Documento de Projeto de Sistema

OrcaHIGH Airlines

Vitória, ES

2024

Registro de Alterações:

Versão	Responsável	Data	Alterações
1.0	Isauflânia	05/04/2024	Versão inicial.
1.1	Thayza	07/04/2024	Introdução.
1.2	Marcelo	08/04/2024	Plataforma de desenvolvimento.
1.3	Thayza	14/04/2024	Introdução, Arquitetura de Software.
1.4	Marcelo	18/04/2024	Modelo Navegação.
1.5	Isauflânia	19/04/2024	Arquitetura de Software, Modelo Navegação
1.6	Thayza	19/04/2024	Modelo Entidade, Modelo Aplicação.
1.7	Isauflânia	20/04/2024	Modelo Aplicação, Camada de Acesso.
1.8	Thayza, Isauflânia, Marcelo	21/04/2024	Revisão.

1 Introdução

Este documento apresenta o projeto (*design*) do sistema *OrcaHIGH Airlines*. O OrcaHIGH Airlines é um sistema de vendas de passagens aéreas, que visa proporcionar, de modo simples e eficaz, a compra de viagens de avião, mostrar curiosidades sobre as aeronaves a seus usuários e, também, visa apresentar informações extras sobre as cidades em que os usuários possam estar interessados.

Os dados sobre voos, rotas e curiosidades extras sobre as aeronaves e sobre as cidades são mantidos no sistema pelo usuário do tipo Administrador, que controla o cadastro dessas informações. Já os usuários do tipo Cliente podem realizar a compra de uma viagem, escolhendo os locais de origem e destino, o tipo da passagem, se é econômica, executiva ou primeira classe, escolhe se deseja realizar a compra de uma passagem só de ida ou o combo de passagem com ida e volta, informar a data de partida e, se foi escolhida a opção de compra ida e volta, informar também a data de volta. Após preencher os dados, o usuário realiza uma pesquisa para visualizar os horários e classes disponíveis para a data escolhida e então poder realizar a compra da passagem para a viagem, sendo essa compra escolha da passagem e pagamento.

Os usuários do tipo Cliente, no dia da viagem, tornam-se um usuário do tipo Passageiro, onde, é necessário que o mesmo faça check-in para poder usufruir da viagem comprada. O cliente é passageiro enquanto está no período da viagem.

O usuário, seja ele Cliente, Administrador ou Passageiro, pode realizar uma pesquisa sobre curiosidades das cidades que estão na rota da viagem ou de algum destino de viagem que desejam realizar. Essa pesquisa apresenta ao usuário os pontos turísticos, os eventos que estão sendo realizados e os que irão acontecer naquela localidade e uma breve descrição acerca daquele local.

Além das funcionalidades de compra e venda de viagens e passagens aéreas, o sistema mantém um registro de informações dos usuários. O acesso a esse tipo de usuário é feito através de um cadastro disponibilizado no site.

Este documento está organizado da seguinte forma: a Seção 2 apresenta a plataforma de software utilizada na implementação do sistema; a Seção 3 apresenta a arquitetura de software; por fim, a Seção 4 apresenta os modelos FrameWeb que descrevem os componentes da arquitetura.

2 Plataforma de Desenvolvimento

Na Tabela 1 são listadas as tecnologias utilizadas no desenvolvimento da ferramenta, bem como o propósito de sua utilização.

Tabela 1 – Plataforma de Desenvolvimento e Tecnologias Utilizadas.

Tecnologia	**Versão**	**Descrição**	**Propósito**
Jakarta EE	10	Conjunto de especificação de APIs e tecnologias, que são implementadas por programas servidores de aplicação.	Redução da complexidade do desenvolvimento, implantação e gerenciamento de aplicações Web a partir de seus componentes de infra-estrutura prontos para o uso.
Java	21	Linguagem de programação orientada a objetos e independente de plataforma.	Escrita do código-fonte das classes que compõem o sistema.
JSF	2.3	API para a construção de interfaces de usuários baseada em componentes para aplicações Web	Criação das páginas Web e sua comunicação com as classes Java.
EJB	4.0.9	API para construção de componentes transacionais gerenciados por *container*.	Implementação das regras de negócio em componentes distribuídos, transacionais, seguros e portáveis.
JPA	3.1	API para persistência de dados por meio de mapeamento objeto/-relacional.	Persistência dos objetos de domínio sem necessidade de escrita dos comandos SQL.
CDI	4.0	API para injeção de dependências.	Integração das diferentes camadas da arquitetura.
Facelets	2.0	API para definição de decoradores (*templates*) integrada ao JSF.	Reutilização da estrutura visual comum às paginas, facilitando a manutenção do padrão visual do sistema.
PrimeFaces	12.0	Conjunto de componentes visuais JSF *open source*.	Reutilização de componentes visuais Web de alto nível.
MySQL Server	8.0	Sistema Gerenciador de Banco de Dados Relacional gratuito.	Armazenamento dos dados manipulados pela ferramenta.
WildFly	26	Servidor de Aplicações para Java EE.	Fornecimento de implementação das APIs citadas acima e hospedagem da aplicação Web, dando acesso aos usuários via HTTP.

Na Tabela 2 vemos os softwares que apoiaram o desenvolvimento de documentos e também do código fonte.

Tabela 2 – Softwares de Apoio ao Desenvolvimento do Projeto

Tecnologia	**Versão**	**Descrição**	**Propósito**
Visual Paradigm	17	Ferramenta CASE acrescida de plugin que implementa o método FrameWeb.	Criação dos modelos de Entidades, Aplicação, Persistência e Navegação.
Overleaf	online	Implementadão do LaTeX	Documentação do projeto arquitetural do sistema.
GitHub	online	Plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o Git.	Versionamento e armazenamento de código do projeto.
WhatsApp	Aplicativo Mobile	Aplicativos de comunicação.	Comunicação da equipe.
IntelliJ	2023.3.2	Ambiente de desenvolvimento (IDE) com suporte ao desenvolvimento Java EE.	Implementação, implantação e testes da aplicação Web Java EE.
Apache Maven	3.9.6	Ferramenta de gerência/construção de projetos de software.	Obtenção e integração das dependências do projeto.

3 Arquitetura de Software

A Figura 1 mostra a arquitetura do sistema *OrcaHIGH Airlines* baseada na arquitetura do FrameWeb, sendo esta formada por três camadas e cinco pacotes. As camadas são: *Presentation*, *Business* e *Data Access*. Os pacotes são *View*, *Control*, *Domain*, *Application* e *Persistence*. Além das camadas e pacotes, são utilizadas algumas tecnologias, como por exemplo, a *Primefaces*, *CDI* e *JPA*.

A primeira camada é a *Presentation*, ou de apresentação. Essa camada compreende a interface da aplicação e é formada pelos pacotes *View* e *Control*. O pacote de visão guarda informações, telas e arquivos que são relacionados à apresentação ao usuário, como as páginas web, imagens e *scripts*. Já o pacote de controle monitora os comandos enviados pelo usuário através do *front-end* e engloba também, classes de controle que se integram com o *framework* Controlador Frontal. As páginas presentes no pacote *View* enviam as entradas dos usuários para as classes do controlador, presentes no pacote de controle, onde estas classes processam e retornam as respostas aos usuários, fazendo com que ambos pacotes sejam dependentes um do outro (GUTERRES, 2019) (SOUZA, 2007) (SOUZA V. E. S.; FALBO, 2007).

A camada *Business* ou de negócio é a que reúne as classes do domínio do problema, representadas no pacote *Domain*, e as implementações dos casos de uso, representadas no pacote *Application*. Esta camada é onde a lógica do negócio está implementada (GUTERRES, 2019) (SOUZA, 2007) (SOUZA V. E. S.; FALBO, 2007). Os pacotes *Control* da camada de apresentação e *Application* da camada de negócio possuem uma conexão de dados. Esse intercâmbio é unilateral, garantindo que a camada de negócios mantenha sua independência em relação à camada de interação com o usuário.

Já a terceira e última estrutura está associada ao banco de dados, incumbida de armazenar e persistir os dados da aplicação. Essa camada se conecta à anterior por meio de uma dependência entre o módulo de *Application* e o *Persistence*. Para uma abstração mais elevada, é empregado um *framework* de mapeamento objeto-relacional, seguindo o padrão *Data Access Object*, ou DAO - Objeto de Acesso a Dados (SOUZA V. E. S.; FALBO, 2007) (SOUZA, 2020).

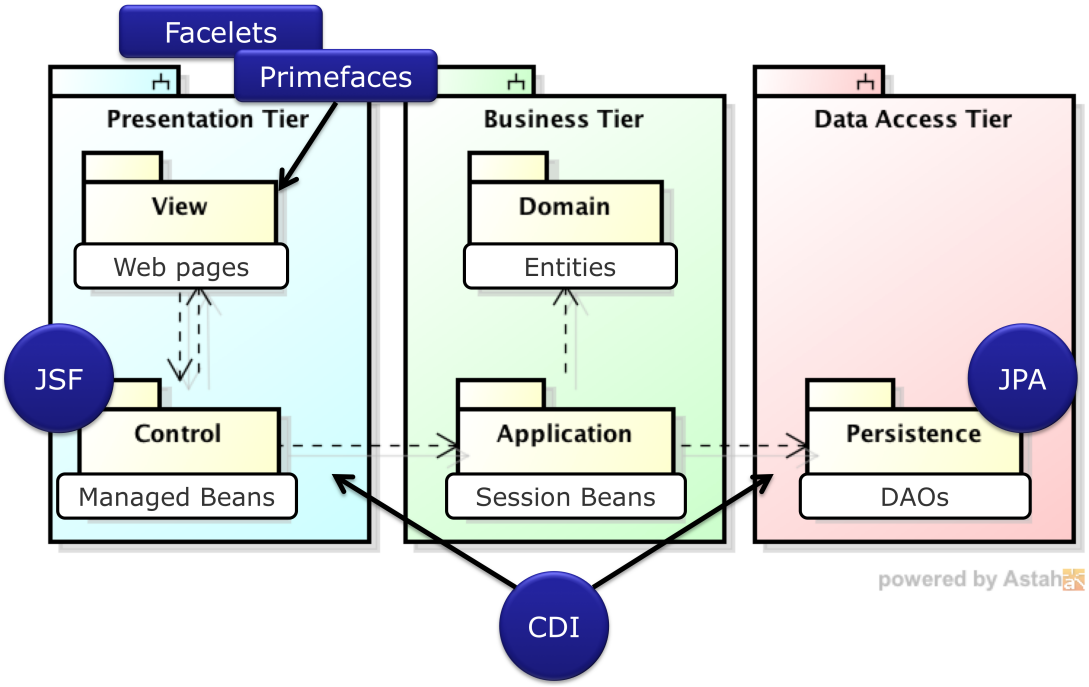

Figura 1 – Arquitetura de Software.

4 Modelagem FrameWeb

OrcaHIGH Airlines é um sistema Web cuja arquitetura utiliza *frameworks* comuns no desenvolvimento para esta plataforma. Desta forma, o sistema pode ser modelado utilizando a abordagem FrameWeb (SOUZA, 2020).

A Tabela 3 indica os *frameworks* presentes na arquitetura do sistema que se encaixam em cada uma das categorias de *frameworks* que FrameWeb dá suporte. Em seguida, os modelos FrameWeb são apresentados para cada camada da arquitetura.

Tabela 3 – *Frameworks* da arquitetura do sistema separados por categoria.

Categoria de *Framework*	***Framework* Utilizado**
Controlador Frontal	JSF
Injeção de Dependências	CDI
Mapeamento Objeto/Relacional	JPA
Segurança	JAAS

4.1 Camada de Negócio

A camada central da arquitetura do sistema, camada de negócio, apresenta dois pacotes, sendo eles: *Application* e *Domain*. Esses pacotes são apresentados pelos modelos de Entidade e Aplicação.

4.1.1 Modelo de Entidade

O modelo de entidade apresenta os objetos do domínio e suas relações para a persistência (GUTERRES, 2019) (SOUZA, 2007). Para o sistema *OrcaHIGH Airlines*, as classes Usuário, sendo ela Admin, Cliente ou Passageiro, Passagem, podendo ser somente de Ida ou de Ida e Volta, tendo como tipos Econômica, Executiva ou Primeira Classe, todas elas podendo ou não ter Bagagem, Pagamento, Compra, Aeroporto, que possui um Endereço pertencente a uma Cidade, essa Cidade pode ter Curiosidades, Voo, Aeronave e Rota integram o modelo do sistema e podem ser vistas na Figura 2.

Um Usuário do tipo Admin realiza os cadastros básicos do sistema, como por exemplo, o cadastro dos dados de Aeronaves, bem como suas curiosidades, as Rotas de Voo da Aeronave e os Aeroportos em que a Aeronave está com rota programada de pouso ou decolagem naquela localidade. Esse tipo de usuário também realiza o cadastro das Curiosidades sobre uma referente Cidade, assim o usuário do tipo Cliente, em algum momento de navegação no sistema, poderá realizar uma pesquisa a respeito.

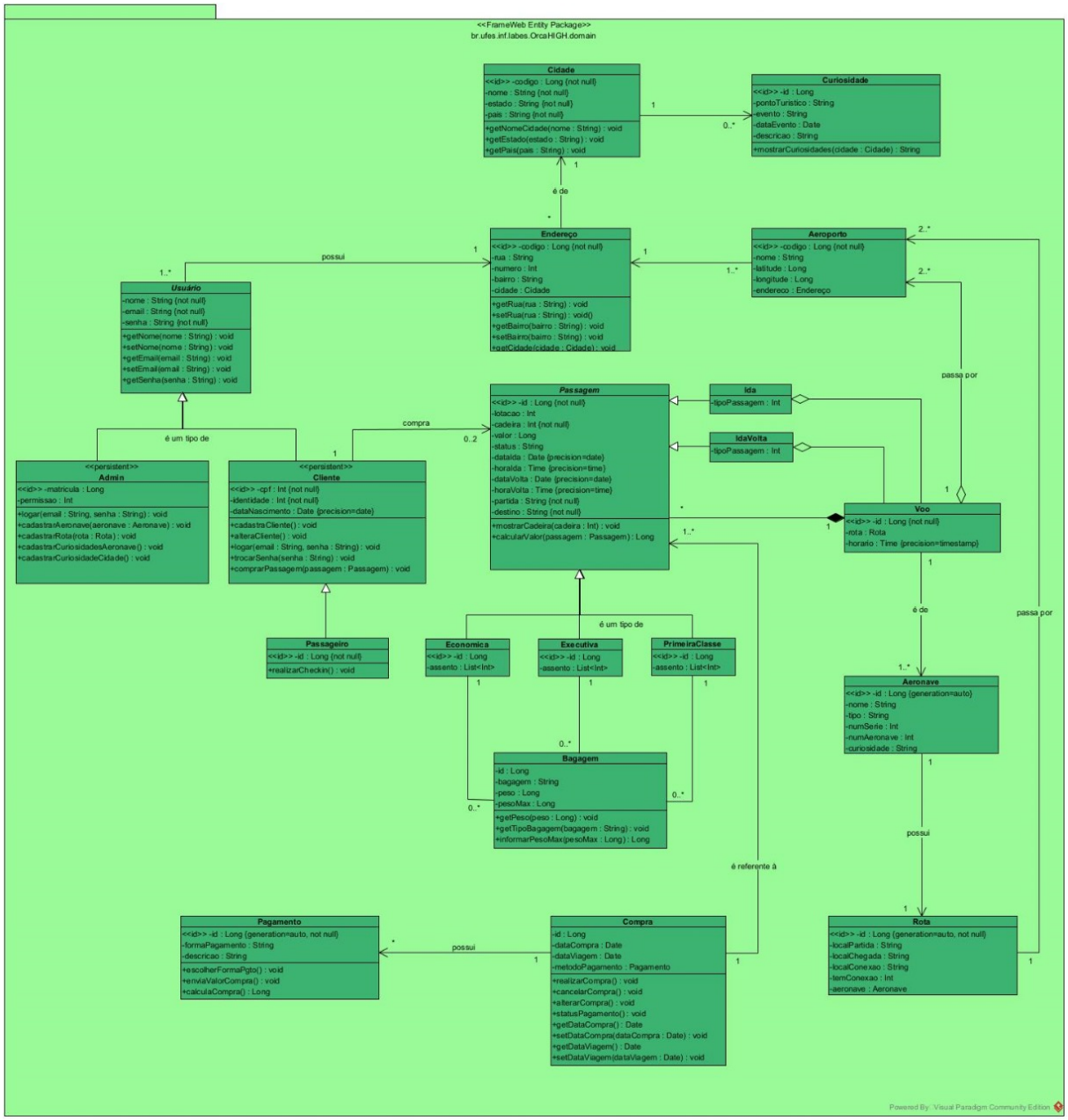

Figura 2 – Modelo de Entidades.

Um usuário do tipo Cliente pode realizar uma compra de Passagem. Para tal evento, ele deve escolher os locais de origem e destino e se é de ida e volta. Em seguida, o Cliente deve informar a data de ida ou as datas de ida e volta, caso escolhida essa opção. O sistema então informa ao Cliente as classes e os horários que estão disponíveis para a data e localidade desejadas. Assim que o Usuário escolhe a classe, dentre os tipos Econômica, Executiva ou Primeira Classe, as poltronas que estão disponíveis são apresentadas para escolha. O Cliente deve selecionar a poltrona para compra da Passagem e seguir para o procedimento de Pagamento. Para tais ações, o Cliente deve cadastrar-se no sistema primeiro.

4.1.2 Modelo de Aplicação

O modelo de aplicação é um diagrama de classes da UML que representa as classes de serviço (SOUZA, 2007). Essas classes são responsáveis pela implementação das funcionalidades das camadas lógicas de negócios do sistema e suas dependências com outras camadas. Esse diagrama é aplicado para conduzir a elaboração das classes do pacote de aplicação. Ele é responsável por apresentar como as classes da camada de aplicação se conectam com as classes de controle e de persistência.

No modelo apresentado na Figura 3, a relação de dependência é demonstrada por *Registration*, *BuscarCidade*, *BuscarVoos* onde as classes de controle (ação), dependem das classes de aplicação (serviço) e estas dependem das classes de persistência, sendo essas (DAOs necessários para auxiliarem as camadas de serviço em seus objetivos).

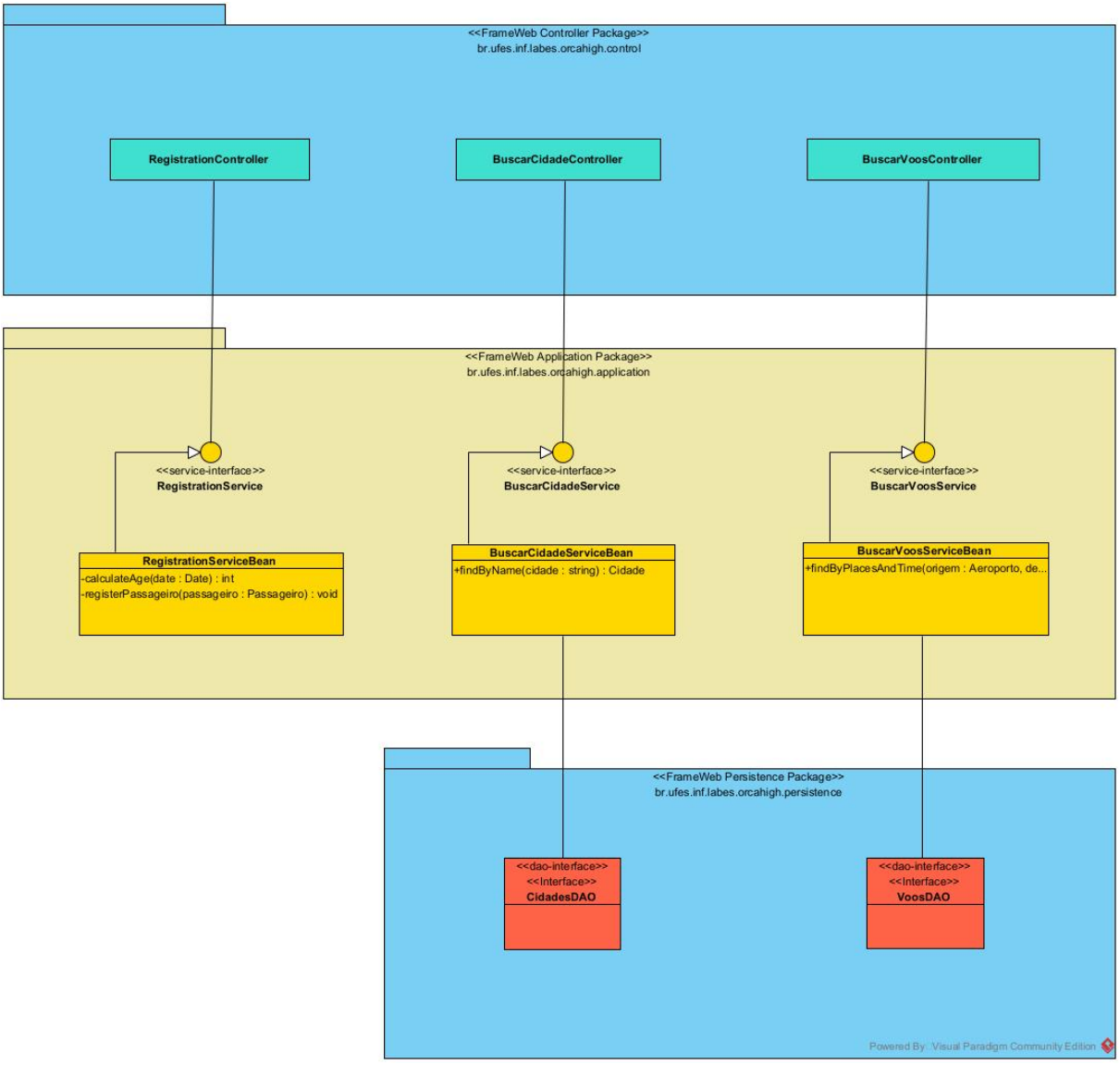

Figura 3 – Modelo de Aplicação do *OrcaHIGH Airlines*.

O modelo de aplicação apresentado cobre os casos de uso apresentados nos modelos

de navegação. As ligações e dependências entre **BuscarCidade**, **BuscarCidadeService**, **BuscarCidadeServiceBean** e **CidadesDAO** são evidentes devido às operações de manipulação da entidade *Buscar Cidade*; assim como as dependências entre **BuscarVoosController**, **BuscarVoosService**, **BuscarVoosServiceBean** e **VoosDAO** em relação a *Buscar Voos*. Por outro lado, em relação as dependências entre **RegistrationController**, **RegistrationService**, **RegistrationServiceBean**, vale ressaltar que não se trata de uma entidade de domínio e, portanto, não há nenhum dado armazenado no banco de dados; os registros são utilizados para orientar o usuário a obter o seu registro (essas, sim, são entidades armazenadas no banco de dados).

4.2 Camada de Acesso a Dados

Vamos abordar mais um aspecto do *OrcaHIGH Airlines*: a estrutura de acesso aos dados, situada na camada lógica. Aqui, temos o pacote de persistência desempenhando um papel central. Esse arranjo de persistência engloba os elementos responsáveis por armazenar os dados gerados pelo sistema. O padrão DAO, visa a simplificação das interações com o banco de dados através do ORM, temos a representação das classes DAO, suas interfaces e suas implementações contidas nesse conjunto (SOUZA, 2007). Contudo, o *OrcaHIGH Airlines* não usa o padrão DAO ou similar, assim, realizamos a implementação de um Modelo de Persistência simplificado.

Quanto às funcionalidades detalhadas neste documento, o modelo de persistência, como mostrado na 4, apresenta os DAOs pertinentes às operações envolvendo entidades como *Cidades* e *Voos*. Portanto, observamos que duas novas funcionalidades já estão planejadas para este ciclo de desenvolvimento do *OrcaHIGH Airlines*:

- A funcionalidade de *busca de voos* engloba a pesquisa de destinos/origens disponíveis em uma data específica. Para isso, foi essencial recuperar da lista de objetos *aeroportos* com base em um período definido pelo usuário, composto pelas datas inicial e final fornecidas;
- A funcionalidade de *busca de cidades* requer a recuperação de uma *cidade* com base no valor do atributo de string *cidade*.

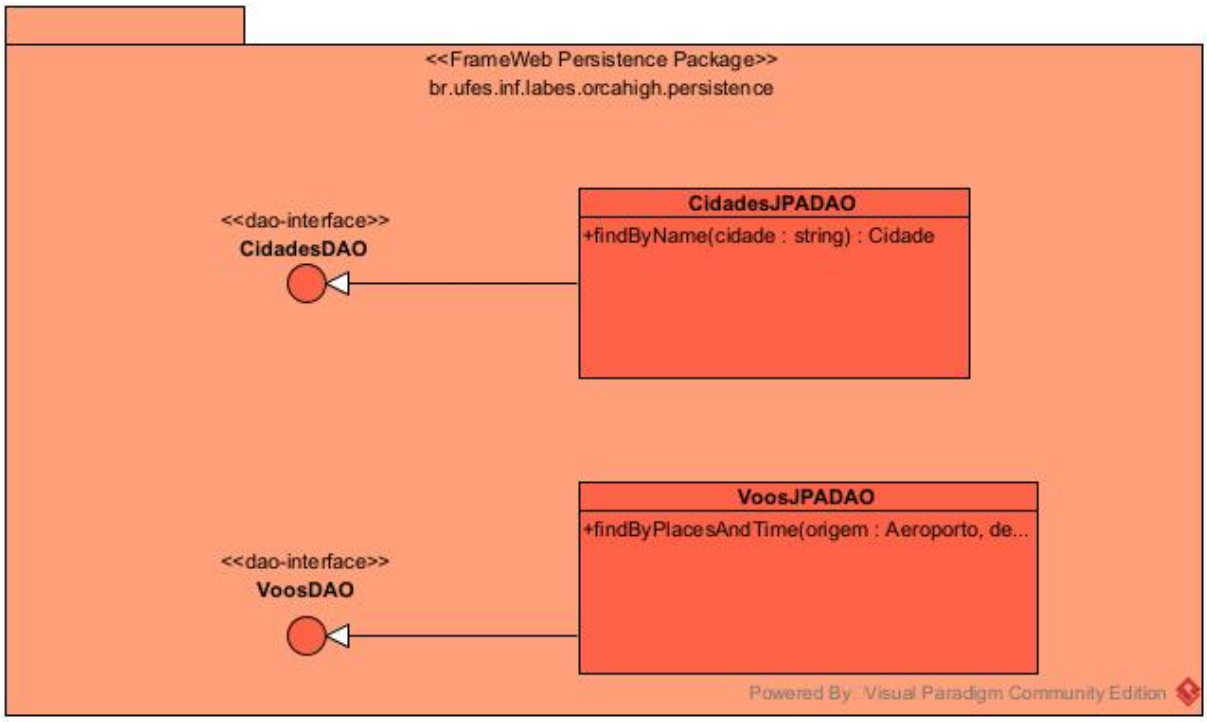

Figura 4 – Modelo de Persistência do *OrcaHIGH Airlines*.

4.3 Camada de Apresentação

A camada de apresentação se responsabiliza pela interface com o usuário. É através desta camada que o sistema disponibiliza suas funcionalidades e se conecta com as regras de negócio da camada subsequente (SOUZA, 2007). Assim, além de possuir classes do pacote de visão, as classes controladoras complementam a camada e disponibilizam acesso ao pacote de aplicação.

Neste contexto, objetivando propiciar uma modelagem aderente aos requisitos do sistema *OrcaHIGH Airlines*, foram exemplificados três submodelos[1] de navegação, detalhando os pacotes *View* e *Controller*, listados abaixo:

1. Modelo de Registro de Passageiro.
2. Modelo de Busca de Curiosidades das Cidades.
3. Modelo de Busca de Voo.

4.3.1 Modelo de Navegação de Registro de Cliente

A rotina aqui apresentada é conduzida pelo usuário durante suas ações de registro no sistema. Somente após este registro o usuário poderá realizar suas aquisições no sistema.

[1] Os modelos em tela foram produzidos através do Visual Paradigm.

Ainda que para finalização da compra de passagens o cadastro seja obrigatório, existem funcionalidades do sistema que não demandam autenticação preliminar, como por exemplo, consultar curiosidades a respeito de uma determinada cidade.

Na Figura 5 apresentamos os passos do modelo com detalhamento de campos e fluxo de informações.

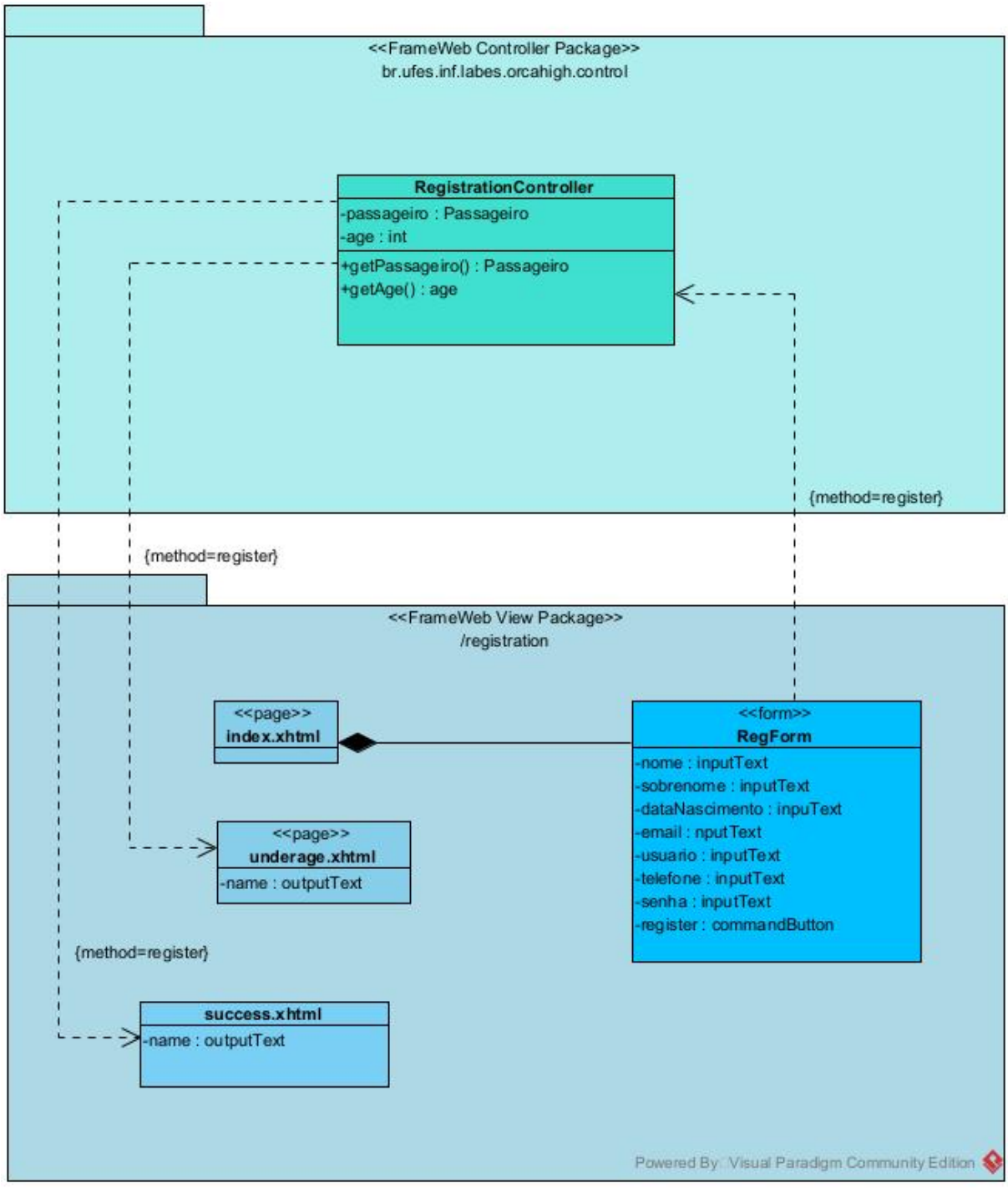

Figura 5 – Modelo de Navegação - Registro de Cliente.

Na página inicial, o sistema disponibiliza ao usuário um formulário. Neste formulário,

o usuário inclui dados como nome, sobrenome, data de nascimento, e-mail e senha. Através do método **RegForm**, o sistema aciona o controlador **RegistrationController** que, por sua vez, interage com a camada de aplicação e eventualmente valida o usuário. Em caso de sucesso, o sistema retorna uma determinada página, a página **success.xhtml**. Caso contrário, o sistema devolve com uma página de exceção, **underage.xhtml**.

4.3.2 Modelo de Navegação de Busca de Cidades

Além da funcionalidade de cadastro de usuário cliente, o sistema disponibiliza ferramentas de busca de cidades e suas curiosidades, visando ofertar aos usuários possíveis destinos. Na medida em que se interesse por um local, o usuário pode navegar para uma tela de busca de voos. Ao final da escolha, caso não possua cadastro ou não esteja logado, o usuário será destinado ao ambiente de registro de clientes. Esta funcionalidade não foi ilustrada no modelo apresentado na Figura 6, em função da sua ocorrência condicional.

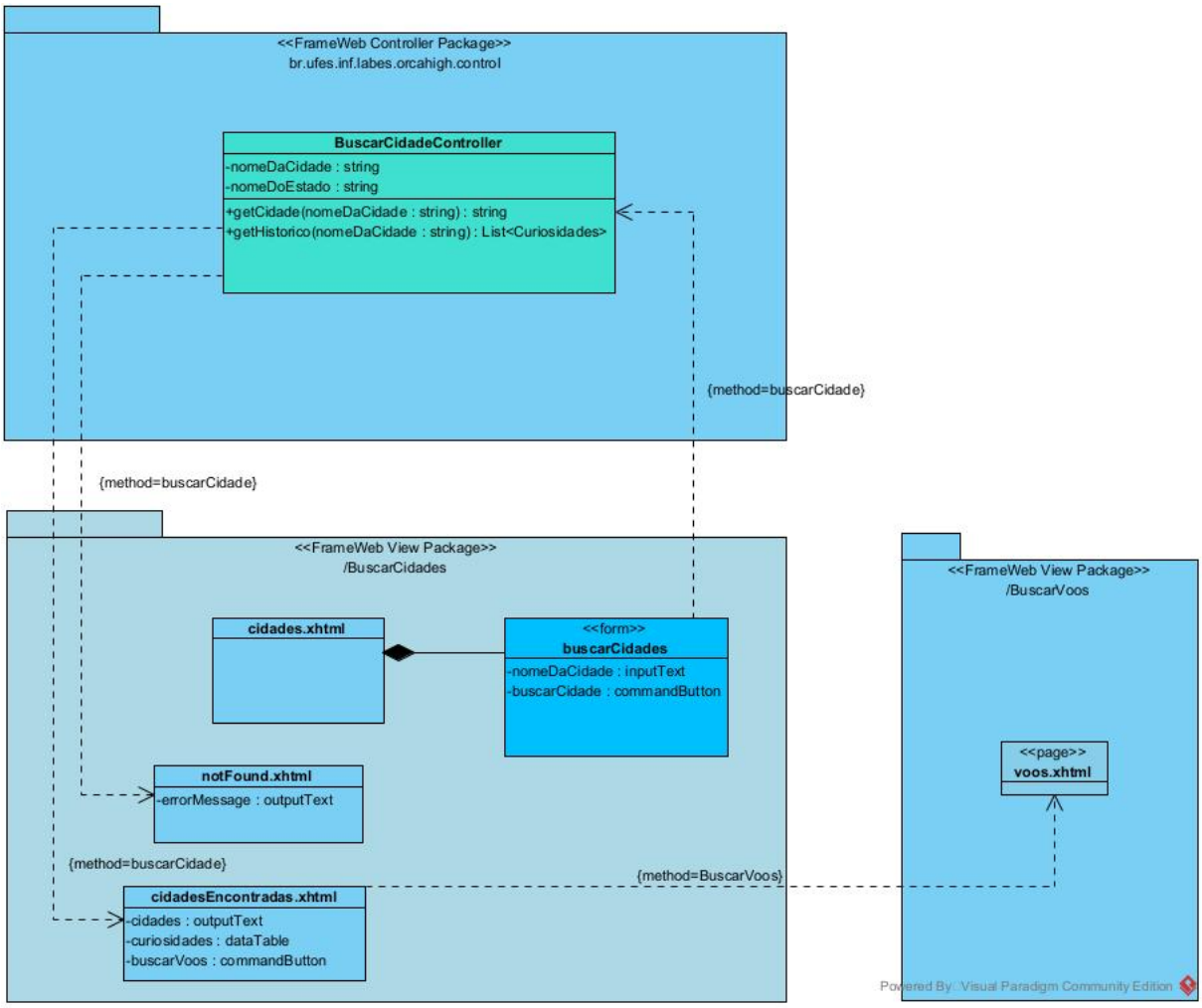

Figura 6 – Modelo de Navegação - Buscar Cidades.

4.3.3 Modelo de Navegação de Busca de Voos

O diagrama apresentado pela Figura 7 detalha a busca por voos. O modelo possui um página de voos que, por sua vez, dispõe um formulário **voosForm**. Através deste formulário, o usuário deve informar origem, destino, dias e horários dos voos, entre outras informações. Ao acionar o controlador **BuscarVoosController**, o sistema verifica a disponibilidade de rotas e horários e retorna para o usuário uma tabela com os voos disponíveis.

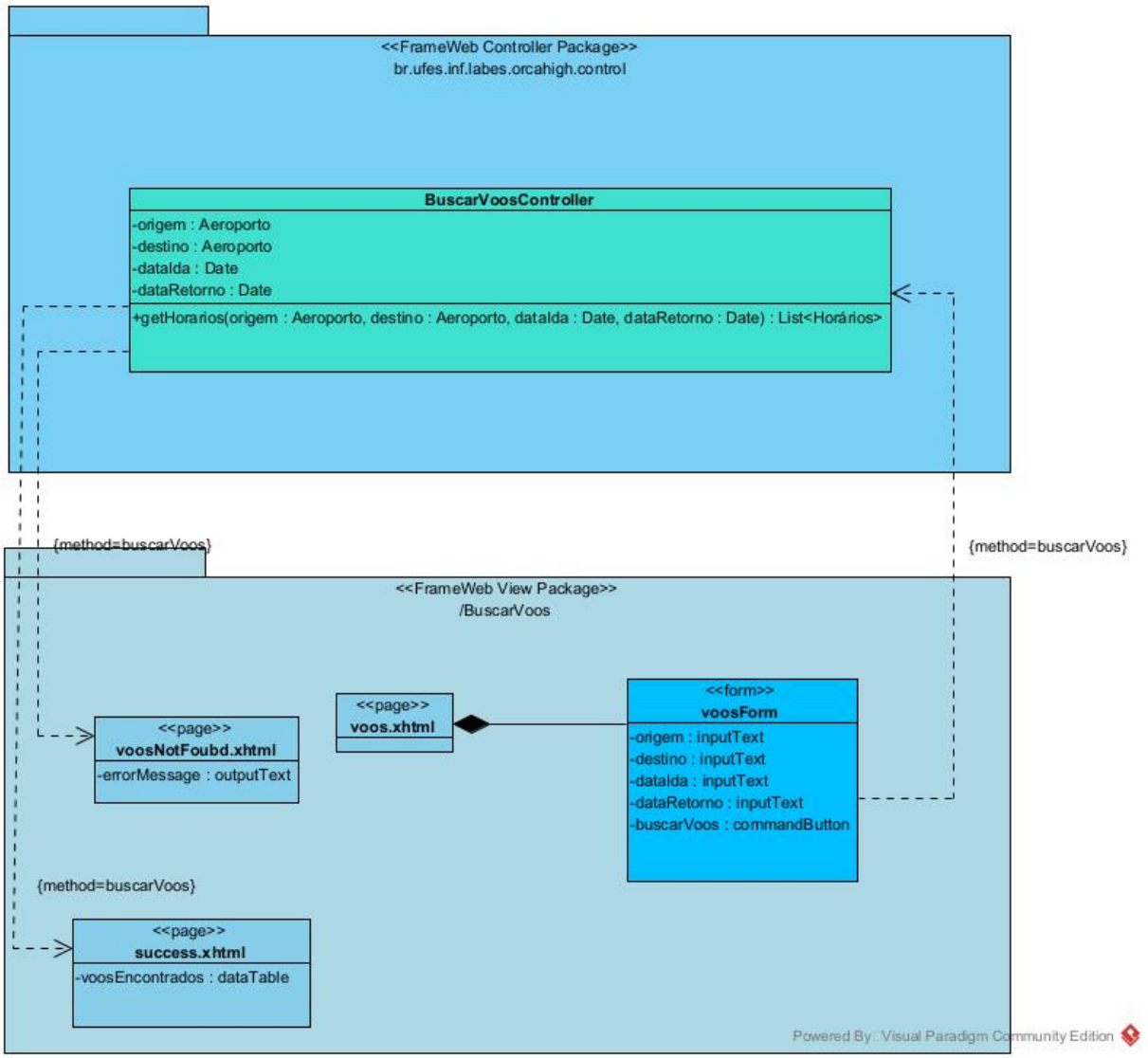

Figura 7 – Modelo de Navegação - Buscar Voos.

Para a eventual aquisição das passagens, o cliente deverá estar registrado no sistema. O processo de pagamento das passagens ocorrerá através de serviço fornecido por instituições financeiras à parte.

Referências

GUTERRES, C. S. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando o framework Play*. Vitória, ES, Brazil: [s.n.], 2019. Disponível em: <https://nemo.inf.ufes.br/wp-content/papercite-data/pdf/aplicacao_do_metodo_frameweb_no_desenvolvimento_de_um_sistema_de_informacao_usando_o_framework_play_2019.pdf>. Citado 3 vezes nas páginas 4, 5 e 6.

SOUZA, V. E. S. *FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web*. Tese (Doutorado), 2007. Citado 6 vezes nas páginas 4, 5, 6, 8, 9 e 10.

SOUZA, V. E. S. The FrameWeb Approach to Web Engineering: Past, Present and Future. In: ALMEIDA, J. P. A.; GUIZZARDI, G. (Ed.). *Engineering Ontologies and Ontologies for Engineering*. 1. ed. Vitória, ES, Brazil: NEMO, 2020. cap. 8, p. 100–124. ISBN 9781393963035. Disponível em: <http://purl.org/nemo/celebratingfalbo>. Citado na página 5.

SOUZA V. E. S.; FALBO, R. A. *Frameweb: A framework-based design method for web engineering*. EATIS, NEMO, 2007. Disponível em: <https://nemo.inf.ufes.br/wp-content/papercite-data/pdf/frameweb___a_framework_based_design_method_for_web_engineering_2007.pdf>. Citado 2 vezes nas páginas 4 e 5.