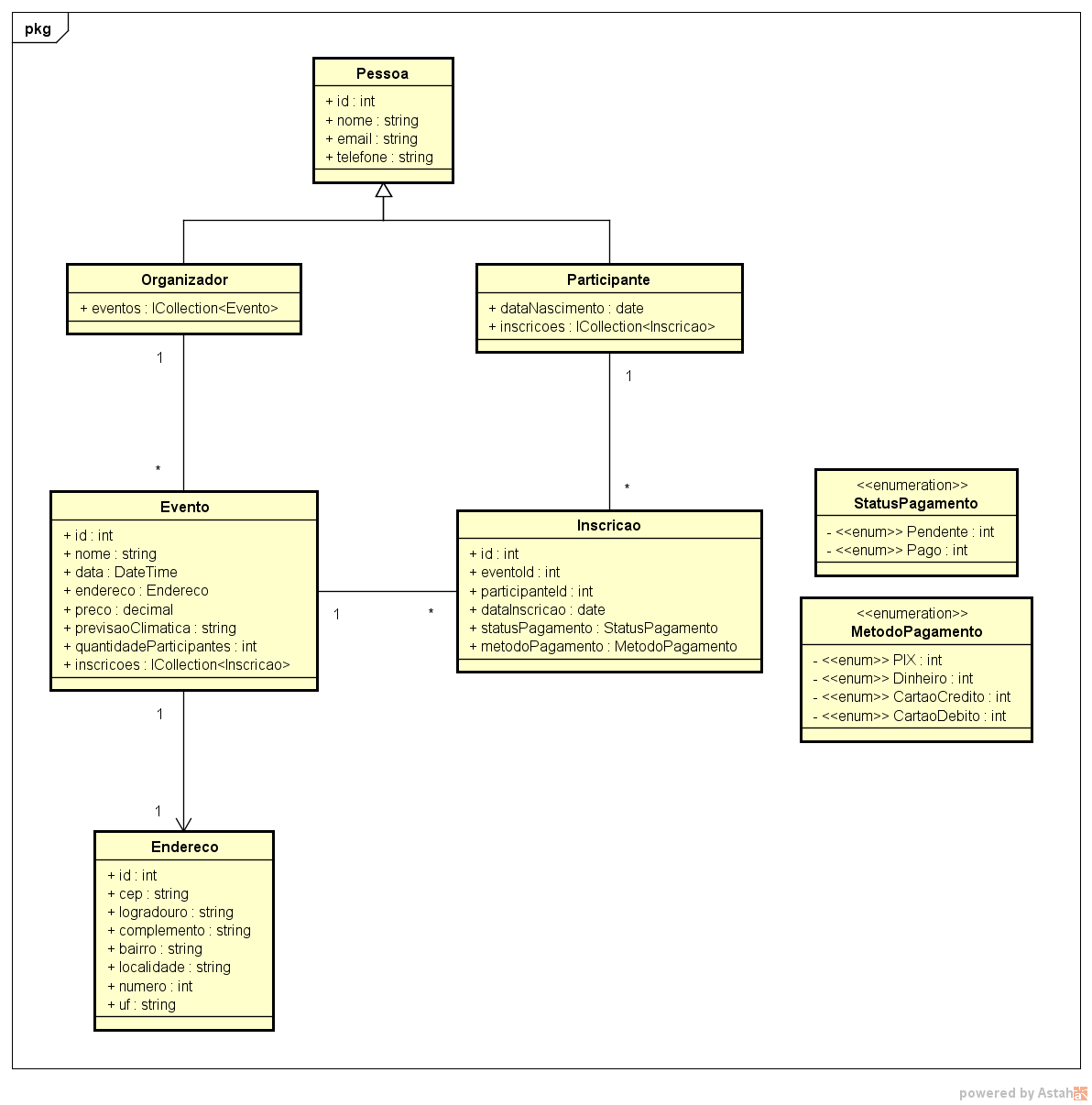


Diagrama 1–Diagrama UML

Fonte: Fornecida eplos autores .

Os principais componentes do Diagrama de Classes incluem:

* **Encapsulamento**
* **Classes**: Representam entidades do sistema, como Evento, Participante, e Organizador, cada uma com seus atributos e métodos.
* **Atributos** **e** **Métodos**: Definem as características e comportamentos de cada classe.
* **Relacionamentos**: Mostram como as classes interagem entre si, como herança, associação e agregação.

## 3.3 Compreendendo o Conceito e a Aplicação de APIs

## 

As APIs (Application Programming Interfaces) são protocolos que permitem a comunicação entre diferentes sistemas de software. Segundo Fielding (2000), APIs "fornecem interfaces que simplificam a interação entre serviços e aplicações, promovendo maior eficiência e modularidade no desenvolvimento".

No projeto de Sistema de Gerenciamento de Eventos, foi implementada uma API para consulta de endereço através do CEP, que exibe os dados do endereço (logradouro, bairro, cidade e estado). Essa funcionalidade permite que o usuário otimize seu tempo e facilite a descrição da localidade do evento.

As principais vantagens da utilização de APIs no sistema incluem:

* **Automatização:** Redução de tarefas manuais, como consulta de dados externos.
* **Escalabilidade:** Possibilidade de integrar novas funcionalidades de forma rápida e eficiente.
* **Precisão:** Acesso a dados atualizados em tempo real, garantindo informações confiáveis para os usuários.

Essa abordagem destaca a importância das APIs como ferramentas que agregam valor ao sistema, otimizando a operação e oferecendo recursos adicionais que atendem às necessidades do público-alvo.

# 4.0 Análise de Requisitos

## 4.1 Levantamento e Especificação

O levantamento e especificação de requisitos têm como objetivo identificar e documentar as funcionalidades que o sistema deve atender para satisfazer as necessidades dos usuários e os objetivos do projeto. No Sistema de Gerenciamento de Eventos, foram definidos:

**Requisitos Funcionais:**

* Permitir que organizadores criem e gerenciem eventos.
* Registrar participantes e gerenciar inscrições.
* Exibir informações meteorológicas em tempo real por meio de uma API de clima.

**Requisitos Não Funcionais:**

* Garantir segurança no acesso às informações do sistema.
* Prover uma interface amigável e de fácil uso.
* Assegurar a escalabilidade para suportar múltiplos eventos simultaneamente.

## 4.2 Identificação de Entidades

As entidades principais do Sistema de Gerenciamento de Eventos foram definidas com base nas necessidades do projeto e nas funcionalidades esperadas. Cada entidade representa um elemento essencial do sistema, conforme descrito abaixo:

* **Evento:** Armazena informações sobre cada evento, como nome, data, local e previsão do tempo.
* **Participante:** Representa os usuários interessados em se inscrever nos eventos, com dados como nome, email e telefone.
* **Organizador:** Gerencia a criação e manutenção de eventos, incluindo o controle de participantes.
* **Inscrição:** Associa participantes aos eventos, registrando a data de inscrição e o status do pagamento.

## 4.3 Relacionamento entre Entidades

Os relacionamentos entre as entidades foram definidos para garantir que o sistema funcione de maneira coesa e integrada:

* Organizador e Evento: Um organizador pode criar e gerenciar vários eventos.
* Evento e Inscrição: Um evento pode ter várias inscrições associadas.
* Participante e Inscrição: Um participante pode se inscrever em múltiplos eventos.

# 5.0 Metodologia

## 5.1 Abordagem de Desenvolvimento

O desenvolvimento do sistema foi organizado em sprints semanais. Na primeira semana, o grupo focou na organização e estruturação teórica do projeto, desenhando toda a parte conceitual. Isso garantiu que não houvesse dúvidas durante a implementação prática e permitiu que as próximas etapas fossem executadas de forma clara e eficiente.

## 5.2 Etapas do Desenvolvimento

O levantamento dos requisitos foi realizado em conjunto pelos membros do grupo, assegurando que todas as funcionalidades fossem bem definidas. O sistema foi modelado com o Diagrama de Classes utilizando o Astah Community, sendo essa a primeira etapa do projeto. O diagrama foi utilizado como um guia para a implementação, organizando e facilitando o processo de codificação. A implementação do código começou somente após a conclusão da modelagem.

## 5.3 Ferramentas e Tecnologias Utilizadas

Para o desenvolvimento, foram utilizadas as seguintes ferramentas e tecnologias:

* Linguagem de programação: C#;
* Ambiente de Desenvolvimento: Visual Studio;
* Integração: API aberta da ViaCep para consulta de endereços.

## 5.4 Justificativa da Metodologia

A metodologia foi escolhida por ser a mais adequada ao prazo disponível para a entrega do projeto. A divisão do trabalho em sprints, com uma etapa inicial dedicada à teoria e ao planejamento, garantiu uma execução organizada e alinhada aos objetivos do grupo. Essa abordagem permitiu que o desenvolvimento fosse realizado de forma ágil e eficiente, com base em um planejamento sólido.

# 6.0 Aplicação

## 6.1 Estrutura do Código

Nesta seção, apresentamos os principais trechos do código que compõem o sistema. Cada print é acompanhado por uma explicação breve do seu funcionamento e como ele contribui para o sistema.

Este trecho de código apresenta a entidade inscrição em eventos do sistema de gerenciamento do sistema. Organizando e padronizando os dados da inscrição, promovendo integridade e clareza na manipulação de informações do sistema.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Figura-6 – Entidade Inscrição

Fonte: Fornecida pelos Autores.

Este trecho de código apresenta a entidade Evento do sistema fazendo um relacionamento com a entidade Organizador através do atributo OrganizadorId e a entidade Inscrição através do atributo Inscrições estabelecendo um relacionamento um-para-muitos, onde cada evento pode conter várias inscrições.

Texto

Descrição gerada automaticamente

Figura-7 – Relacionamento entre entidades

Fonte: Fornecida pelos Autores.

Este trecho de código apresenta a classe ViaCepIntegration, responsável pela integração com a API do ViaCep. Através do método assíncrono GetViaCepData, que recebe um parâmetro de entrada (cep), a classe realiza uma requisição para obter dados de endereço associados ao CEP fornecido.

Texto

Descrição gerada automaticamente

Figura-8 – Integração da API

Fonte: Fornecida pelos Autores.

Este trecho de código configura a estrutura principal da aplicação. Inicialmente, são definidos os serviços principais, como controle de views, acesso ao banco de dados e integração com a API ViaCEP. Em seguida, o aplicativo é construído e configurado com middlewares essenciais, como redirecionamento HTTP, arquivos estáticos, roteamento e autorização. Assim, a rota padrão é definida.

Texto

Descrição gerada automaticamente

Figura-9 – Estrutura Main

Fonte: Fornecida pelos Autores.

## 6.2 Demonstração de Uso

Por fim, a aplicação será demonstrada com base em exemplos práticos, como:

* Criação de um novo evento e registro de participantes.
* Consulta de informações de endereço pela integração com a API ViaCep.
* Exibição de detalhes de eventos organizados.

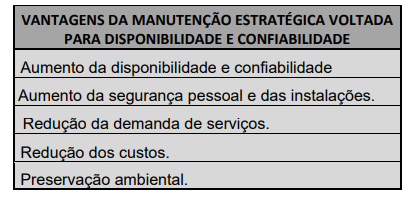


Figura-10 - Vantagens da manutenção voltada para disponibilidade e confiabilidade

Fonte: Villanueva (2015)

# 7. CONSIDERAÇÕES FINAIS

O trabalho desenvolvido buscou abordar a implementação de um sistema de gerenciamento para eventos, focando na organização eficiente, na automação dos processos de inscrição e no suporte às decisões por meio de integração com APIs, garantindo uma experiência satisfatória para organizadores e participantes.

A elaboração deste projeto foi guiada pela metodologia de sprints semanais, que permitiu um desenvolvimento estruturado e adaptável às necessidades identificadas ao longo do processo. A aplicação do Diagrama de Classes como base de modelagem assegurou uma visão clara e consistente do sistema, facilitando sua implementação.

A construção do projeto integrou técnicas de análise e ferramentas modernas de desenvolvimento, como o Astah Community para modelagem e o Visual Studio para codificação. O resultado é um sistema escalável e robusto, preparado para atender às demandas atuais de gerenciamento de eventos e pronto para incorporar futuras inovações.

# 8. NOTAS DE PARTICIPAÇÃO DOS COMPONENTES DA EQUIPE

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome** | **Autoavaliação** | **Avaliação Equipe** | **Avaliação Professor** |
| **Pedro Amando Gandos Citelli** |  |  |  |
| **Pedro Lucas Reis de Oliveira Sousa** |  |  |  |
| **Pablo Irineu de Souza** |  |  |  |
| **Paulo Ryan Garani Salgado** |  |  |  |
| **Victor Hugo Varelo Hartung** |  |  |  |

**REFERÊNCIAS BIBLIOGRÁFICAS**

ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. ABNT. NBR 5462: 1994. **Confiabilidade e Manutenabilidade.** Rio de Janeiro: ABNT, 1994.

BELL, D. **Fundamentos básicos de UML: O diagrama de classes, uma introdução aos diagramas de estrutura em UML 2.** IBM Corporation, 2016.

CYSNEIROS, L.M.; LEITE, J.C.S.P. **Definindo Requisitos Não Funcionais**. In: XI Simpósio Brasileiro de Engenharia de *Software*. Out, 1997.

DIIRR, B. **Projeto detalhado - Diagrama de classes (de projeto).** Disponível em; <http://www.ic.uff.br/~anselmo/cursos/ProjSoft/apresentacoes/Projeto%20detalhado%20-%20Classes.pdf >. Acesso em: 08 Nov. 2024.

FIGUEIREDO, E. **Requisitos Funcionais e Requisitos Não Funcionais.** Disponível em: <https://homepages.dcc.ufmg.br/~figueiredo/disciplinas/aulas/req-funcional-rnf\_v01.pdf>. Acesso em: 08 Nov. 2024.

GROFFE, R. J. **Modelagem de sistemas através de UML: uma visão geral.** 2013. Disponível em: <https://www.devmedia.com.br/modelagem-de-sistemas-atraves-de-uml-uma-visao-geral/27913>. Acesso em: 08 Nov. 2024.

GUEDES, G. T. A. **UML 2: Uma Abordagem Prática.** São Paulo: Novatec Editora, 2009.

MACÊDO, D. **Introdução a UML e seus diagramas**. Disponível em: <https://www.diegomacedo.com.br/introducao-a-uml-e-seus-diagramas/>. Acesso em 08 Nov. 2024.