UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GÉSSICA DE WALLAU

APLICATIVO PARA ESCOLAS DE DANÇA

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2016

GÉSSICA DE WALLAU

APLICATIVO PARA ESCOLAS DE DANÇA

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientadora: Profa. Beatriz Terezinha Borsoi

PATO BRANCO
2016

ATA Nº: **285**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DA ALUNA GESSICA DE WALLAU**.**

Às 14:30 hrs do dia 29 de novembro de 2016, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Andréia Scariot Beulke (Convidada) e Adriana Ariati (Convidada), para avaliar o Trabalho de Diplomação da aluna Gessica de Wallau, matrícula 01209736, sob o título **Aplicativo para escolas de dança**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação a candidata foi entrevistada pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 15:00 hrs foi encerrada a sessão.

_____
Profa. Beatriz Terezinha Borsoi, Dr.
Orientadora

_____
Profa. Andréia Scariot Beulke, Esp.
Convidada

_____
Profa. Adriana Ariati, Esp.
Convidada

_____	_____
Profa. Eliane Maria de Bortoli Fávero, M.Sc	Prof. Edilson Pontarolo, Dr.
Coordenadora do Trabalho de Diplomação	Coordenador do Curso

A Folha de Aprovação assinada encontra-se na Coordenação do Curso

RESUMO

WALLAU, Géssica de. Aplicativo para escolas de dança. 2016. 56f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2016.

A dança é uma atividade existente desde os primórdios da humanidade. Isso pode ser constatado em registros como as representações gráficas encontradas em cavernas. As pessoas dançam por motivos diversos, como, por exemplo, de festivos a fúnebres. Com o passar do tempo, a dança passou por especialização e grande diversificação. Atualmente existem tipos diversos de danças e locais específicos para seu aprendizado e prática. Escolas de dança disponibilizam profissionais para o aprendizado de tipos diferentes de dança. Os alunos dessas escolas podem ser organizados em turmas e realizar apresentações como os espetáculos de encerramento de ano letivo que tipicamente ocorrem em escolas de balé. Esses espetáculos precisam ser organizados quanto aos figurinos utilizados e sequenciamento de apresentação de maneira que um mesmo aluno possa participar de apresentações distintas em um mesmo evento. Essas escolas também precisam gerir o controle de pagamentos e recebimentos, como de funcionários e despesas de manutenção e mensalidades dos alunos. Considerando esse contexto, um aplicativo foi desenvolvido, como resultado deste trabalho, visando auxiliar no controle das atividades realizadas por escolas de dança. O aplicativo é web e foi desenvolvido com a linguagem C#, o banco de dados PostgreSQL e NHibernate para o mapeamento objeto relacional.

Palavras-chave: C#. Sistema para escolas de dança. NHibernate.

ABSTRACT

WALLAU, Géssica de. Software to dance schools. 2016. 56f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2016.

Dance exists, so it can be evidenced in records such as the graphic representations found in caves, since the beginnings of humanity. People danced and dance for many reasons, festive or funeral. Over time, the dance has undergone specializations and great diversification. Currently there are different kinds of dances and specific places for your learning and practice. Dance schools provide professionals for learning different types of dance. Students from these schools can be organized into classes and perform presentations such as the school year closing spectacles that typically occur in ballet schools. These shows need to be organized regarding the costumes used and presentation sequencing so that the same student can participate in different presentations in the same event. These schools also need to manage the control of payments and receipts, such as staff and maintenance expenses, and student montly payment. Considering this context an application was developed, as a result of this work, aiming at assisting in the control of the activities performed by dance schools. The application is web and developed with C # language, PostgreSQL database and NHibernate for the object-relatinal mapping.

Keywords: C#. Software to dance schools. NHibernate.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGO

LISTA DE SIGLAS

MVVM	*Model-View-ViewModel*
RF	Requisitos Funcionais
RNF	Requisitos Não Funcionais
SIG	Sistema de Informação Gerencial
WPF	*Windows Presentation Foundation*
XAML	*Extensible Application Markup Language*
WPF	*Windows Presentation Foundation*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais, os objetivos e a justificativa de realização do trabalho. Por fim, está a apresentação dos capítulos subsequentes.

1.1 CONSIDERAÇÕES INICIAIS

A dança é uma atividade existente desde o tempo dos homens primitivos. Nessa época, dançava-se instintivamente, como forma de agradecimento pela vida, pela natureza e pela conquista de alimentos. O homem primitivo costumava registrar, na forma de pintura em cavernas, grutas e/ou galerias subterrâneas, cenas de caça, como se fossem um ritual, visto que a obtenção de alimento era considerada algo sagrado (SEED, 2016).

Em termos de história, é retratado o surgimento da dança como sendo na Pré-História, quando os homens batiam os pés no chão. Essas batidas aos poucos foram fornecendo mais intensidade aos sons e permitindo descobrir que podiam fazer outros ritmos, conjugando os passos com as mãos, com o uso das palmas (BARROS, 2016). É fato que posteriormente foram acrescentados os instrumentos musicais, mesmo que fossem, ainda, bastante rústicos.

Durante o século XIV, predominavam as danças religiosas realizadas pelos artesãos e utilizadas nas festas em família. No início dos anos 1900, a dança contemporânea começou a ser inserida nos salões e em escolas de dança (TIPOS DE DANÇA, 2016).

Atualmente existem diversos tipos ou estilos de dança, como, por exemplo, folclórico, clássico, moderno e de rua. É comum haver escolas para ensinar determinados estilos dança e locais como salões específicos para que essas danças possam ser praticadas.

Para a aprendizagem e prática de danças passaram a surgir as escolas específicas em termos de estilo ou tipo de dança. Em determinado tipo de escola, como as de balé, por exemplo, como forma de avaliar os alunos e promover o estabelecimento, repete-se a tradição de espetáculos de dança em determinadas

datas comemorativas. Esses espetáculos, também, geralmente, ocorrem no final do ano como forma de encerrar as atividades do respectivo ano letivo.

Com o desenvolvimento dessas escolas e o aumento do número de alunos, esses espetáculos ficaram cada vez mais elaborados e grandiosos. Surgindo, assim a necessidade de planejar e gerenciar tais eventos e seus participantes. Além disso, é necessário o controle das turmas de alunos, das receitas e despesas e de outras atividades realizadas pela escola, como as próprias aulas de dança, além dos espetáculos em si. Considerando esse contexto, este trabalho visa o desenvolvimento de um aplicativo para o controle das principais atividades realizadas por escolas de dança. A ênfase está no gerenciamento dos espetáculos, mas as principais atividades de negócio desse tipo de estabelecimento serão consideradas pelo aplicativo.

1.2 OBJETIVOS

A seguir são apresentados os objetivos pretendidos com a realização deste trabalho.

1.2.1 Objetivo Geral

Desenvolver um sistema para auxiliar no controle das principais atividades rde uma escola de dança.

1.2.2 Objetivos Específicos

- Prover um controle para gerenciamento dos espetáculos realizados e os respectivos figurinos necessários para realizá-los.
- Fornecer dados que auxiliem no controle financeiro das atividades da escola.

- Auxiliar no controle das atividades que cada aluno realiza na escola, os valores de mensalidade e seu pagamento e dos descontos possíveis, de acordo com as modalidades de dança praticadas.

1.3 JUSTIFICATIVA

Um espetáculo de dança é formado por vários alunos de uma escola. Esses alunos podem fazer parte de uma ou várias turmas, sendo que, em cada turma, na maioria das vezes, possui um papel/personagem diferente, de acordo com o tema estipulado para o espetáculo. Com um sistema que auxilie no gerenciamento desses eventos, os professores poderão organizar, de maneira mais eficiente, a sequência de apresentação de turmas e/ou alunos. Assim, se um aluno participa de uma turma em duas modalidades distintas, ele deve ter um tempo mínimo (uma ou mais de uma apresentação, por exemplo), para trocar de figurino, para, assim, poder se apresentar com as turmas das quais ele participa.

É essencial que o administrador da escola tenha controle do papel/personagem de cada aluno e/ou turma, para poder planejar e organizar a confecção dos figurinos. Da mesma maneira que nas apresentações, o aluno pode ter mais de um figurino, e, como o pagamento por eles é de sua responsabilidade, o administrador deve ter o controle dos recebimentos, visto que, geralmente, é ele quem repassa o valor total à empresa que os fabrica.

Com o controle de alunos e as modalidades praticadas por cada um deles, o administrador da escola pode definir um valor ou percentual de desconto que pode oferecer para cada aluno nas suas mensalidades. Assim, um controle de receitas e despesas facilita a gestão do negócio.

Com o auxílio de uma ferramenta para apoio nas atividades da gestão, tanto da escola em geral, quanto dos eventos realizados, o administrador da escola e os seus professores podem trabalhar melhor na divulgação (*marketing*) dos trabalhos realizados, visando ampliar a quantidade de alunos e de empresas parceiras. Parcerias com clubes, academias, spas, boutiques e outros podem gerar descontos para os participantes.

1.4 ESTRUTURA DO TRABALHO

O restante deste texto está estruturado em capítulos. O Capítulo 2 apresenta o referencial teórico que fundamenta o tipo de aplicação desenvolvida, um sistema gerencial. As tecnologias e as ferramentas utilizadas na modelagem e no desenvolvimento do sistema e o método empregado são apresentadas no Capítulo 3. O Capítulo 4 apresenta os resultados obtidos com a realização do trabalho. Nesse capítulo, o sistema é apresentadopor meio da descrição de suas prinicpais funcionalidades,telas e partes relevantes de código. O texto é finalizado com as considerações finais que estão no Capítulo 5. Essas considerações são seguidas das referências utilizadas na composição do referencial teórico do texto.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o suporte conceitual do aplicativo desenvolvido como resultado deste trabalho que é um sistema de informação.

2.1 CONCEITO DE SISTEMAS DE INFORMAÇÃO

Pereira e Fonseca (1997, p. 241), conceituam Sistemas de Informação como mecanismos de apoio à gestão, desenvolvidos com base na tecnologia de informação e com suporte da informática para atuar como condutores das informações que visam facilitar, agilizar e otimizar o processo decisório nas organizações. Especializando o conceito de Sistemas de Informação, Oliveira (2008) define Sistemas de Informações Gerenciais (SIG) como os processos que são utilizados para transformar dados em informações que auxiliem no processo decisório da empresa.

Os Sistemas de Informação têm por objetivo gerar informações para suporte à tomada de decisão. Por meio desses sistemas, dados são coletados, processados e transformados em informação (STAIR, 2008). O processo de transformação dos dados produzidos pelos processos de negócio da empresa em informação útil para apoio à tomada de decisão tem sido suportado pelos Sistemas de Informação. Esse suporte é no sentido de armazenamento, recuperação e análise dos dados, visando descoberta de relações entre os dados que possam ser úteis aos gestores.

Os SIGs são sistemas ou processos que fornecem as informações necessárias para gerenciar com eficácia as organizações (MARTINEZ, 2016). Um SIG gera informações que podem fornecer suporte para o processo de tomada de decisão que é fundamental para expandir e mesmo manter empresas e negócios no mercado sendo competitivos. Para Martinez (2016) esses sistemas resultam da interação colaborativa entre pessoas, tecnologias e procedimentos, manipulando dados que ajudam uma organização a atingir as suas metas.

Considerando o grande volume de dados que as empresas têm manipulado, a necessidade de auxilio de Sistemas de Informação para o armazenamento e a seguranças desses dados é cada vez mais evidente. Além disso, os dados armazenados passam a ser cada vez mais úteis no processo de tomada de decisão,

apoiando o gestor na escolha de direcionamento para o negócio e na definição de estratégias para manter a empresa no mercado, atender as expectativas dos clientes, ampliar o negócio e lidar com a concorrência.

2.2 IMPORTÂNCIA DOS SISTEMAS DE INFORMAÇÃO

Um Sistema de Informação pode, sob variadas condições, trazer benefícios para as empresas. Entre esses benefícios, Oliveira (2000) e Stair e Reynolds (2002) destacam:

a) Reduzir os custos das operações;

b) Facilitar o acesso às informações, com relatórios mais consistentes e com menor esforço por parte do usuário;

c) Aumentar a produtividade das pessoas pela facilidade de registro e recuperação dos dados;

d) Estimular maior interação entre os tomadores de decisão;

e) Fornecer projeções mais amplas e efetivas dos efeitos das decisões;

f) Auxiliar na melhoria da estrutura organizacional, facilitando o fluxo de informações;

g) Colaborar para a redução do grau de centralização de decisões na empresa;

h) Facilitar na adaptação da empresa para enfrentar acontecimentos não previstos.

Visando atender esses objetivos, Rezende e Abreu (2008) destacam que, em geral, os Sistemas de Informação procuram atuar como:

a) Ferramentas para auxiliar no funcionamento das empresas;

b) Instrumentos que possibilitam uma avaliação analítica e, quando necessária, sintética das empresas;

c) Facilitadores dos processos internos e externos;

d) Meios para suportar a qualidade, produtividade e inovação organizacional;

e) Geradores de modelos de informações para auxiliar nos processos decisórios;

f) Produtores de informações oportunas e geradores de conhecimento;

g) Valores agregados e complementares à modernidade, perenidade, lucratividade e competitividade empresarial.

As diversas formas de atuação dos sistemas permitem que as empresas conheçam o seu potencial interno e estejam melhor preparadas para atuar no meio externo e manter-se no mercado, mesmo com a ampla concorrência advinda da globalização e das tecnologias de comunicação e informação que tem sido empregadas para fazer negócio e divulgar produtos.

3 MATERIAIS E MÉTODO

A seguir estão os materiais e o método utilizados para a modelagem e a implementação do sistema obtido como resultado deste trabalho.

3.1 MATERIAIS

Para a modelagem e a implementação do sistema, foram utilizadas as ferramentas e tecnologias apresentadas no Quadro 1.

Ferramenta / Tecnologia	**Referência (site)**	**Finalidade**
Toad Data Modeler	http://www.casestudio.com/enu/default.aspx	Modelagem do diagrama de banco de dados.
Visual Paradigm	https://www.visual-paradigm.com/	Modelagem de classes, casos de uso.
PGAdmin III	https://www.pgadmin.org/	Gerenciador de banco de dados.
PostgreSQL	https://www.postgresql.org/	Sistema de banco de dados.
Microsoft Visual Studio	https://www.visualstudio.com/	Ambiente de desenvolvimento.
C#	https://msdn.microsoft.com/pt-br/library/kx37x362.aspx	Linguagem de programação.
NHibernate	http://nhibernate.info/	Persistência de dados. Mapeamento de objetos relacionais.

Quadro 1 – Tecnologias e ferramentas utilizadas na modelagem e implementação do aplicativo

3.2 MÉTODO

O método consiste nas atividades de levantamento de requisitos, análise, projeto e desenvolvimento (PRESSMAN, 2008). A seguir estão descritas, de forma sucinta, as etapas desenvolvidas para a realização deste trabalho.

Levantamento de requisitos

O levantamento de requisitos do ponto de vista de negócio ou do usuário foi a primeira atividade realizada. Esses requisitos foram definidos com o auxílio de uma profissional de uma escola de dança que contribuiu na definição das funcionalidades essenciais do sistema em termos de negócio. Essas funcionalidades foram organizadas em requisitos funcionais e não funcionais e posteriormente organizados sob a forma de casos de uso com os atores e suas permissões de acesso a essas funcionalidades.

Análise e projeto

Após a definição dos requisitos, o banco de dados foi modelado gerando um diagrama de entidades e relacionamentos e o descritivo das tabelas desse banco. Como as tabelas representam entidades persistentes do banco de dados, percebeu-se que um diagrama de classes não seria necessário.

Desenvolvimento

O desenvolvimento foi realizado utilizando a linguagem de programação C# e o banco de dados PostegreSQL. Os testes foram informais e realizados à medida que o código era produzido tendo como objetivo identificar erros de codificação e o atendimento das funcionalidades definidas.

4 RESULTADO

Este capítulo apresenta o resultado da realização deste trabalho. O resultado é o desenvolvimento de um sistema para escolas de dança.

4.1 ESCOPO DO SISTEMA

O sistema desenvolvido como resultado deste trabalho é para escolas de danças, visando o auxílio no controle das suas atividades rotineiras (receitas e despesas) e de eventos realizados por ela. Como exemplo dessas atividades, pode-se citar a formação de novas turmas de determinada modalidade de dança, o recebimento de mensalidade dos alunos, a criação de tabelas de descontos para alunos que praticam mais de uma modalidade, o planejamento e a administração de eventos realizados pela escola, bem como a composição de parcerias (clubes, academias, spas, boutiques, etc). A solução proposta pode ser entendida no contexto apresentado a seguir.

Uma escola de dança é composta, basicamente, por profissionais (que irão ministrar as aulas), modalidades (são os diversos tipos de dança que podem ser ofertados pela escola, sendo alguns deles: ballet clássico, jazz, stiletto, hip hop, dança contemporânea, etc) e alunos (que irão praticar as modalidades ofertadas pela escola, mediante pagamento de mensalidade estipulada pela escola, de acordo com a(s) modalidade(s) praticadas).

Geralmente, em datas comemorativas, a escola realiza eventos, tanto para incentivar os alunos (como forma de valorização), quanto para promover o estabelecimento. São exemplos de eventos realizados: workshops, confraternizações, visitas, apresentações em festivais regionais, gravações de comerciais/vídeos institucionais para divulgação nas mídias/redes sociais e espetáculos. Enquanto que os primeiros são eventos mais simples, de menor duração, o último costuma ser um evento grandioso, como forma de encerramento do ano letivo e uma das principais maneiras de divulgar o trabalho dos alunos, profissionais e parceiros da escola.

Para esses espetáculos, primeiramente, é definido o tema principal: a mensagem que a escola deseja mostrar. Depois disso, os personagens são criados.

Eles são representados pelos alunos (que pode ser tanto uma ou mais turmas de alunos, quanto somente um aluno, em apresentação solo) e que podem estar ou não acompanhados de seus professores (profissionais da escola). Após definir os personagens, é necessário planejar e confeccionar os figurinos utilizados por eles. Sendo essa a base do espetáculo, customiza-se os detalhes do evento: sequências da apresentação, trilha sonora, coreografias, iluminação, cenário, fotografia e filmagem, etc.

O software modelado e implementado neste trabalho visa ser uma ferramenta de apoio ao profissional em termos de organização dos eventos e espetáculos promovidos pela escola e uma forma de controle administrativo para o administrador da escola de dança.

4.2 MODELAGEM DO SISTEMA

O Quadro 2 apresenta os Requisitos Funcionais (RF) definidos para o sistema.

Identificação	Nome	Descrição
RF01	Cadastrar usuários	São as pessoas que terão acesso ao sistema.
RF02	Cadastrar profissionais	São os professores das modalidades de dança da escola.
RF03	Cadastrar alunos	São os alunos da escola.
RF04	Cadastrar modalidades	As modalidades são os tipos de dança trabalhadas pela escola: ballet, jazz, stiletto, hip hop, etc.
RF05	Gerenciar turma	Toda turma terá um nome, um nível (infantil, juvenil e adulto), um profissional responsável, uma modalidade, uma sala, seus respectivos alunos, listas de presença e diários de classe.
RF06	Gerenciar evento	Os eventos promovidos pela escola podem ser de vários tipos: espetáculos, workshops, visitas, entrevistas, confraternização, etc.
RF07	Gerenciar caixa	Proporciona o gerenciamento do caixa da escola, ou seja, suas receitas (créditos) e suas despesas (débitos), permitindo efetuar manualmente esses lançamentos.
RF08	Cadastrar parceiros	As parcerias podem conceder benefícios para com outras empresas, que podem ser: clubes, academias, boutiques, spas, centros de estética, educadores físicos, etc.
RF09	Estabelecer descontos	Configura uma tabela de descontos, os quais serão concedidos ao aluno que praticar mais de uma modalidade de dança.
RF10	Controlar figurinos	Fornece o controle dos figurinos utilizados pelos personagens dos eventos realizados pela escola. Essa rotina estará ligada a de Gerenciamento de Eventos.
RF11	Controlar personagens	Realiza o controle dos personagens dos eventos realizados pela escola. Essa rotina estará ligada a de Gerenciamento de Eventos.

RF12	Realizar matrícula	A matrícula é um documento que comprova e permite a participação de alunos em determinada turma de determinada modalidade de dança. Ela gerará uma taxa a ser recebida.
RF13	Registrar diário de classe	O registro dos diários de classe serve para que o profissional anote as observações pertinentes a uma determinada turma, para uma determinada data.
RF14	Inserir dados de documentos	Os documentos representarão contas a pagar (despesas básicas da escola): aluguel, conta de água, luz, internet, etc, e contas a receber: proventos de parceiros/patrocinadores, etc.

Quadro 2 – Requisitos funcionais

Os Requisitos Não Funcionais (RNF) definidos para o sistema são apresentados no Quadro 3.

Identificação	Nome	Descrição
RNF01	Acesso ao sistema	O acesso ao sistema será realizado por meio de autenticação de *login* e senha.

Quadro 3 – Requisitos não funcionais

Os requisitos foram organizados em um diagrama de caso de uso que descreve as principais funcionalidades do sistema, baseadas nos requisitos funcionais e a interação dessas funcionalidades com os atores (que categorizam os usuários) do sistema.

Para o sistema para gerenciamento de escolas de dança foram definidos os seguintes atores:

a) Profissional – realiza o registro das aulas por meio do controle de presenças e do preenchimento do diário de classe. Cada turma possui um profissional associado e é esse profissional quem faz o registro de aulas, presenças e diários de classe da respectiva turma.
b) Administrador da escola – realiza as principais funcionalidades do sistema relacionadas aos alunos como cadastros, matrículas, mensalidades e descontos, caixa (pagamentos e recebimentos), gerenciamento de eventos e formação de parcerias. Esse usuário também herda as permissões de profissional.

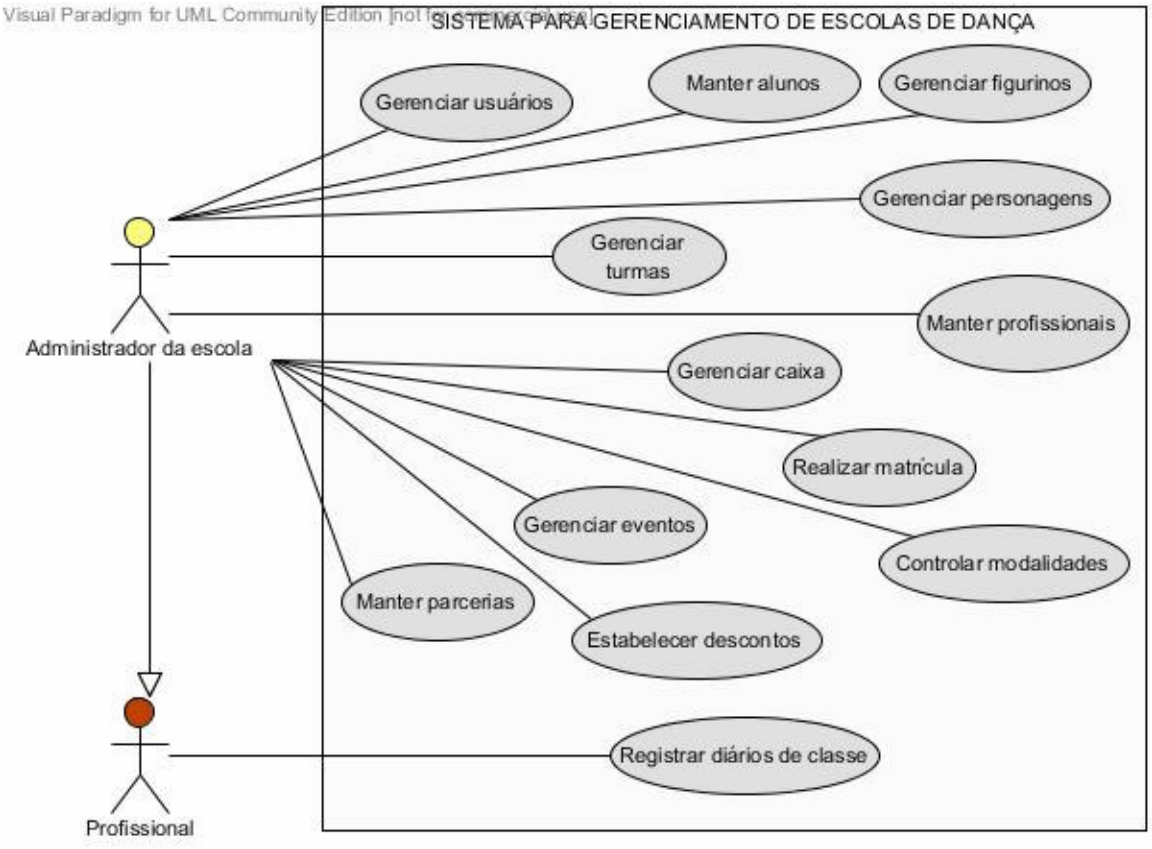

Figura 1 – Diagrama de caso de uso

Os quadros a seguir apresentam a expansão de casos de uso. No Quadro 4 está a expansão da operação de inclusão dos cadastros.

Caso de uso: Inclusão de registros. **Descrição:** Essa rotina é responsável pelo cadastro de registros, que podem ser: usuários, profissionais, modalidades, salas, parceiros, eventos e lançamentos de caixa (dos tipos “crédito” e “débito”). **Evento iniciador:** Ator abre a rotina de cadastro. **Atores:** Administrador. **Pré condição:** Não há. **Sequência de eventos:** 1. Ator clica sobre a rotina de cadastro; 2. Sistema apresenta a tela; 3. Ator informa os dados do cadastro; 4. Sistema grava o registro no banco de dados. **Pós condição:** Registro de cadastro.

Nome do fluxo alternativo (expansão)	Descrição
Linha 4: Dados obrigatórios não preenchidos	4.1 Ao salvar a rotina, o sistema valida se todos os campos obrigatórios foram informados; 4.2 O sistema emite mensagem ao usuário, informando-o que há campos obrigatórios não preenchidos; 4.3 O sistema retorna para a tela de cadastro em forma de edição, para que o usuário informe os campos obrigatórios não preenchidos.

Quadro 4 – Expansão da operação de inclusão de registro

A expansão da operação de alteração dos cadastros é apresentada no Quadro 5.

Caso de uso: Alteração de registros. **Descrição:** Essa rotina é responsável pela alteração de registros, que podem ser: usuários, profissionais, modalidades, salas, parceiros e eventos. **Evento iniciador:** Ator abre a rotina de cadastro. **Atores:** Administrador. **Pré condição:** Não há. **Sequência de eventos:** 1. Ator clica sobre a rotina de cadastro; 2. Sistema apresenta a tela; 3. Ator pesquisa o registro para alteração; 4. Sistema carrega os dados do registro selecionado; 5. Ator altera os dados desejados; 6. Sistema grava as alterações do registro no banco de dados. **Pós condição:** Registro alterado.	
Nome do fluxo alternativo (expansão)	**Descrição**
Linha 4: Registro informado não existe	4.1 Ao carregar o registro solicitado, o sistema verifica que ele não existe; 4.2 Sistema apresenta mensagem ao usuário, informando-o de que o registro solicitado não existe.
Linha 6: Dados obrigatórios não preenchidos	6.1 Ao salvar a rotina, o sistema valida se todos os campos obrigatórios foram informados; 6.2 O sistema emite mensagem ao usuário, informando-o que há campos obrigatórios não preenchidos; 6.3 O sistema retorna para a tela de cadastro em forma de edição, para que o usuário informe os campos obrigatórios não preenchidos.

Quadro 5 – Expansão da operação de alteração de registro

A operação de consulta de dados de cadastros é apresentada no Quadro 6.

Caso de uso:
Consulta de registros.
Descrição:
Essa é responsável pela consulta dos dados dos registros, que podem ser: usuários, profissionais, modalidades, salas, parceiros e eventos.
Evento iniciador:
Ator abre a rotina do cadastro do registro.
Atores:
Administrador.
Pré condição:
Não há.
Sequência de eventos:
1. Ator clica sobre a rotina de cadastro;
2. Sistema apresenta a tela;
3. Ator informa o registro que deseja visualizar;
4. Sistema carrega os dados do registro na tela;
5. Ator visualiza os dados do registro;
6. Ator solicita o fechamento da tela.
7. Sistema encerra a tela.

Pós condição:
Registro alterado.

Nome do fluxo alternativo (expansão)	**Descrição**
Linha 4: Registro informado não existe	4.1 Ao carregar o registro solicitado, o sistema verifica que ele não existe; 4.2 Sistema apresenta mensagem ao usuário, informando-o de que o registro solicitado não existe.

Quadro 6 – Expansão da operação de consulta de registro

A operação de exclusão de registros dos cadastros do sistema é apresentada de forma expandida no Quadro 7.

Caso de uso:
Exclusão de registros.
Descrição:
Essa rotina é responsável pela exclusão dos registros, que podem ser: usuários, profissionais, modalidades, salas, parceiros e eventos, desde que estes não estejam relacionados entre si.
Evento iniciador:
Ator abre a rotina do cadastro do registro.
Atores:
Administrador.
Pré condição:
Não há.
Sequência de eventos:
1. Ator clica sobre a rotina de cadastro;
2. Sistema apresenta a tela;
3. Ator informa o registro que deseja visualizar;

4. Sistema carrega os dados do registro na tela; 5. Ator visualiza os dados do registro; 6. Ator solicita a exclusão do registro; 7. Sistema verifica se o registro pode ser excluído; 8. Sistema exclui o registro do banco de dados; 9. Sistema apresenta a mensagem de que o registro foi excluído com sucesso. **Pós condição:** Registro excluído.

Nome do fluxo alternativo (expansão)	**Descrição**
Linha 4: Registro informado não existe	4.1 Ao carregar o registro solicitado, o sistema verifica que ele não existe; 4.2 Sistema apresenta mensagem ao usuário, informando-o de que o registro solicitado não existe.
Linha 7: Verificar se registro pode ser excluído	7.1 Ao tentar excluir um registro e o mesmo possuir vínculo com os demais cadastros do sistema, o processo de exclusão é interrompido; 7.2 Sistema apresenta mensagem ao usuário informando-o de que o registro não pode ser excluído; 7.3 Sistema retorna para a tela do registro.

Quadro 7 – Expansão da operação de exclusão de registro

A expansão do caso de uso realizar matrícula é apresentada no Quadro 8.

Caso de uso: Realizar matrícula **Descrição:** Essa rotina é responsável pelo registro de matrícula dos alunos na escola. O registro de matrícula gera uma taxa a ser recebida no caixa. **Evento iniciador:** Ator abre a rotina “Realizar matrícula”. **Atores:** Administrador. **Pré condição:** Possuir modalidades, profissionais e turmas cadastrados no sistema. **Sequência de eventos:** 1. Ator clica sobre a rotina de “Realizar matrícula”; 2. Sistema apresenta a tela; 3. Ator clica sobre a opção de “cadastrar aluno” e informa seus dados; 4. Ator insere a(s) turma(s) nas quais o aluno tem interesse em frequentar; 5. Ator solicita a gravação da matrícula; 6. Sistema grava o registro de matrícula no banco de dados; 7. Sistema gera a taxa de matrícula; 8. Sistema vincula o aluno no cadastro da(s) turma(s) selecionada(s); **Pós condição:** Registro da matrícula do aluno.

Nome do fluxo alternativo (expansão)	**Descrição**

Linha 4: Turmas não encontradas	4.1 O sistema verificará se há alguma turma ativa; 4.2 O sistema apresentará a mensagem de que não foram encontradas turmas ativas disponíveis; 4.3 O sistema voltará a tela de matrícula.

Quadro 8 – Expansão do caso de uso realizar matrícula

O Quadro 9 apresenta a expansão do caso de uso estabelecer desconto.

Caso de uso:
Estabelecer descontos
Descrição:
Essa rotina serve como uma tabela de descontos a conceder aos alunos que participam de mais de uma modalidade. O sistema deve proporcionar configuração para indicar a quantidade de modalidades e o respectivo percentual de desconto.
Evento iniciador:
Ator abre a rotina "Tabela de descontos".
Atores:
Administrador.
Pré condição:
Possuir modalidades, profissionais e turmas cadastradas no sistema.
Sequência de eventos:
1. Ator clica sobre a rotina de "Tabela de Descontos";
2. Sistema apresenta a tela;
3. Ator configura as faixas de desconto a ser concedidas;
4. Ator solicita gravação do registro;
5. Sistema grava a tabela de desconto no banco de dados.

Pós condição:
Registro da tabela de desconto.

Nome do fluxo alternativo (expansão)	**Descrição**
Linha 5: Dados obrigatórios não preenchidos	5.1 Ao salvar a rotina, o sistema valida se todos os campos obrigatórios foram informados; 5.2 O sistema emite mensagem ao usuário, informando-o que há campos obrigatórios não preenchidos; 5.3 O sistema retorna para a tela de "tabela de preço" em forma de edição, para que o usuário informe os campos obrigatórios não preenchidos.

Quadro 9 – Expansão do caso de uso estabelecer desconto

A expansão do caso de uso gerenciar caixa é apresentada no Quadro 10.

Caso de uso:
Gerenciar caixa
Descrição:
Essa rotina serve para gerenciar as receitas e despesas da escola de dança: controlar todas as entradas e saídas monetárias.
Evento iniciador:
Ator abre a rotina "Caixa".

Atores: Administrador. **Pré condição:** Não há. **Sequência de eventos:** 1. Ator clica sobre a rotina de “Caixa”; 2. Sistema apresenta a tela; 3. Ator informa a data dos lançamentos que deseja visualizar; 4. Sistema apresenta os registros existentes; 5. Ator pode solicitar a inclusão de um lançamento de caixa nessa rotina; 6. Sistema gera e apresenta o quadro de horários. **Pós condição:** Apresentação dos lançamentos de caixa efetuados.

Nome do fluxo alternativo (expansão)	**Descrição**
Linha 5: Dados obrigatórios não preenchidos	5.1 Ao salvar a rotina, o sistema valida se todos os campos obrigatórios foram informados; 5.2 O sistema emite mensagem ao usuário, informando-o que há campos obrigatórios não preenchidos; 5.3 O sistema retorna para a tela de “diário de classe” em forma de edição, para que o usuário informe os campos obrigatórios não preenchidos.

Quadro 10 – Expansão do caso de uso gerenciar caixa

No Quadro 13 está a expansão do caso de uso gerenciar eventos.

Caso de uso: Gerenciar eventos **Descrição:** Essa rotina serve para gerenciar os eventos da escola. Eles podem ser de diversos tipos: espetáculo, *workshop*, festival, oficina, entrevista, visitas, etc. Dependendo do tipo de evento, haverá configurações específicas a serem feitas. **Evento iniciador:** Ator abre a rotina “Gerenciar Eventos”. **Atores:** Administrador. **Pré condição:** Não há. **Sequência de eventos:** 1. Ator clica sobre a rotina de “Gerenciar Evento”; 2. Sistema apresenta a tela; 3. Ator informa os dados obrigatórios e facultativos; 4. Sistema grava esses dados no banco de dados. **Pós condição:** Registro de evento.

Nome do fluxo alternativo (expansão)	**Descrição**
Linha 4: Dados obrigatórios não	4.1 Ao salvar a rotina, o sistema valida se todos os campos obrigatórios foram informados;

preenchidos	4.2 O sistema emite mensagem ao usuário, informando-o que há campos obrigatórios não preenchidos; 4.3 O sistema retorna para a tela de “eventos” em forma de edição, para que o usuário informe os campos obrigatórios não preenchidos.

Quadro 11 – Expansão do caso de uso gerenciar eventos

O registro de diários de classe é apresentado no Quadro 12.

Caso de uso: Registrar diários de classe **Descrição:** Essa rotina serve para descrever as atividades trabalhadas com a turma na data informada. **Evento iniciador:** Ator clica sobre a rotina “Diário de Classe”. **Atores:** Profissional responsável pela turma. **Pré condição:** Possuir turmas e profissionais cadastrados no sistema. **Sequência de eventos:** 1. Ator abre a rotina de “Diários de Classe”; 2. Sistema apresenta a tela; 3. Ator informa o profissional; 4. Ator informa a data; 5. Ator seleciona a turma; 6. Ator descreve, no campo “descrição”, as observações pertinentes a aula dada na data informada; 7. Sistema insere o registro do diário de classe no banco de dados. **Pós condição:** Registro do diário de classe.	
Nome do fluxo alternativo (expansão)	**Descrição**
Linha 5: Apresentar apenas as turmas do profissional selecionado	5.1 Ao carregar a consulta de turmas, apresentar apenas as turmas em que o profissional ministra aulas.
Linha 7: Dados obrigatórios não preenchidos	7.1 Ao salvar a rotina, o sistema valida se todos os campos obrigatórios foram informados; 7.2 O sistema emite mensagem ao usuário, informando-o que há campos obrigatórios não preenchidos; 7.3 O sistema retorna para a tela de “diário de classe” em forma de edição, para que o usuário informe os campos obrigatórios não preenchidos.

Quadro 12 – Expansão do caso de uso registrar diários de classe

A Figura 2 contém o diagrama de entidade e relacionamento, que representa o banco de dados da aplicação.

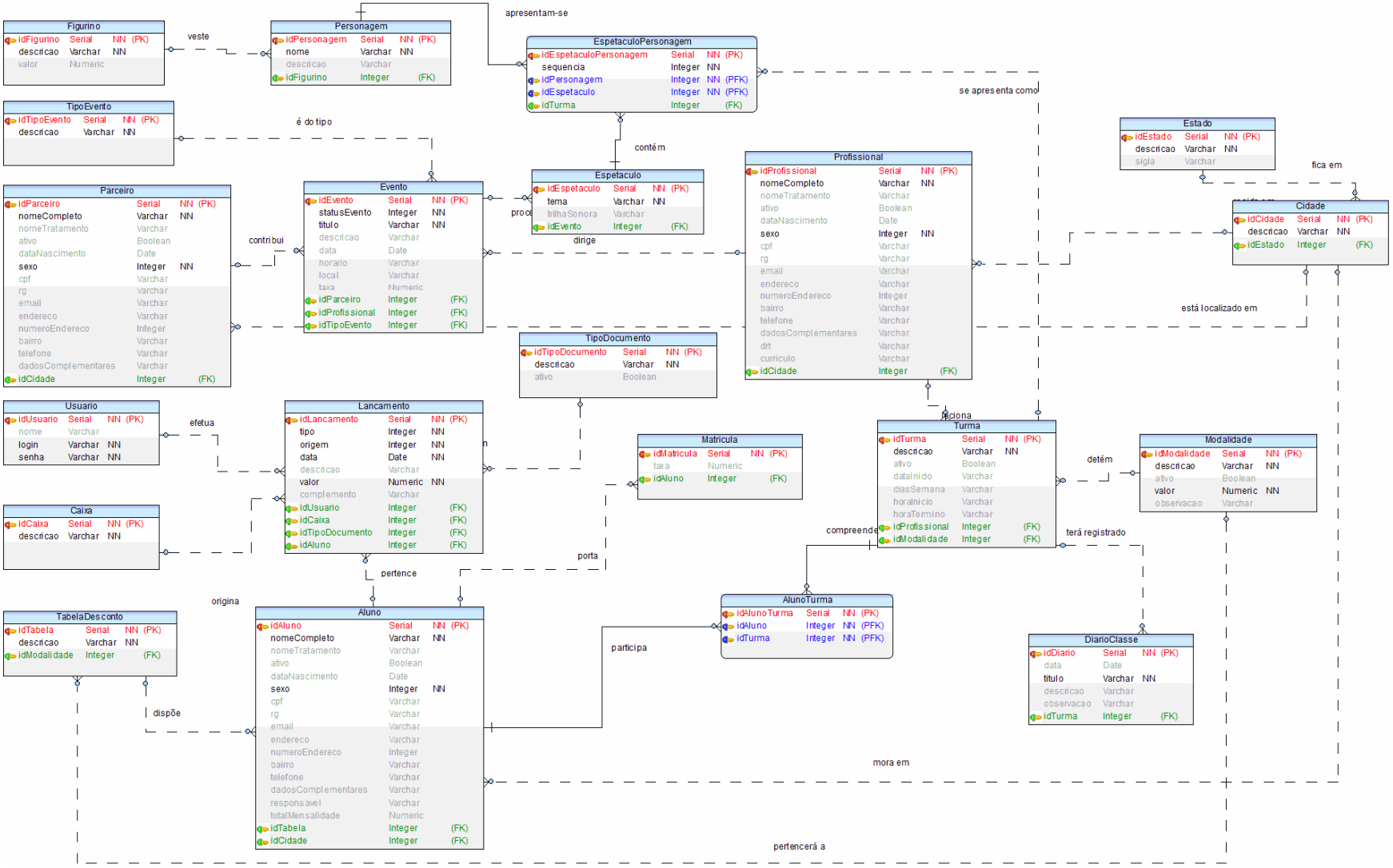

Figura 2 – Diagrama de entidades e relacionamentos do banco de dados

No Quadro 13 estão representados os campos da tabela “Figurino”.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idFigurino	Serial	Não	Sim	Sim	
descricao	Texto	Não	Não	Não	
valor	Numérico	Sim	Não	Não	

Quadro 13 – Campos da tabela Figurinos

O Quadro 14 apresenta a tabela “Personagem” que vestirá um figurino.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idPersonagem	Serial	Não	Sim	Não	
nome	Texto	Não	Não	Não	
descricao	Texto	Sim	Não	Não	
idFigurino	Numérico	Não	Não	Sim	Tabela Figurino

Quadro 14 – Campos da tabela Personagem

O Quadro 15 apresenta os campos da tabela “Espetaculo”. Um espetáculo é composto por personagens.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idEspetaculo	Serial	Não	Sim	Não	
tema	Texto	Não	Não	Não	
trilhaSonora	Texto	Sim	Não	Não	
idPersonagem	Numérico	Não	Não	Sim	Tabela Personagem
idEvento	Numérico	Não	Não	Sim	Tabela Evento

Quadro 15 – Campos da tabela Espetaculo

O Quadro 16 apresenta a tabela “PersonagemEspetaculo” que vincula os personagens relacionados a um determinado espetáculo.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idEspetaculo	Serial	Não	Sim	Sim	Tabela Espetaculo
idPersonagem	Serial	Não	Sim	Sim	Tabela Personagem

Quadro 16 – Campos da tabela PersonagemEspetaculo

O Quadro 17 contém os campos da tabela “Eventos”. Um evento poderá contar com um patrocinador (que é um parceiro da escola) e é dirigido por um profissional da escola.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idEvento	Serial	Não	Sim	Não	
tipoEvento	Numérico	Não	Não	Não	
statusEvento	Numérico	Não	Não	Não	
titulo	Texto	Não	Não	Não	
descricao	Texto	Sim	Não	Não	
data	Data	Sim	Não	Não	
horario	Hora	Sim	Não	Não	
local	Texto	Sim	Não	Não	
taxa	Numérico	Sim	Não	Não	
idParceiro	Numérico	Não	Não	Sim	Tabela Parceiro
idProfissional	Numérico	Não	Não	Sim	Tabela Profissional

Quadro 17 – Campos da tabela Eventos

O Quadro 18 apresenta a tabela “Parceiro”.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idParceiro	Serial	Não	Sim	Não	
nomeCompleto	Texto	Não	Não	Não	
nomeTratamento	Numérico	Não	Não	Não	
ativo	Booleano	Sim	Não	Não	
dataNascimento	Data	Sim	Não	Não	
sexo	Numérico	Não	Não	Não	
cpf	Texto	Sim	Não	Não	
rg	Texto	Sim	Não	Não	
email	Texto	Sim	Não	Não	
endereco	Texto	Sim	Não	Não	
numeroEndereco	Numérico	Sim	Não	Não	
bairro	Texto	Sim	Não	Não	
idCidade	Texto	Sim	Não	Sim	Tabela Cidade
telefone	Texto	Sim	Não	Não	
dadosComplementares	Texto	Sim	Não	Não	

Quadro 18 – Campos da tabela Parceiro

Os campos da tabela “Profissional” são apresentados no Quadro 19. Um profissional é quem ministra as aulas das diferentes modalidades de dança na escola.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idProfissional	Serial	Não	Sim	Não	
nomeCompleto	Texto	Não	Não	Não	
nomeTratamento	Numérico	Não	Não	Não	
ativo	Booleano	Sim	Não	Não	
dataNascimento	Data	Sim	Não	Não	
sexo	Numérico	Não	Não	Não	
cpf	Texto	Sim	Não	Não	
rg	Texto	Sim	Não	Não	
email	Texto	Sim	Não	Não	
endereco	Texto	Sim	Não	Não	
numeroEndereco	Numérico	Sim	Não	Não	
bairro	Texto	Sim	Não	Não	
IdCidade	Texto	Sim	Não	Sim	Tabela Cidade
telefone	Texto	Sim	Não	Não	
dadosComplementares	Texto	Sim	Não	Não	
drt	Texto	Sim	Não	Não	
curriculo	Texto	Sim	Não	Não	

Quadro 19 – Campos da tabela Profissional

O Quadro 20 apresenta a tabela “Usuario”, que são as pessoas que têm acesso ao sistema.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idUsuario	Serial	Não	Sim	Não	
nome	Texto	Sim	Não	Não	
login	Texto	Não	Não	Não	
senha	Texto	Sim	Não	Não	

Quadro 20 – Campos da tabela Usuario

O Quadro 21 apresenta a tabela “Caixa” que conterá os lançamentos de receitas e despesas.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idCaixa	Serial	Não	Sim	Não	
descricao	Texto	Não	Não	Não	

Quadro 21 – Campos da tabela Caixa

O Quadro 22 apresenta as informações da tabela “Lancamento”, que são as movimentações financeiras que a escola faz.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idLancamento	Serial	Não	Sim	Não	
tipo	Numérico	Não	Não	Não	
origem	Numérico	Não	Não	Não	
data	Data	Não	Não	Não	
descricao	Texto	Sim	Não	Não	
valor	Numérico	Não	Não	Não	
complemento	Texto	Sim	Não	Não	
idUsuario	Numérico	Não	Não	Sim	Tabela Usuario
idCaixa	Numérico	Não	Não	Sim	Tabela Caixa
idTipoDocumento	Numérico	Não	Não	Sim	Tabela TipoDocumento
idAluno	Numérico	Não	Não	Sim	Tabela Aluno

Quadro 22 – Campos da tabela Lancamentos

No Quadro 23, temos a representação da tabela “TipoDocumento”, que indica o tipo do lançamento de caixa: se é um recebimento de mensalidades, de matrículas, de taxas para inscrição em eventos da escola, etc.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idTipoDocumento	Serial	Não	Sim	Não	
descricao	Texto	Não	Não	Não	
ativo	Booleano	Sim	Não	Não	

Quadro 23 – Campos da tabela TipoDocumento

O Quadro 24 apresenta as informações da tabela “Aluno”, que são as pessoas que praticam as modalidades de dança ofertadas pela escola. Esse aluno poderá ter uma tabela de desconto para aplicar a suas mensalidades, dependendo da quantidade de modalidades que ele pratica.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idAluno	Serial	Não	Sim	Não	
nomeCompleto	Texto	Não	Não	Não	
nomeTratamento	Numérico	Não	Não	Não	
ativo	Booleano	Sim	Não	Não	
dataNascimento	Data	Sim	Não	Não	
sexo	Numérico	Não	Não	Não	
cpf	Texto	Sim	Não	Não	
rg	Texto	Sim	Não	Não	
email	Texto	Sim	Não	Não	
endereco	Texto	Sim	Não	Não	
numeroEndereco	Numérico	Sim	Não	Não	
bairro	Texto	Sim	Não	Não	
IdCidade	Texto	Sim	Não	Sim	Tabela Cidade
telefone	Texto	Sim	Não	Não	
dadosComplementares	Texto	Sim	Não	Não	
responsavel	Texto	Sim	Não	Não	
idTabela	Numérico	Não	Não	Sim	Tabela TabelaDesconto

Quadro 24 – Campos da tabela Aluno

No Quadro 25, está representada a tabela “TabelaDesconto”, que é relacionada com o cadastro do aluno. Alunos podem ter desconto em sua mensalidade, caso pratiquem determinada quantidade de modalidades de dança.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idTabela	Serial	Não	Sim	Não	
descricao	Texto	Não	Não	Não	
idModalidade	Numérico	Não	Não	Sim	Tabela Modalidade

Quadro 25 – Campos da tabela TabelaDesconto

O Quadro 26 representa a tabela “Matricula”, que é vinculada ao aluno da escola.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idMatricula	Serial	Não	Sim	Não	
taxa	Numérico	Sim	Não	Não	
idAluno	Numérico	Não	Não	Sim	Da tabela Aluno

Quadro 26 – Campos da tabela Matricula

No Quadro 27 está a representação da tabela “Modalidade”, que são os tipos de dança ofertados pela escola.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idModalidade	Serial	Não	Sim	Não	
descricao	Texto	Não	Não	Não	
ativo	Booleano	Sim	Não	Não	
descricao	Texto	Sim	Não	Não	
valor	Numérico	Não	Não	Não	
observacao	Texto	Sim	Não	Não	

Quadro 27 – Campos da tabela Modalidade

Os campos da tabela “Turma” e suas informações estão no Quadro 28. A uma turma é relacionado um profissional, que será responsável por ministrar as aulas.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idTurma	Serial	Não	Sim	Não	
descricao	Texto	Não	Não	Não	
ativo	Booleano	Sim	Não	Não	
dataInicio	Data	Sim	Não	Não	
horaInicio	Hora	Sim	Não	Não	
horaTermino	Hora	Sim	Não	Não	
idProfissional	Numérico	Não	Não	Sim	Da tabela Profissional
idModalidade	Numérico	Não	Não	Sim	Da tabela Modalidade

Quadro 28 – Campos da tabela Turma

O Quadro 29 apresenta a tabela “AlunoTurma”, que é uma relação do tipo M:N, que significa “muitos para muitos”. Uma turma pode ter vários alunos; um aluno pode participar de várias turmas; por isso, a necessidade de relacionar o aluno com suas respectivas turmas.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idAlunoTurma	Serial	Não	Sim	Não	
idAluno	Numérico	Não	Não	Sim	Da tabela Aluno
idTurma	Numérico	Não	Não	Sim	Da tabela Turma

Quadro 29 – Campos da tabela AlunoTurma

O Quadro 30 apresenta as informações da tabela “DiarioClasse”, que contém as anotações do profissional para a turma desejada. Ele pode fazer anotações do tipo: conteúdo repassado em sala, planejamento para o semestre, evolução dos alunos, etc.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idDiario	Serial	Não	Sim	Não	
data	Data	Sim	Não	Não	
titulo	Texto	Não	Não	Não	
descricao	Texto	Sim	Não	Não	
observacao	Texto	Sim	Não	Não	
idTurma	Numérico	Não	Não	Sim	Da tabela Turma

Quadro 30 – Campos da tabela DiarioClasse

O Quadro 31 apresenta a tabela “Cidade” que é utilizada para os cadastros que envolvam endereço.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idCidade	Serial	Não	Sim	Não	
descricao	Texto	Sim	Não	Não	
idEstado	Numérico	Não	Não	Sim	Da tabela Estado

Quadro 31 – Campos da tabela Cidade

O Quadro 32 apresenta a tabela “Estado” que armazena das Unidades de Federação às quais pertencem as cidades.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
idEstado	Serial	Não	Sim	Não	
descricao	Texto	Não	Não	Não	

sigla	Texto	Não	Não	Não	

Quadro 32 – Campos da tabela Estado

4.3 APRESENTAÇÃO DO SISTEMA

O leiaute do sistema é composto pelo menu superior e a área de trabalho, como mostra a Figura 3.

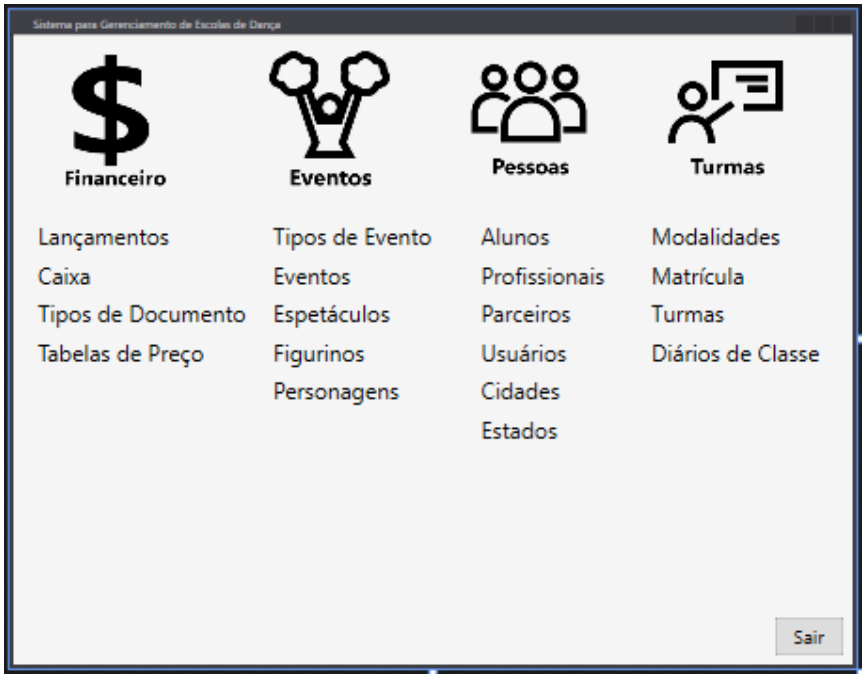

Figura 3 – Tela inicial do sistema

Antes de efetuar matrículas e realizar configurações de eventos e espetáculos, é necessário realizar alguns cadastros, como o de profissionais e de modalidade, que posteriormente serão relacionados a uma turma, por exemplo. A tela da Figura 4 apresenta o formulário de cadastro de profissionais.

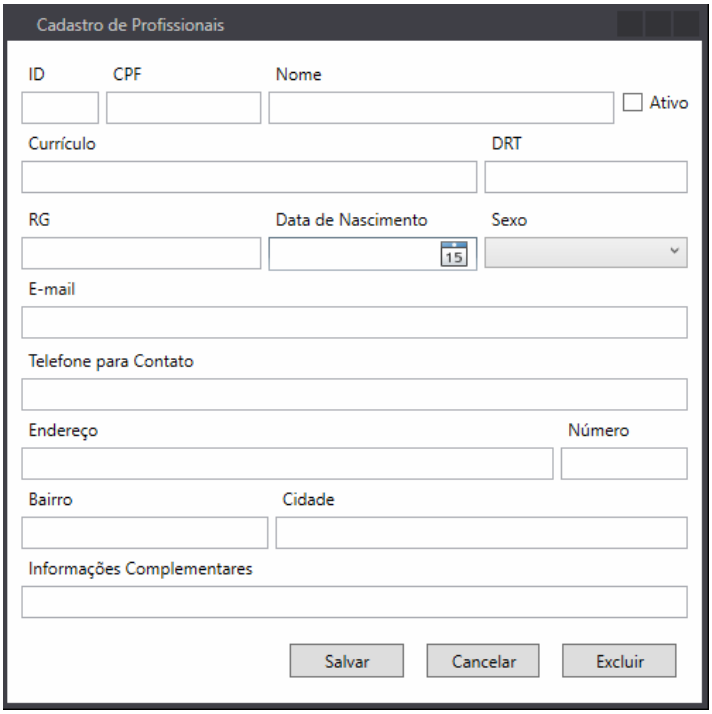

Figura 4 – Formulário de cadastro de profissionais

Para pesquisar dados já inseridos, foi implementado a tela de consulta de registros que é chamado quando teclar F2 em cima do campo ID de qualquer formulário que possua esse campo. Para facilitar a identificação dessa funcionalidade, ao passar o cursor do mouse em cima do campo ID, é apresentado um *hint* para indicar a função. A região circulada da Figura 5 apresenta essa funcionalidade.

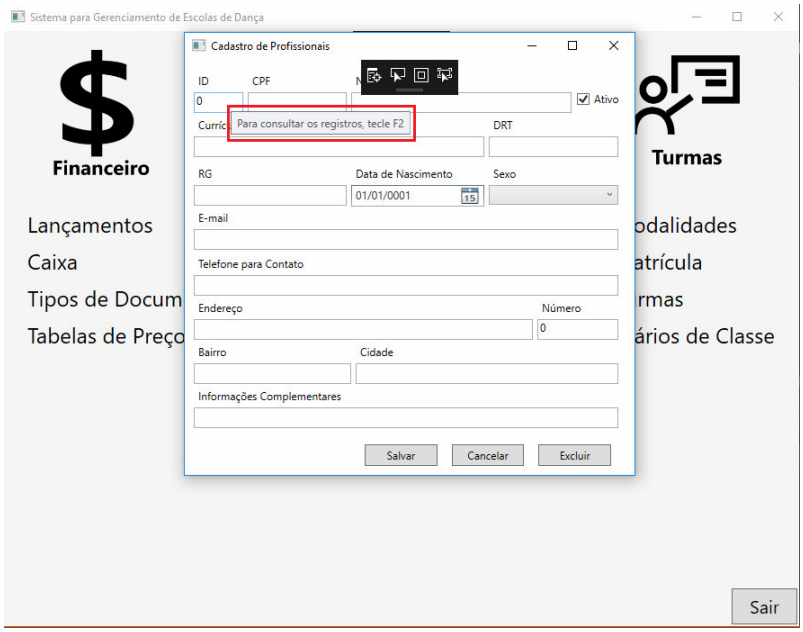

Figura 5 - Hint do campo ID que indica que o F2 abre janela de consulta

Sendo assim, ao teclar F2 no campo ID, o sistema apresentará a tela de consulta de registros, conforme indica a Figura 6.

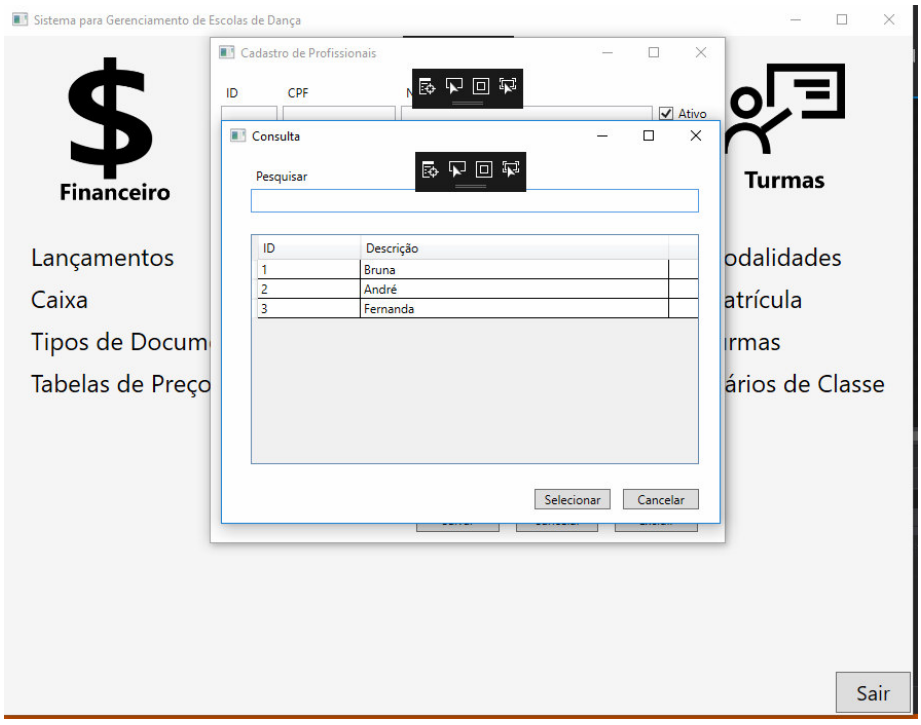

Figura 6 - Tela de consulta de profissionais, apresentada ao teclar F2 no campo ID

Também é necessário cadastrar as modalidades de dança oferecidas pela escola (como mostra o formulário da Figura 7).

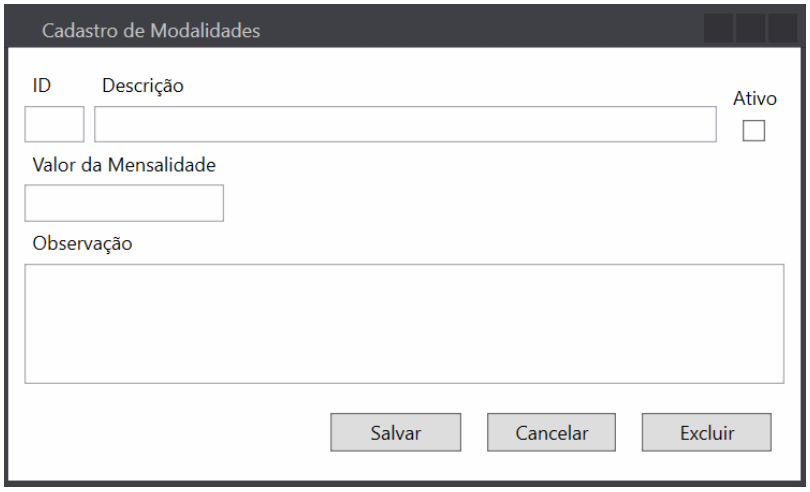

Figura 7 – Formulário de cadastro de modalidades de dança

Tendo os cadastros base, é possível cadastrar as turmas de determinada modalidade, que determinado profissional irá lecionar. A Figura 8 apresenta essa rotina.

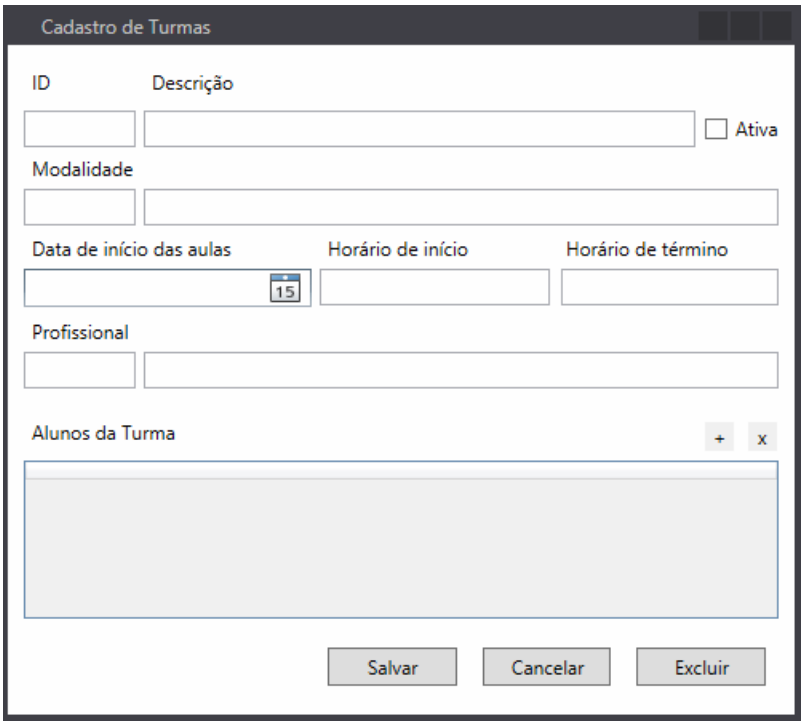

Figura 8 – Formulário de cadastro de turmas

Para efetuar uma matrícula, é necessário escolher a opção “Matrícula”, no menu “Turmas”. A Figura 9 apresenta a tela responsável por gravar esse registro.

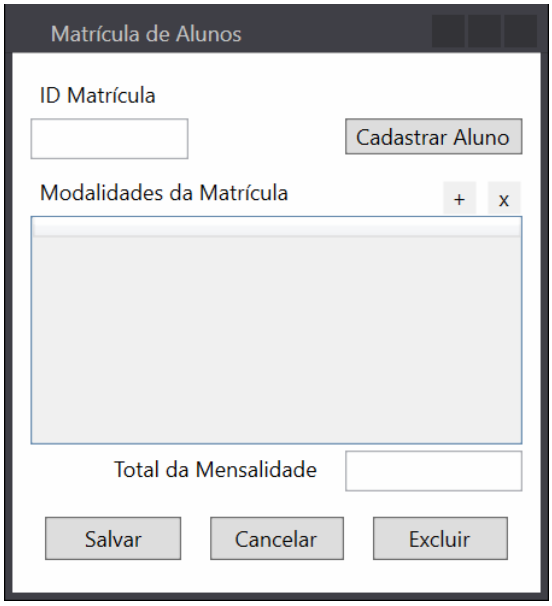

Figura 9 – Formulário de cadastro de matrícula

Ao clicar no botão “Cadastrar Aluno”, o sistema apresentará a tela para inclusão dos dados do aluno, conforme Figura 10. Essa tela também pode ser acessada pelo menu “Pessoas”, na opção “Cadastro de Aluno”.

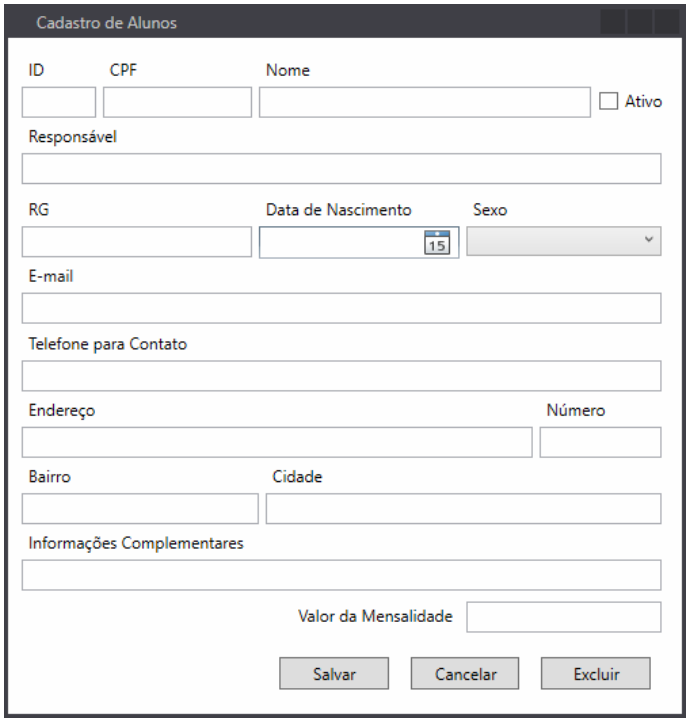

Figura 10 – Formulário de cadastro de alunos

A Figura 11 apresenta a tela de inclusão e configuração de eventos.

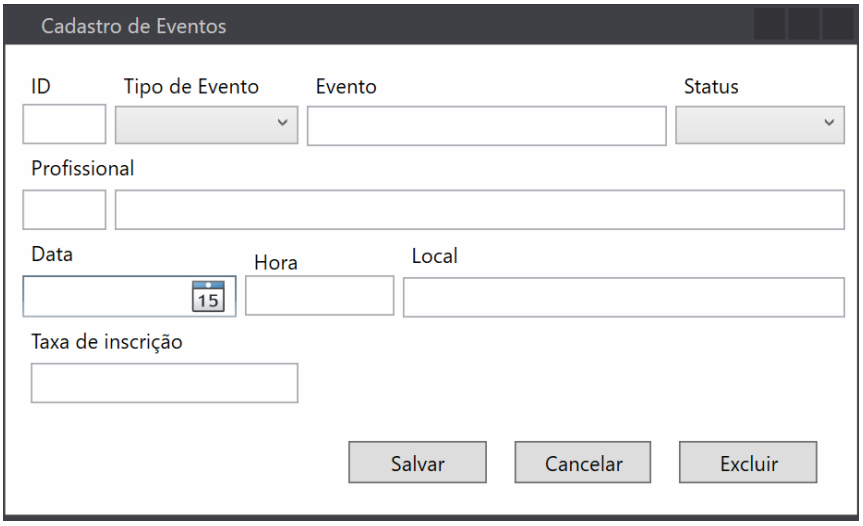

Figura 11 – Formulário de cadastro de eventos

A tela responsável pela configuração do espetáculo é apresentada na Figura 12.

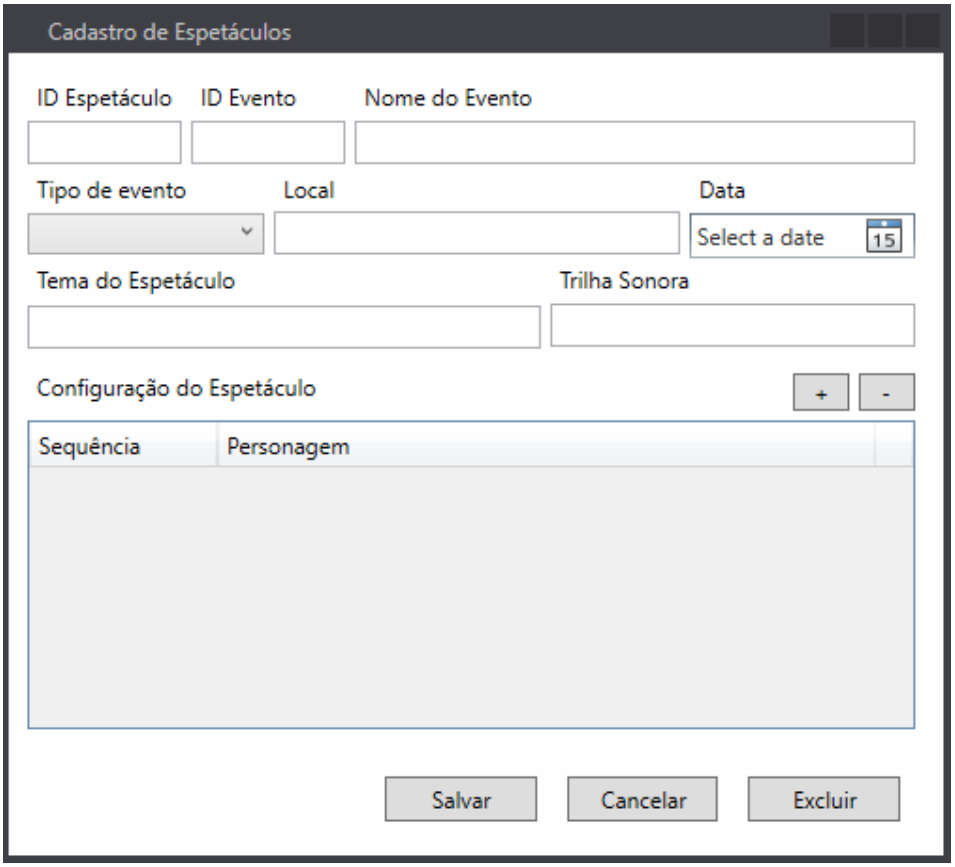

Figura 12 – Formulário de cadastro de eventos

Ao clicar no botão “+” da tela da Figura 12, o sistema apresentará outra tela para selecionar o personagem e a turma ou aluno (Figura 13). Nela, será necessário informar um número de sequência para apresentação e selecionar um personagem e uma turma, através da tela de consulta que é chamada pelo atalho F2 ao clicar no campo logo abaixo do identificar Personagem e Turma.

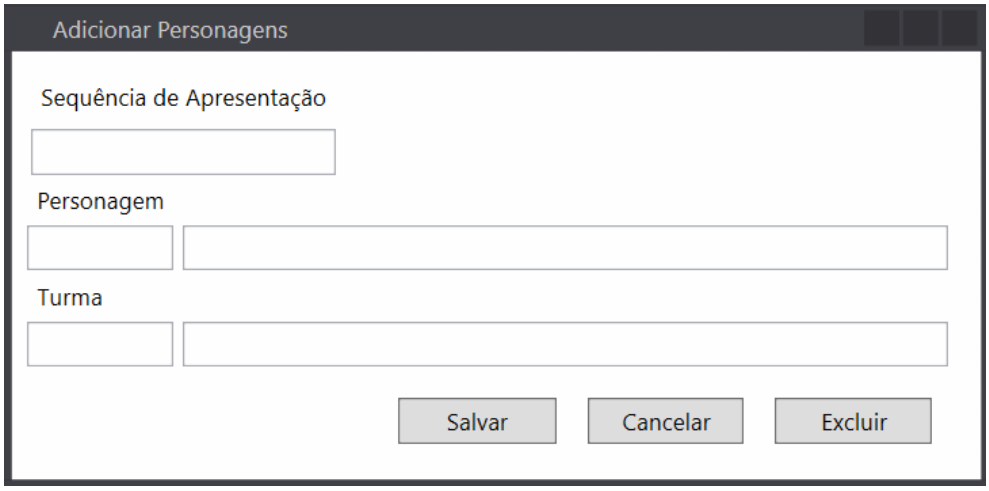

Figura 13 – Formulário de cadastro de personagem

4.4 DESENVOLVIMENTO DO SISTEMA

O padrão de desenvolvimento adotado para esse projeto foi o *View-Model-ViewModel* (MVVM), que separa, de forma clara, elementos de regra de negócio (*Model*), apresentação (*View*) e lógica de apresentação (*ViewModel*). Sendo assim, o sistema é composto por uma solução que é dividida em três projetos: View, Model e View Model, conforme Figura 14.

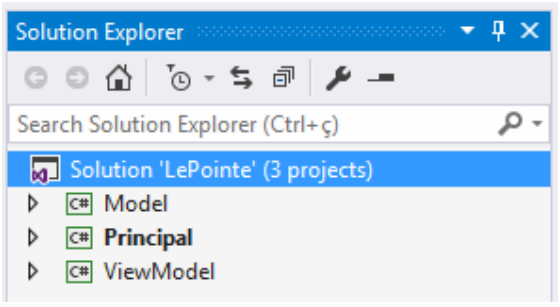

Figura 14 – Organização dos projetos da aplicação

O primeiro projeto abrange o modelo da aplicação (*Model*) e contém as entidades, enumerados, mapeamento dos objetos e classe de acesso ao banco de dados, conforme apresenta a Figura 15.

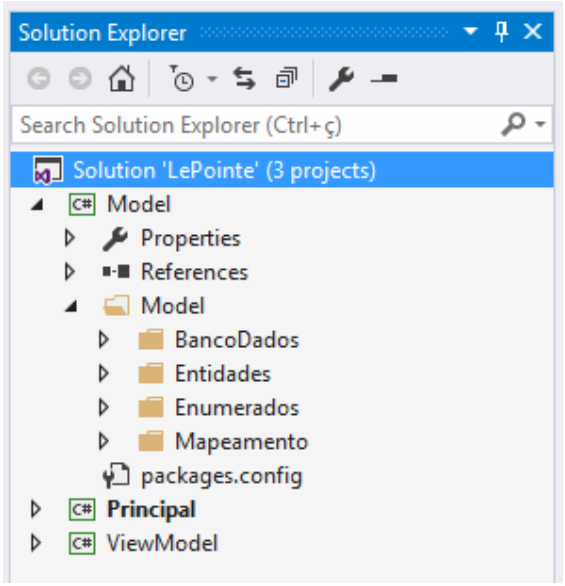

Figura 15 – Projeto modelo da aplicação

O segundo projeto, chamado de “Principal”, comporta a apresentação da aplicação, ou seja, as telas do sistema, juntamente com o controle padrão utilizado por elas. A Figura 16 representa sua organização.

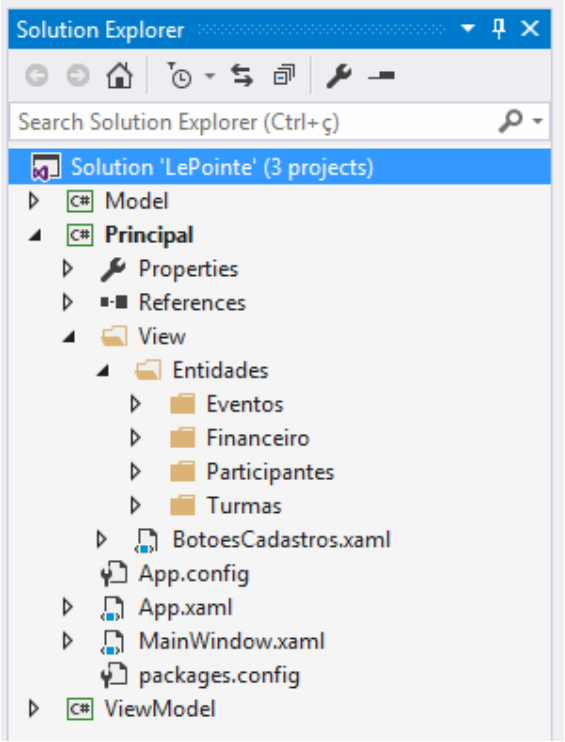

Figura 16 – Projeto principal da aplicação

O terceiro projeto abrange a lógica de apresentação. É composta pelas classes ViewModel, que descende de uma classe base, que possui os métodos abstratos das operações com o banco de dados (salvar, excluir e editar). Esse projeto é representado pela Figura 17.

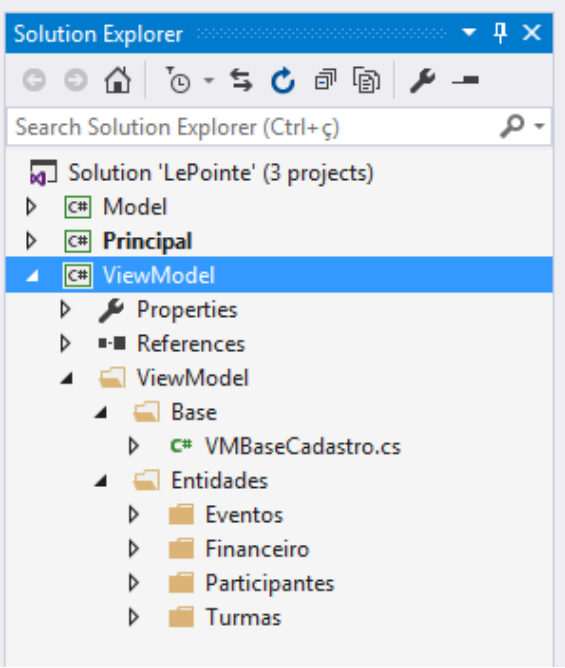

Figura 17 – Projeto lógica da aplicação

A tela principal é composta de um menu, sendo seu código *Extensible Application Markup Language* (XAML) da seguinte maneira: depois de estruturado a aplicação no padrão MVVM, inicia-se a codificação das classes base para o funcionamento do sistema.

O primeiro passo foi criar a sessão de conexão com banco, que se dá pela implementação da classe “SessionFactory”, apresentada na Listagem 1.

```
namespace Model.Model.BancoDados
{
    public static class SessionFactory
    {
        private static ISessionFactory sessionFactory;
        public static ISessionFactory AcessoSessionFactory { get {
return sessionFactory; } }
        public static void Inicializar(ConexaoBD conexao)
        {
            var mapper = new ModelMapper();
mapper.AddMappings(Assembly.GetExecutingAssembly().GetExportedTypes(
));
            HbmMapping mapDominio =
mapper.CompileMappingForAllExplicitlyAddedEntities();
            mapDominio.autoimport = false;
            var configuration = new Configuration();
            configuration.DataBaseIntegration(c =>
            {
                c.Driver<NpgsqlDriver>();
                c.ConnectionString = conexao.DescConexao;
                c.Dialect<PostgreSQL82Dialect>();
```

```
                c.Batcher<NonBatchingBatcherFactory>();
                c.BatchSize = short.MaxValue;
                c.LogFormattedSql = true;
                c.LogSqlInConsole = false;
            });
            configuration.AddMapping(mapDominio);
            sessionFactory = configuration.BuildSessionFactory();
        }
    }
}
```

Listagem 1 – Implementação da classe SessionFactory

Criada a sessão de conexão foi criada a *string* de conexão ao banco de dados, implementada na classe “ConexaoBD”, conforme o código apresentado na Listagem 2.

```
namespace Model.Model.BancoDados
{
    public class ConexaoBD
    {
        static string server = "127.0.0.1";
        static string porta = "5432";
        static string user = "postgres";
        static string password = "12345";
        static string nomeBanco = "eDance";
        public ConexaoBD()
        {
            DescConexao = String.Format("Server={0};Port={1};User
Id={2};Password={3};Database={4};",
                server, porta, user, password, nomeBanco);
        }
        public string DescConexao { get; private set; }
    }
}
```

Listagem 2 – Implementação da classe ConexaoBD

Tendo a conexão com o banco de dados preparada, foi implementada a classe de acesso aos dados. Todas as entidades do sistema possuem uma propriedade em comum: a chave primária. Por esse motivo, todas descendem de uma classe genérica chamada “EntidadeComum” (código apresentado na Listagem 3).

```
namespace LePointe.Model.Entidades
{
    public class EntidadeComum
    {
        private long id;
        public virtual long ID
        {
            get { return id; }
            set { id = value; }
```

```
            }
        }
    }
```

Listagem 3 – Código para criação de entidade primária

Em seguida foi necessário decidir como as chaves primárias dos objetos do banco seriam geradas. Optou-se por declarar o tipo de dado das chaves primárias como “serial”, para que o PostgreSQL gerasse o arquivo de sequência para essas chaves, utilizando assim, o mecanismo de auto incremento do próprio banco de dados.

Como todas as chaves primárias possuem o prefixo “id” concatenado com o nome da tabela, a situação foi resolvida com a implementação apresentada na Listagem 4.

```
    namespace LePointe.Model.Mapeamento.Map.Entidades
    {
        public class BaseMapping<TEntidade> :
ClassMapping<TEntidade>
            where TEntidade : EntidadeComum, new()
        {
            public BaseMapping(string tableName)
            {
                Table(tableName);
                Id(x => x.ID, m =>
                {
                    m.Column("id" + tableName);
                    m.Access(Accessor.Field);
                    m.UnsavedValue(0);
                    m.Generator(Generators.Sequence, g =>
g.Params(new
                    {
                        sequence = $"{tableName}_id{tableName}_seq"
                    }));
                });
            }
        }
    }
```

Listagem 4 – Geração dos dados para campos chave primárica

Em seguida foram criados os arquivos de mapeamento dos objetos do banco de dados. A Listagem 5 apresenta a codificação da tabela “Figurino”:

```
    public class FigurinoMapeamento : BaseMapping<Figurino>
        {
            public FigurinoMapeamento()
                : base("figurino")
            {
                Property(x => x.Descricao, m =>
                {
```

```
                    m.Column("descricao");
                    m.Access(Accessor.Field);
                });
                Property(x => x.Valor, m =>
                {
                    m.Column("valor");
                    m.Access(Accessor.Field);
                });
            }
        }
```

Listagem 5 – Codificação da tabela Figurino

Tendo a base do sistema pronta, as telas foram implementadas. A tela principal é composta de um menu, sendo seu código XAML apresentado na Listagem 6.

```
<Window x:Class="View.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        xmlns:local="clr-namespace:View"
        mc:Ignorable="d"
        Title="Sistema para Gerenciamento de Escolas de Dança"
Height="700" Width="900">
    <Grid Background="WhiteSmoke">
        <Image x:Name="imgFinanceiro" HorizontalAlignment="Left"
Height="139" Margin="44,21,0,0" VerticalAlignment="Top" Width="122"
Source="View/Imagens/financeiro.png"/>
        <Image x:Name="imgEventos" HorizontalAlignment="Left"
Height="150" Margin="272,10,0,0" VerticalAlignment="Top" Width="140"
Source="View/Imagens/eventos.png"/>
        <Image x:Name="imgPessoas" HorizontalAlignment="Left"
Height="150" Margin="487,10,0,0" VerticalAlignment="Top" Width="122"
Source="View/Imagens/pessoas.png"/>
        <Image x:Name="imgTurmas" HorizontalAlignment="Left"
Height="150" Margin="696,10,0,0" VerticalAlignment="Top" Width="122"
Source="View/Imagens/turmas.png"/>
        <Button x:Name="btnSair" Content="Sair"
HorizontalAlignment="Left" Margin="809,619,0,0"
VerticalAlignment="Top" Width="73" Height="40" FontSize="20"
Click="btnSair_Click"/>

        <Button x:Name="btnLancamentos" Content="Lançamentos"
HorizontalAlignment="Left" Margin="24,196,0,0"
VerticalAlignment="Top" Width="233" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
Click="btnLancamentos_Click" BorderBrush="{x:Null}"/>
        <Button x:Name="btnCaixa" Content="Caixa"
HorizontalAlignment="Left" Margin="24,237,0,0"
VerticalAlignment="Top" Width="233" FontSize="24"
```

```
HorizontalContentAlignment="Left" Background="WhiteSmoke"
Click="btnCaixa_Click" BorderBrush="{x:Null}"/>
        <Button x:Name="btnTiposDocumento" Content="Tipos de
Documento" HorizontalAlignment="Left" Margin="24,278,0,0"
VerticalAlignment="Top" Width="233" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
Click="btnTiposDocumento_Click" BorderBrush="{x:Null}"/>
        <Button x:Name="btnTabelaPreco" Content="Tabelas de Preço"
HorizontalAlignment="Left" Margin="24,319,0,0"
VerticalAlignment="Top" Width="233" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}"/>

        <Button x:Name="btnEventos" Content="Eventos"
HorizontalAlignment="Left" Margin="274,237,0,0"
VerticalAlignment="Top" Width="140" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}" Click="btnEventos_Click"/>
        <Button x:Name="btnEspetaculos" Content="Espetáculos"
HorizontalAlignment="Left" Margin="274,278,0,0"
VerticalAlignment="Top" Width="146" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}" Click="btnEspetaculos_Click"/>
        <Button x:Name="btnFigurinos" Content="Figurinos"
HorizontalAlignment="Left" Margin="274,319,0,0"
VerticalAlignment="Top" Width="146" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}" Click="btnFigurinos_Click"/>
        <Button x:Name="btnPersonagens" Content="Personagens"
HorizontalAlignment="Left" Margin="274,360,0,0"
VerticalAlignment="Top" Width="146" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}" Click="btnPersonagens_Click"/>

        <Button x:Name="btnAlunos" Content="Alunos"
HorizontalAlignment="Left" Margin="495,196,0,0"
VerticalAlignment="Top" Width="150" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}" Click="btnAlunos_Click" />
        <Button x:Name="btnProfissionais" Content="Profissionais"
HorizontalAlignment="Left" Margin="495,237,0,0"
VerticalAlignment="Top" Width="150" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}" Click="btnProfissionais_Click" />
        <Button x:Name="btnParceiros" Content="Parceiros"
HorizontalAlignment="Left" Margin="495,278,0,0"
VerticalAlignment="Top" Width="150" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}" Click="btnParceiros_Click" />
        <Button x:Name="btnUsuarios" Content="Usuários"
HorizontalAlignment="Left" Margin="495,319,0,0"
VerticalAlignment="Top" Width="146" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}" Click="btnUsuarios_Click" />
```

```
        <Button x:Name="btnModalidades" Content="Modalidades"
HorizontalAlignment="Left" Margin="676,196,0,0"
VerticalAlignment="Top" Width="189" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}" Click="btnModalidades_Click"/>
        <Button x:Name="btnMatricula" Content="Matrícula"
HorizontalAlignment="Left" Margin="676,237,0,0"
VerticalAlignment="Top" Width="189" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}" Click="btnMatricula_Click"/>
        <Button x:Name="btnTurmas" Content="Turmas"
HorizontalAlignment="Left" Margin="676,278,0,0"
VerticalAlignment="Top" Width="189" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}" Click="btnTurmas_Click"/>
        <Button x:Name="btnDiariosClasse" Content="Diários de
Classe" HorizontalAlignment="Left" Margin="676,319,0,0"
VerticalAlignment="Top" Width="189" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}" Click="btnDiariosClasse_Click"/>

        <Button x:Name="btnCidades" Content="Cidades"
HorizontalAlignment="Left" Margin="495,360,0,0"
VerticalAlignment="Top" Width="150" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}" Click="btnCidades_Click"/>
        <Button x:Name="btnEstados" Content="Estados"
HorizontalAlignment="Left" Margin="495,401,0,0"
VerticalAlignment="Top" Width="150" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}" Click="btnEstados_Click"/>
        <Button x:Name="btnTipoEventos" Content="Tipos de Evento"
HorizontalAlignment="Left" Margin="274,196,0,0"
VerticalAlignment="Top" Width="178" FontSize="24"
HorizontalContentAlignment="Left" Background="WhiteSmoke"
BorderBrush="{x:Null}" RenderTransformOrigin="0.318,-5.639"
Click="btnTipoEventos_Click"/>
    </Grid>
</Window>
```

Listagem 6 – Código XML do menu

Para cada item do menu, foi definido, no evento “Click”, a criação da View e da ViewModel de cada tela, exigência do próprio *Windows Presentation Foundation* (WPF). Sendo assim, a Listagem 7 apresenta a implementação da criação e chamada dessas classes.

```
namespace View
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
```

```
            public MainWindow()
            {
                InitializeComponent();
            }
            private void mniFigurino_Click(object sender,
RoutedEventArgs e)
            {
                var view = new VFigurino();
                var vm = new VMFigurino();
                view.DataContext = vm;
                view.ShowDialog();
            }
```

Listagem 7 – View e model de cada tela

Para as telas de cadastro houve a necessidade de criar um *User Control* que contém as principais operações a realizar: salvar registro, alterar, excluir e cancelar. O conceito de comportamento dessas operações é:

- O usuário informa os dados sem pesquisar: assume a operação “novo cadastro”;
- O usuário pesquisa um dado já existente, utilizado o código “ID”: assume a operação “alterar cadastro”;
- O usuário pesquisa um dado já existente e clica no botão “cancelar”: assume a operação “cancelar”, ou seja, não grava alterações e os campos da tela são limpos;
- O usuário pesquisa um dado já existente e clica no botão “excluir”: assume a operação “excluir cadastro”.

A invocação dessas operações é apresentada na Listagem 8.

```
      <UserControl x:Class="LePointe.View.BotoesCadastros"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
                  xmlns:local="clr-namespace:LePointe.View"
                  mc:Ignorable="d"
                  d:DesignHeight="300" d:DesignWidth="300"
Height="40">
          <Grid>
              <Grid.ColumnDefinitions>
                  <ColumnDefinition Width="*"/>
                  <ColumnDefinition Width="*"/>
                  <ColumnDefinition Width="*"/>
              </Grid.ColumnDefinitions>
              <Button x:Name="btnSalvar" Content="Salvar"
Grid.Column="0" Margin="8" Click="btnSalvar_Click"/>
```

```
            <Button x:Name="btnCancelar" Content="Cancelar"
Grid.Column="1" Margin="8" Click="btnCancelar_Click"/>
            <Button x:Name="btnExcluir" Content="Excluir"
Grid.Column="2" Margin="8" Click="btnExcluir_Click"/>
        </Grid>
    </UserControl>

    public override void SalvarCadastro()
            {
                if (Cadastro == null)
                    return;
                using (var sessao =
SessionFactory.AcessoSessionFactory.OpenSession())
                    using(var tx = sessao.BeginTransaction())
                {
                    sessao.SaveOrUpdate(Cadastro);
                    tx.Commit();
                }
                CancelarCadastro();
            }

            public override void CancelarCadastro()
            {
                Cadastro = null;
                OnPropertyChanged("");
            }

            public override void ExcluirCadastro()
            {
                using (var sessao =
SessionFactory.AcessoSessionFactory.OpenSession())
                using (var tx = sessao.BeginTransaction())
                {
                    sessao.Delete(Cadastro);
                    tx.Commit();
                }
                CancelarCadastro();
            }
```

Listagem 8 – Código UserControl

Como essas operações são padrões e existentes em todas as telas, os métodos são declarados como abstratos em uma classe chamada "VMBaseCadastro" e todas as *viewmodels* do sistema descendem dela. Por esse motivo, elas devem sobrescrevê-las. A Listagem 9 apresenta a declaração desses métodos dentro da classe VMBaseCadastro.

```
namespace ViewModel.ViewModel.Base
{
    public abstract class VMBaseCadastro : INotifyPropertyChanged
    {
        public void OnPropertyChanged(string nomePropriedade)
        {
            if (PropertyChanged != null)
                PropertyChanged(this, new
PropertyChangedEventArgs(nomePropriedade));
        }
        public event PropertyChangedEventHandler PropertyChanged;
        public abstract bool ValidarCadastro();
        public abstract void SalvarCadastro();
        public abstract void CancelarCadastro();
        public abstract void ExcluirCadastro();
    }
```

Listagem 9 – Classe VMBaseCadastro

5 CONCLUSÃO

O desenvolvimento deste projeto foi baseado nas necessidades e interesses de uma escola de dança localizada na cidade de Cascavel, Paraná. Os requisitos do ponto de vista do usuário para o sistema foram fornecidos por uma pessoa que gerencia a escola e que também é professora de dança na referida escola. Embora o sistema tenha sido baseado em uma escola específica, ele se aplica às escolas de danças em geral. Isso porque as rotinas essenciais realizadas nesse tipo de estabelecimento foram consideradas.

Como resultado da realização do trabalho foi implementado um sistema com arquitetura Cliente-Servidor, seguindo o padrão de desenvolvimento *Model-View-View Model*.

O padrão de desenvolvimento escolhido possibilita que futuramente a aplicação possa ser modificada para arquitetura n camadas, conforme exigência de usuários (escolas de dança) de maior porte. A arquitetura projetada permite o aproveitamento do código escrito, visto que, como o próprio padrão adotado sugere, a regra do negócio, a apresentação e a lógica de apresentação estão separadas.

A ferramenta utilizada para mapear e persistir os objetos do banco de dados, NHibernate, torna dispensável a escrita manual de códigos SQL pelo desenvolvedor, entretanto, implementar as classes de mapeamento entre entidades e tabelas pode ser algo trabalhoso, a princípio. Essa prática pode tornar o projeto mais legível e de melhor manutenibilidade.

Como trabalhos futuros complementares ao desenvolvido, está o desenvolvimento da rotina de geração de mensalidade (para um determinado período de tempo), saldos de caixa, listas de presença, agenda do profissional, controle de usuários e autonomias, emissão de recibos, geração de comissão para os profissionais, ficha de avaliação individual de alunos, dentre outros.

REFERÊNCIAS

BARROS, Jussara de. **Dança**. Brasil Escola. Disponível em <http://brasilescola.uol.com.br/artes/danca.htm>. Acesso em: 21 ago. 2016.

MARTINEZ, Marina. Sistema de informação gerencial. **Info Escola**. Disponível em: <http://www.infoescola.com/administracao_/sistema-de-informacao-gerencial/>. Acesso em: 13 set. 2016

OLIVEIRA, Djalma de P. R. de. **Sistemas de informações gerenciais**: estratégias, táticas, operacionais. 7 ed. São Paulo: Atlas, 2000.

OLIVEIRA, Djalma de P. R. de. **Sistemas de informações gerenciais**: estratégicas táticas operacionais. 12 ed. São Paulo: Editora Atlas, 2008.

PEREIRA, Maria José L. de B.; FONSECA, João Gabriel M. **Faces da decisão**: as mudanças de paradigmas e o poder da decisão. São Paulo: Makron Books, 1997.

PRESSMAN, Roger. **Engenharia de software**. Rio de Janeiro: McGraw-Hill, 2008.

REZENDE, Denis A.; ABREU, Aline F. de. **Tecnologia da informação aplicada a sistemas de informação empresariais**: o papel estratégico da informação e dos sistemas de informação nas empresas. 5 ed. São Paulo: Atlas, 2008.

SEED. **História da dança**. Disponível em: <http://www.arte.seed.pr.gov.br/modules/conteudo/conteudo.php?conteudo=102>. Acesso em: 21 ago. 2016.

STAIR, Ralph M.; REYNOLDS, George W. **Princípios de sistemas de informação**. 4 ed. ltc: 2002.

STAIR, Ralph M. **Princípios de sistemas de informação**. 6 ed. Rio de Janeiro: LTC, 2008.

TIPOS DE DANÇA. Disponível em: <http://tipos-de-danca.info/>. Acesso em: 21 ago. 2016.