SERVIÇO NACIONAL DE APRENDIZAGEM COMERCIAL

SENAC

CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PROJETO INTEGRADOR III: PROPOSTA DE SISTEMA ORIENTADO A OBJETOS

DA

GA

JE

MU

RA

...

EAD - ENSINO À DISTÂNCIA - 2023

DA

GA

JE

MU

RA

PROJETO INTEGRADOR III: PROPOSTA DE SISTEMA ORIENTADO A OBJETOS

Enoque Felipe dos Santos Leal

TRABALHO PARA APROVAÇÃO EM DISCIPLINA

EAD - ENSINO À DISTÂNCIA - 2023

Resumo

Desenvolvimento de um software baseado em Programação Orientada a Objetos para fornecimento de integração e automação das atividades administrativas de cadastramento, fornecimento de produtos e serviços controle de clientes para uma universidade. Onde serão elaborados Controle do processo de implementação com metodologia Ágil, escolha da plataforma de desenvolvimento, demonstração abstrata e desenvolvimento com processo de modelagem UML. Durante o desenvolvimento da ideia inicial foi realizado desenvolvimento de caso de uso para representar um cenário de gestão escolar e foi levantado questões para as seguintes partes Pessoa física, Pessoa jurídica, Professores, Fornecedores e Alunos. Este processo inicial também foi discutido custos e viabilidade de desenvolvimento partindo do zero ou utilizar ferramentas já aplicadas no mercado em atendimento ao solicitado para não onerar o preço final, visto que deve ser um produto acessível e com uma boa proposta para aceitação do produto a ser desenvolvido. Será um desafio econômico financeiro devido à concorrência com um mercado existente nessa área, sendo que a maioria dos produtos oferecidos. Nosso potencial será na facilidade de implementação trazendo assim uma perspectiva de celeridade do processo podendo partir com desenvolvimento para futuros clientes.

Palavras-chave: Processo, Sistema, Desenvolvimento, Software.

Sumário

1. Introdução

Na primeira fase do desenvolvimento de qualquer projeto de software, é necessário realizar a documentação de todos os requisitos, funcionais e não-funcionais, do sistema a ser desenvolvido, e para isso é utilizada a UML (Unified Modeling Language), que é um modelo de padronização das especificações de um projeto que envolva programação orientada a objetos. Tal padronização prevê a adoção de treze tipos de diagramas, sendo um deles o diagrama de casos de uso, que será utilizado ao longo do desenvolvimento deste primeiro tópico.

1.1. Desenvolvimento de Caso de Uso

Tendo em vista a necessidade de desenvolvimento de um sistema de gestão de dados de um centro universitário e considerando a existência de cinco atores que fazem parte deste sistema - pessoa física, pessoa jurídica, professores, fornecedores e alunos, foram desenvolvidos o seguinte diagrama de casos de uso:

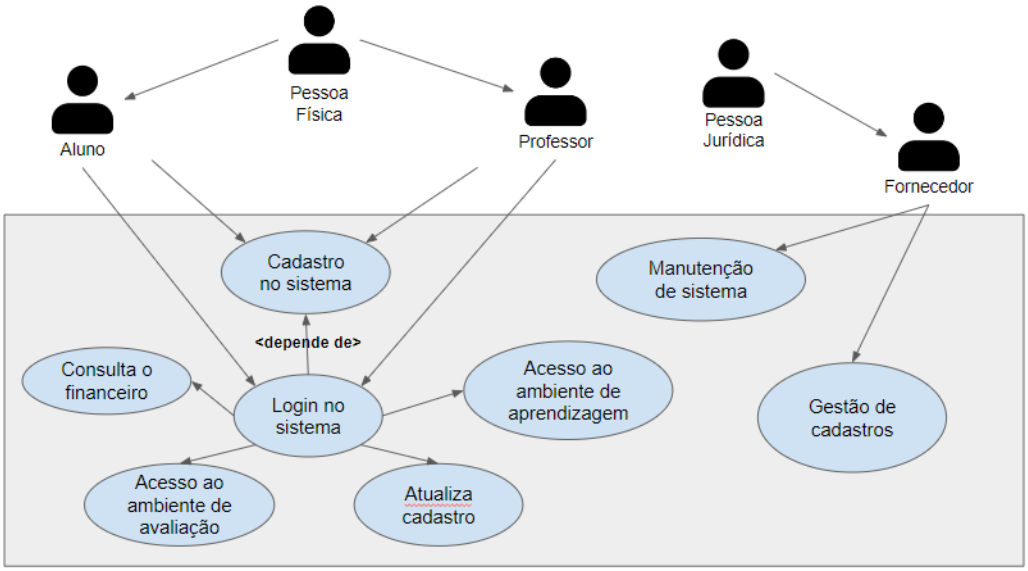

Figura 1 - Diagrama de casos de uso

1.2. Abstração

De acordo com Barnes e Kölling (2010, p. 50), “abstração é a capacidade de ignorar detalhes das partes para focalizar a atenção em um nível mais elevado de um problema”. Trata-se de um conceito importante na programação orientada a objetos, uma vez que a etapa de levantamento de requisitos envolve um grau considerável de abstração, necessária para elaborar os diversos diagramas abordados na UML.

1.3. Metodologia Ágil Utilizada

A metodologia ágil escolhida para este projeto foi a Scrum, que consiste em um framework de atividades e processos de software organizados em cinco etapas principais: levantamento de requisitos, análise, projeto, evolução e entrega. Nesta primeira parte do trabalho, daremos maior foco ao levantamento de requisitos, onde é realizada a modelagem de sistema com uso da UML.

2. Descrição de Cenário

Considerando o diagrama de casos de uso apresentado no tópico 1.1, é possível observar que temos dois atores principais de negócios (a pessoa física e a pessoa jurídica), sendo que abaixo da pessoa física temos dois atores principais de sistema, o aluno e o professor. Atores principais de sistema são aqueles que interagem diretamente com o sistema, iniciando eventos.

Nos quadros a seguir, é possível observar todos os cenários contemplados no diagrama de casos de uso para cada caso descrito.

Cenário	Cadastro no sistema
Atores	Aluno e Professor
Pré-condição	Ser Aluno ou Professor na escola
Fluxo principal	1. Usuário executa o programa; 2. Usuário navega até aba de cadastro; 3. Usuário preenche dados cadastrais; 4. Usuário conclui o cadastro.
Fluxo alternativo 1	1. Usuário executa o programa; 2. Usuário navega até aba de cadastro; 3. Usuário preenche dados cadastrais; 4. Sistema exibe mensagem “Usuário existente. Deseja efetuar login?”; 5. Usuário realiza login com seus dados previamente criados.
Fluxo alternativo 2	1. Usuário executa o programa; 2. Usuário navega até aba de cadastro; 3. Usuário preenche dados cadastrais; 4. Sistema exibe mensagem “Usuário existente. Deseja efetuar login?”; 5. Usuário clica em “Esqueci minha senha”; 6. Usuário recebe instruções de redefinição de senha; 7. Usuário redefine senha; 8. Usuário realiza login com seus dados previamente criados.
Pós-condição	Usuário encontra-se logado no sistema.

Quadro 1 - Cenário de cadastro no sistema

Cenário	Gestão de cadastros
Atores	Fornecedores
Pré-condição	Ter CNPJ
Fluxo principal	1. Usuário executa o programa; 2. Usuário navega até aba de consulta de cadastro; 3. Usuário preenche número do CNPJ fornecedor; 4. Usuário abre o cadastro;
Fluxo alternativo 1	5. CNPJ não encontrado na base de dados; 6. Usuário clica em cadastrar fornecedor; 7. Usuário preenche dados cadastrais do fornecedor; 8. Sistema exibe mensagem “Usuário Cadastrado com Sucesso!”; 9. Usuário abre o cadastro o fornecedor
Pós-condição	Fornecedor cadastrado no sistema.

Quadro 2 - Cenário gestão de cadastros de fornecedores no sistema

3. Desenvolvimento e Estudos

Nesta seção, abordaremos o desenvolvimento das classes e os relacionamentos no sistema de gestão de dados do centro universitário.

3.1. Definição das classes

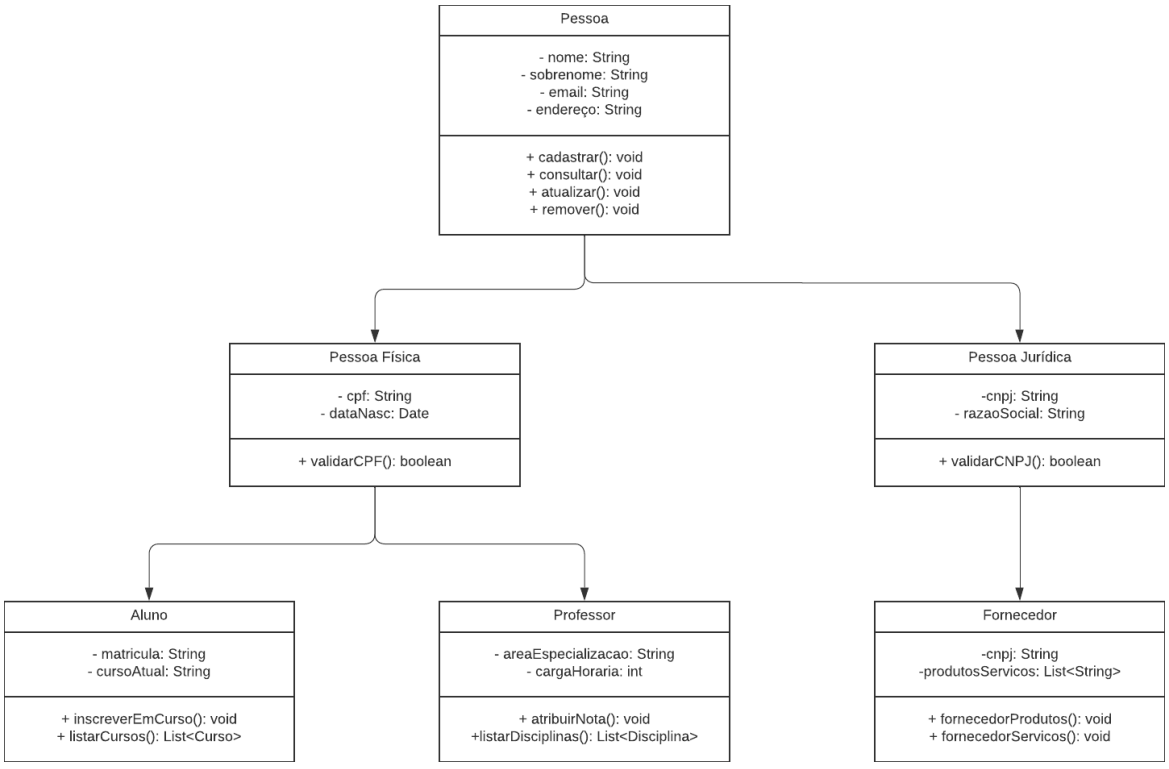

Figura 2 - Classes Definidas

3.2. Relacionamentos (Herança, Interface, Associação)

Estabelecemos os seguintes relacionamentos entre as classes:

- Herança de "Pessoa": "Pessoa Física" e "Pessoa Jurídica" herdam atributos e métodos de "Pessoa".

- Associação entre "Aluno" e "Curso": Um aluno pode se inscrever em vários cursos, e um curso pode ter vários alunos (relação muitos-para-muitos).

- Associação entre "Professor" e "Disciplina": Um professor pode lecionar várias disciplinas, e uma disciplina pode ser lecionada por vários professores (relação muitos-para-muitos).

- Associação entre "Aluno" e "Professor": Um aluno pode ter um orientador/professor responsável.

- Associação entre "Aluno" e "Fornecedor": Um aluno pode se relacionar com fornecedores para atividades extracurriculares.

Este diagrama de classes representa uma visão geral das entidades e relacionamentos do sistema de gestão de dados do centro universitário. A implementação final dependerá dos requisitos detalhados do projeto.

4. Programação

4.1 Levantamento da Linguagem

A Linguagem de programação utilizada será em C#, HTML e CSS conforme estrutura do esboço da GUI apresentada abaixo em Figma.

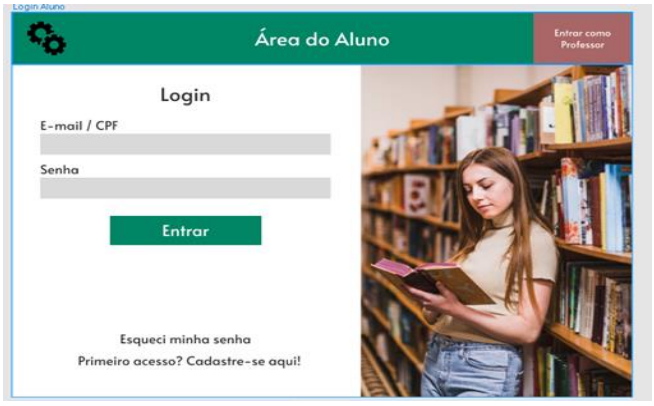

Figura 3 – Entrada Aluno

Figura 4 – Não reconhece a Senha do Aluno

Figura 5 – Cadastro Aluno

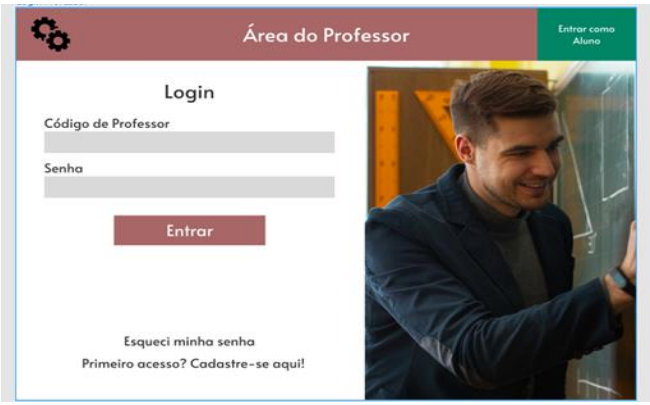

Figura 6 – Entrada Professor

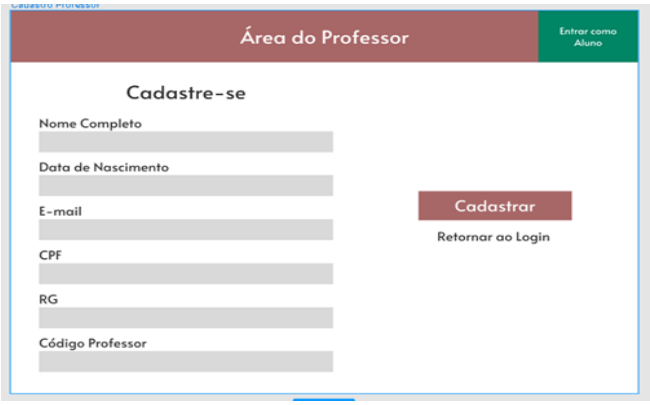

Figura 7 – Cadastro Professor

Figura 8 – Entrada Fornecedor

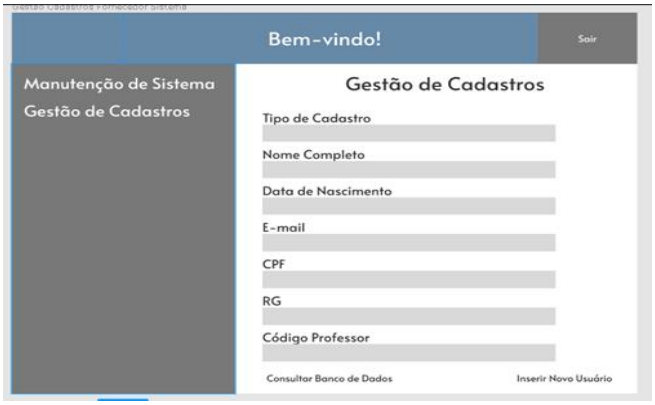

Figura 9 – Cadastro Fornecedor

4.2 Apresentação Previa do Programa

```
1 using System;
2 using System.Collections.Generic;
3 using System.Net.Http;
4 using System.Text;
5 using System.Threading.Tasks;
6 class Program {
7    static List<string> names = new List<string>();
8    static List<DateTime> datadenascimento = new List<DateTime>();
9    static List<string> emails = new List<string>();
10  static List<string> identidade = new List<string>();
11  static List<int> cpf = new List<int>();
12  static List<int> matricula = new List<int>();
13  static void Main() {
```

```
14   while (true)  {
15     Console.WriteLine("1. Adicionar Aluno");
16     Console.WriteLine("2. Mostrar Lista");
17     Console.WriteLine("3. Login");
18     Console.WriteLine("4. Sair");
19     Console.Write("Escolha um código: ");
20     if (!int.TryParse(Console.ReadLine(), out int choice) || choice < 1 || choice > 4) {
21        Console.WriteLine("Opção inválida. Tente novamente.\n");
22  continue; }
23       switch (choice)  {
24       case 1:                    // Código de cadastro aqui (sem alterações)
25          Console.WriteLine("Inicie o cadastro:");
26                                  // (restante do código de cadastro)
27  break;
28       case 2:                    // Código para mostrar a lista (sem alterações)
29          Console.WriteLine("Lista do cadastro:");
30                                 // ... (restante do código de mostrar lista)
31  break;
32       case 3:
33                                 // Código de login aqui
34          Console.WriteLine("Bem-vindo à tela de login!");
35          Console.Write("Digite seu CPF: ");
36         if (!int.TryParse(Console.ReadLine(), out int cpfInput)) {
37              Console.WriteLine("CPF inválido. Tente novamente.\n");
38  continue; }
39              Console.Write("Digite sua senha (mínimo 8 caracteres): ");
40              string senhaInput = Console.ReadLine();
41        if (RealizarLogin(cpfInput, senhaInput)) {
42              Console.WriteLine("Login bem-sucedido!")    }
43        else {
```

```
44          Console.WriteLine("Usuário ou senha inválidos. Tente novamente.") }
45  break;
46      case 4:
47                  Console.WriteLine("Saindo do programa!");
48 return;
49          default:
50                  Console.WriteLine("Opção inválida. Tente novamente.\n");
51 break;
52          }
53      }
54   }
55   static bool RealizarLogin(int cpfInput, string senhaInput) {
56 // Simulação de uma API de autenticação (usando JSONPlaceholder para fins de teste)
57                      // Endpoint para autenticação
58      string apiUrl = "https://jsonplaceholder.typicode.com/posts";
59              using (HttpClient client = new HttpClient()) {
60  try  {
61            // Simulação de uma requisição POST para autenticação
62  var content = new StringContent($"{{\"usuario\": \"{cpfInput}\", \"senha\": \"{senhaInput}\"}}", Encoding.UTF8, "application/json");
63   HttpResponseMessage response = client.PostAsync(apiUrl, content).Result;
64            return response.IsSuccessStatusCode;
65   }
66    catch (Exception ex)  {
67         Console.WriteLine($"Erro de conexão: {ex.Message}");
68             return false;
69        }
70    }
71  }
```

4.2.1 Acesso aos códigos HTML e CSS

Neste aspecto devido à complexidade do código foi disponibilizado um esboço do projeto na plataforma Github.

Onde segue conforme endereço eletrônico de acesso abaixo:

https://github.com/JessicaSchmidtG/Projeto_Integrador_III.git

Conclusão

Nosso Trabalho trouxe uma perspectiva diferenciada e Ágil, contudo a linguagem utilizada ficou presa diante as plataformas de sistemas operacionais atuais dificultando assim o interesse para nosso projeto. Também a falta de desenvolvimento voltado à segurança cibernética e a Lei Geral de Proteção de Dados que são abordagens atuais como a implementação em nuvem deixa a desejar, por outro lado alguns sistemas simples e para pequenos negócios pode ser uma opção para uma solução rápida e de baixo custo. Sendo assim a viabilidade deste projeto fica sujeita a pequenas empresas com um sistema fechado para seu controle e segurança. E não sendo viável para empresas de médio e grande porte onde este projeto ficaria sujeito a grandes alterações aumentando seu custo e tempo de implantação.

Referências

BARNES, David J; KÔLLING, Michael. Programação Orientada a Objetos com Java: uma introdução pratica usando o BlueJ. São Paulo: Pearson, 2010.

PESSÔA Filho, Joaquim. Programação Orientada a Objetos com C#; Capitulo 01 a 08. São Paulo: Editora Senac São Paulo, 2003.

Programação Orientada a Objetos – Aula 05. Prof. Dr. Fernando Fernandes; bbcollab. Disponível em: <https://ca-lti.bbcollab.com/collab/ui/session/playback>. Acesso em: 15 de Setembro de 2023 às 21:45.

Doze Princípios do Software Ágil; agilemanifesto. Disponível em: <https://agilemanifesto.org/iso/ptbr/principles.html>. Acesso em: 9 de Setembro de 2023 às 20:33.

UML Class Diagrams Examples; agilemanifesto. uml-diagrams. Disponível em: < https://www.uml-diagrams.org/class-diagrams-examples.html>. Acesso em: 23 de Setembro de 2023 às 22:12.

Projeto Integrador III Front Finalizado, Grupo37; Github. Disponivel em: <https://github.com/JessicaSchmidtG/Projeto_Integrador_III/tree/10ee8d68076cc6e25620ab1edc8b03eaca7a780b> Acesso em: 21 de Novembro de 2023.