**Documento de Arquitetura de Software**



**DrinkIt**

**Histórico de Versões**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** | **Revisor** |
| 05/03/11 | 1.0 | Modelagem e desenvolvimento | Gabriel Lima Gomes e Gustavo da Rosa | - |

|  |  |
| --- | --- |
| **Cliente** | FATEC - Interno |
| **Documento** | Documento de Arquitetura de Software: *DrinkIt* |
| **Data** | 07 de outubro de 2019 |
| **Autor** | **Gabriel Lima Gomes**  [gabriellimagomes.14@hotmail.com](mailto:gabriellimagomes.14@hotmail.com)  **Gustavo da Rosa**  [guga-rosa13@hotmail.com](mailto:guga-rosa13@hotmail.com) |

**Página de Assinaturas**

|  |  |  |
| --- | --- | --- |
| Revisado e Aprovado por: |  |  |
|  |  | 27/11/10 |

**Índice**

Camada de Apresentação 4

Camada de Persistência 11

Diagrama de caso de uso geral do sistema 6

Escopo 4

Objetivo 4

Pacote Model 11

Qualidade 17

Realização dos Casos de Uso Significativos 6

Representação Arquitetural 4

Tamanho e Performance 17

Visão de Dados 14

Visão de Implementação 14

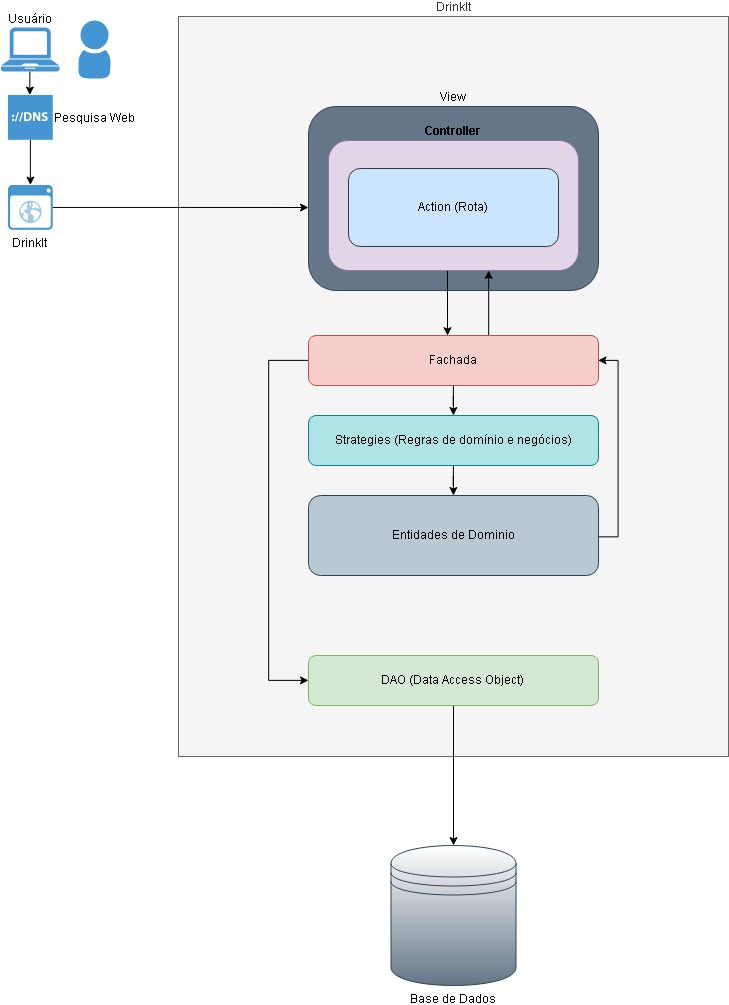
Objetivo

Este documento tem por objetivo apresentar uma visão arquitetural dos sistemas da linha **DrinkIt**. O intuito é salientar diferentes aspectos deste produto, obtidos a partir de decisões arquiteturais realizadas no âmbito dos sistemas da linha **DrinkIt**.

Escopo

O escopo deste documento é documentar as partes significativas do ponto de vista da arquitetura do modelo de design, como sua divisão em subsistemas e pacotes. Além disso, mostra sua divisão em classes e utilitários de classe.

Representação Arquitetural

Os sistemas serão desenvolvidos tendo como base a arquitetura ilustrada na Figura 1. Toda a arquitetura será baseada nos padrões de projetos tradicionais do GoF.

**Figura 1 - Modelo Arquitetural Genérico**

As Classes de Domínio são as classes que representam a abstração do negócio no estilo *anêmico* (somente com os atributos).

As Classes de Negócio representam as classes responsáveis por aplicar as regras de negócio do sistema como, por exemplo, Cadastrar Bebida. Constarão também nas classes de negócio o relacionamento com os DAOs responsáveis por persistir e recuperar os objetos no banco de dados.

Como mostra a figura, nem todas as camadas conversam entre si, gerando menor acoplamento, e aumentando a manutenção e escalabilidade do sistema, obedecendo a arquitetura MVC.

* 1. Restrições Arquiteturais

Foram identificadas algumas orientações / restrições pertinentes ao desenvolvimento deste subsistema:

* Utilização do framework .NET 4.5, ou versões superiores;
* Utilização do servidor IIS;
* Utilização do framework ASP.NET MVC e da tecnologia Ajax para desenvolvimento da camada de apresentação;
* Utilização do framework de mapeamento objeto-relacional ADO.NET;
* Utilização do MSSQL Server.
  1. Objetivos e Restrições Arquiteturais

Alguns requisitos registrados que impactam diretamente a arquitetura do sistema *DrinkIt*, são:

* Utilização da Linguagem C#
* Considerar a utilização de software Livre, quando possível
* O Sistema de Gerenciamento de Banco de Dados a ser considerado em implementações de âmbito corporativo será o SGBD MSSQL Server.

1. Visão de Use Case

Esta seção apresenta os Casos de Uso arquiteturalmente significativos, que foram selecionados considerando-se o pacote do Modelo de Casos de Uso que representa o sistema DrinkIt.

A classificação dos casos de uso, em termos de significância, foi realizada com base na observação de pelo menos um dos seguintes critérios:

* Casos de uso que estendem outros Casos de Uso
* Casos de Uso que são incluídos em outros Casos de Uso e
* Casos de uso que acessam sistemas externos
  1. Diagrama de Caso de Manter Dados de Atributos

Uma imagem contendo texto, mapa

Descrição gerada automaticamente

Figura 4.1 Diagrama de Caso de Uso de Manter dados de Atributos

* 1. Descrição dos Casos de Uso Arquiteturalmente Significativos

**Gerenciar conta**

Este caso de uso se inicia quando o usuário cria sua conta no sistema, ou então, opcionalmente, decide editar os dados da mesma.

**Gerenciar bebidas**

Este caso de uso se inicia quando o administrador cadastra uma nova bebida no sistema, ou então altera dados sobre uma bebida existente, editando eles ou trocando seu status (ativo/inativo).

**Gerenciar estoque**

Este caso de uso se inicia quando o administrador dá entrada ou baixa no estoque de uma bebida específica, sendo necessário, no caso das baixas, informar o motivo de tal decisão.

**Gerenciar cartões**

Este caso de uso se inicia opcionalmente quando o cliente cadastra mais cartões de crédito em sua conta, para utilizar em suas futuras compras.

**Gerenciar Endereços**

Este caso de uso se inicia opcionalmente quando o cliente cadastra mais endereços em sua conta, de entrega e/ou de cobrança, para utilizar em suas futuras compras.

**Gerenciar pedidos**

Este caso de uso se inicia quando o cliente realiza seu pedido, escolhendo as bebidas desejadas, ou então, opcionalmente, verifica os detalhes de um pedido já feito, tendo a opção de solicitar um cupom de troca a partir do mesmo.

1. Visão de Lógica

Esta visão apresenta elementos de design significativos do ponto de vista da arquitetura, descrevendo a organização do Sistema DrinkIt em pacotes, bem como a organização desses pacotes em camadas.

O Diagrama com as camadas do sistema DrinkIt é ilustrado na figura 5.1.



Figura 5.1 – Diagrama de camadas do DrinkIt

**Apresentação**: Contém classes para as interfaces gráficas com os usuários (GUI). Através destas interfaces os usuários conseguem interagir com o DrinkIt, com o intuito de incluir, alterar e excluir produtos.

**Negócio**: Contém classes que controlam a execução das funcionalidades e detém as informações e regras do negócio relacionado ao DrinkIt.

**Persistência**: Contém classes responsáveis por persistir as entidades de modelo. Por exemplo, contém as classes que permitem ler e gravar os objetos no banco de dados relacional.

Neste momento, é importante ressaltar que a camada de apresentação envolve componentes que são executados na porção servidora e na porção cliente. Na porção servidora, são executados os componentes que montam as páginas html/jsp de resposta e controlam o fluxo de interação com o usuário. Assim, envolve interpretação de páginas jsp, utilização de helpers para montagem das páginas de resposta, delegação de solicitação para a camada de negócio, obtenção de respostas da camada de negócio, dentre outros. Já na porção cliente, estão os códigos javascript que são executados nos navegadores dos usuários com o intuito de facilitar a interação. Por exemplo, podem fazer uma pré-verificação dos dados digitados (como o preenchimento de campos obrigatórios), solicitação de páginas via ajax, de modo a tornar mais dinâmica a interação com o usuário, dentre outros.

* 1. Camada de Apresentação

Nesta camada, temos o pacote form que contém todos os arquivos relacionados à exibição de informações para usuário, o que engloba páginas HTML, imagens, javascript, dentre outros. Já o pacote de controle desta camada, contém as Actions responsáveis pela comunicação com as classes da camada de negócio. Estes pacotes podem ser vistos na Figura 5.3.



Figura 5.3: Camada de Apresentação

## Camada de Negócio

Nesta camada, temos o pacote controle que contém as classes responsáveis por controlar as regras de negócio da aplicação. O pacote model, contém as classes que representam o modelo, ou seja, aquelas que contém as informações sobre o Sistema DrinkIt. O pacote controller contém as classes de controle do negócio. O pacote complementador contém classes que auxiliam na complementação de informações e o pacote filtro contém as classes que auxiliam na filtragem de informações.

A imagem 5.4 ilustra os pacotes descritos.



Figura 5.4: Camada de Negócios

* + 1. Pacote Controller

A figura 5.5 ilustra as principais classes de controle.

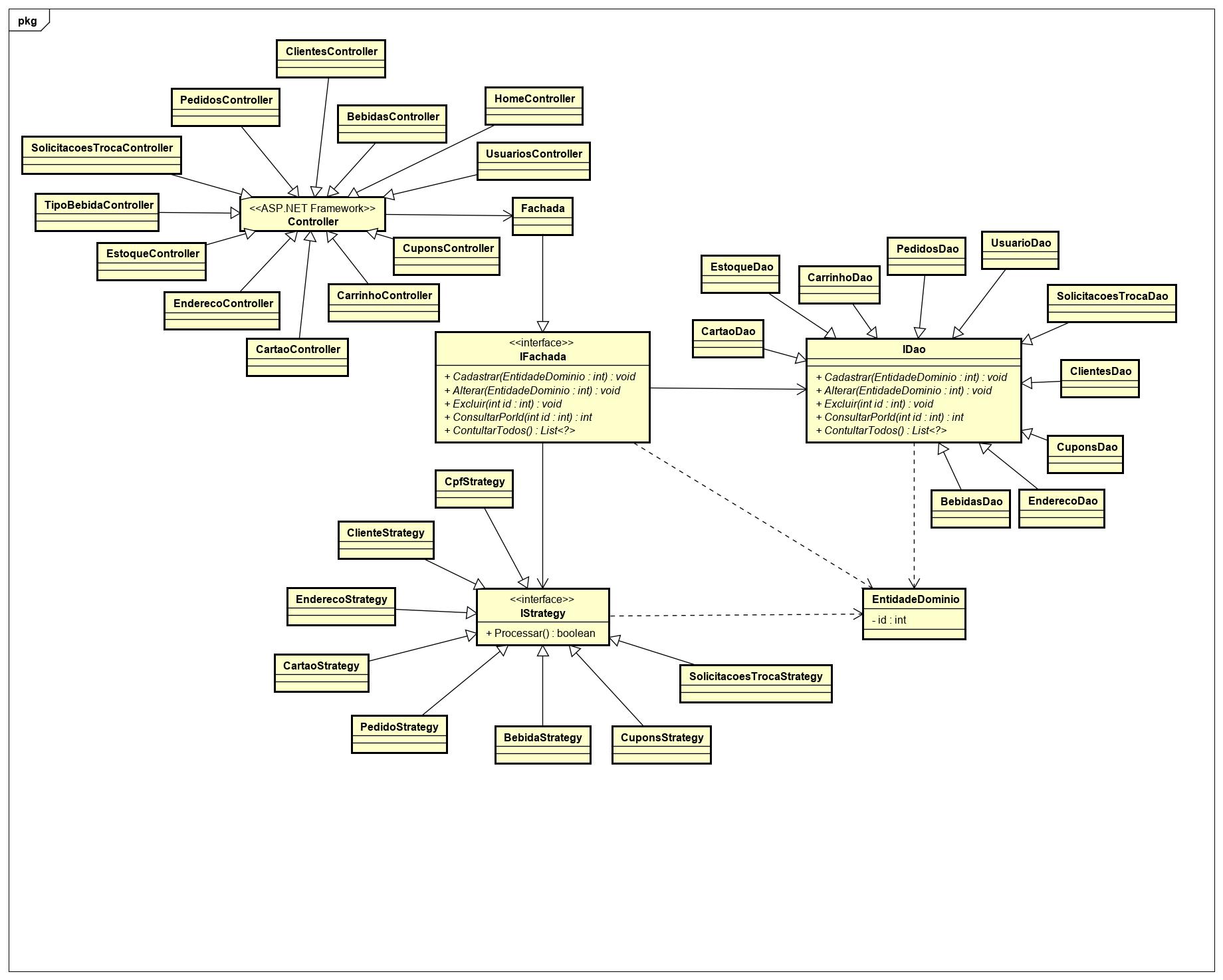


Figura 5.5: Classes de controle

* + 1. Pacote Model

A figura 5.6. ilustra as principais classes do modelo.

Uma imagem contendo texto, mapa

Descrição gerada automaticamente

Figura 5.6: Classes de Modelo

## Camada de Persistência

Nesta camada temos o pacote dao que contém as classes e interfaces responsáveis por persistir as informações do DrinkIt no BD relacional. O pacote ADO.NET contido em dao, possui as classes que dependem diretamente do ADO.NET, que é o framework utilizado para realizar o mapeamento objeto relacional.



Figura 5.7: Camada de Persistência.

## Realização dos Casos de Uso Significativos

Uma imagem contendo captura de tela

Descrição gerada automaticamente

1. Visão de Implantação

Esta seção descreve as configurações da rede física (hardware) na qual o DrinkIt será implantado e executado.

Trata-se de uma visão do Modelo de Implantação que, para a configuração em questão, indica os nós físicos (computadores, CPUs), que executarão o subsistema DrinkIt, e as respectivas interconexões (barramento, LAN, etc). A figura 6 ilustra o modelo de implantação para o DrinkIt.



Figura 6: Visão de Implantação do DrinkIt

Na Figura 6 observa-se os seguintes nós físicos:

* **Web Client Application (WCA):** Aplicativos com interface de usuário via navegador, construídos com base no Framework Bootstrap.
* **Servidor DB**: Nó que contém o BD Central do Sistema DrinkIt.
* **Servidor de Controle de Estoque**: Nó contendo o back-end do sistema DrinkIt.

1. Visão de Implementação

Esta visão descreve a estrutura geral de implementação, a decomposição do software em camadas de implementação.

A estrutura geral de implementação para o DrinkIt é baseada na estrutura da Visão Lógica, assim, não há necessidade de detalhar os diagramas de camadas e pacotes de implementação, uma vez que são fortemente baseados naqueles desenvolvidos para Visão Lógica.

1. Visão de Dados

O mecanismo de persistência utilizado no sistema DrinkIt utiliza-se o banco de dados Relacional MSSQL Server, juntamente com o framework de persistência de dados, ADO.NET. O controle de transações adotado envolve a utilização do Framework ASP.NET MVC em conjunto com o ADO.NET.

As figuras 8.1 e 8.2, apresentam a visão lógica e física da base de dados do DrinkIt.

Uma imagem contendo texto

Descrição gerada automaticamente

Figura 8.1 – Modelo Lógico

Uma imagem contendo texto, mapa

Descrição gerada automaticamente

Figura 8.2 – Modelo Físico

A Tabela 2 define o mapeamento das principais classes de modelo para entidades do modelo lógico do BD MSSQL Server.

Note que existem alguns campos nas entidades lógicas do BD que não estão mapeadas diretamente com as classes de modelo da Visão Lógica contidas neste documento.

Tabela 2: Mapeamento Objeto-Relacional

|  |  |
| --- | --- |
| **Classe** | **Entidade** |
|  |  |
|  |  |

1. Tamanho e Performance

O sistema DrinkIt será usado para vendas online e consequentemente terá uma grande base.

Seus servidores provavelmente irão passar por períodos de picos de utilização (por exemplo, próximo a datas comemorativas, ou períodos de festividades).

As estimativas do número de usuários e de carga de utilização em períodos de pico de utilização, bem como maiores informações sobre questões relacionadas ao tamanho e desempenho do sistema DrinkIt, podem ser obtidas no documento de requisitos não funcionais.

1. Qualidade

O sistema DrinkIt será usado para vendas online e, consequentemente tratando de altos volumes financeiros e grande número de operações de vendas diariamente.

Eventuais erros e/ou falhas na sua operação podem levar a prejuízos significativos tanto em termos financeiros, portanto na fase de design deve-se levar em consideração como fatores prioritários a confiabilidade e robustez do sistema.

Adicionalmente, o sistema DrinkIt pode ser alvo de ataques de “hackers” para roubar ou simplesmente corromper informações, possibilidade aumentada pela interface do sistema disponível na Internet, para evitar que tais ataques tenham sucesso, uma infraestrutura de segurança deve ser especificada e projetada.

Maiores informações sobre questões relacionadas aos requisitos de qualidade do sistema DrinkIt podem ser obtidas no documento de requisitos não funcionais.

1. Referências

Unified Modeling Language: <http://www.omg.org/technology/documents/formal/uml.htm>

RUP. Rational Unified Process.

ASP.NET MVC - <https://docs.microsoft.com/pt-br/aspnet/>

GOF -

<https://www.devmedia.com.br › design-patterns-padroes-gof>

MVC -

[https://tableless.com.br › mvc-afinal-e-o-que](https://tableless.com.br › mvc-afinal-e-o-que )