Projeto GAMA

Cadastro de Usuários

Desafio a Vaga de Tech Lead

Edital 2021/25

André Luiz Braga, D.Sc

# Definição

A requisição do cliente consiste na criação de um sistema de cadastro de usuários e seus respectivos perfis e cargos. Os cargos e os perfis devem poder ser gerenciados com as funcionalidades de edição e listagem. Os usuários devem poder ser cadastrados ou editados, armazenando também a data de registro do cadastro. Os usuários devem ter obrigatoriamente Nome, CPF e cargo e opcionalmente, Data de Nascimento e Sexo.

# Escopo

Este projeto irá assumir as seguintes premissas, por razão de tempo para o desenvolvimento do protótipo:

* A interface com o usuário será simples, assim como o banco de dados, desta forma, utilizando apenas tipos de dados básicos de texto e numéricos. Entradas de dados do tipo data, por exemplo, poderão usar componentes sofisticados de entrada, como pequenos calendários em uma versão posterior
* Da mesma forma, CPFs não serão checados por validade oficial
* Utilizaremos o banco de dados JavaDB incorporado ao JDK e a ferramenta NetBeans, contudo, a utilização de um padrão de Objetos de Acesso de Dados (DAO) consistente provê o necessário isolamento e facilidade de incorporação de outros bancos, como o MySQL, em próximas versões.
* A interface com o usuáro, ou o “Front-End” (View do Modelo MVC) será feita em JAVA Swing em uma aplicação DESKTOP, uma vez que a construção de uma aplicação web requer mais tempo e requisitos de configuração e infraestrutura do cliente, o qual nesse ponto são inclusive desconhecidas.
* As funções de “Alteração” de Cargo e Perfil não estarão ativas, dado que tais entidades não possuem atributos a serem alterados, contudo, as interfaces com o usuário e o esqueleto principal estarão disponíveis para tai extensão rápida se necessário.
* Apesar de não requerido, o Usuário, Cargo e Perfil terão funções de “Listar” disponíveis para facilitar o processo de edição
* Funções de busca não foram requeridas, portanto não serão expostas, a não ser que necessárias internamente.
* Visando possibilitar a incorporação de diversos front-ends em versões futuras, o sistema terá um componente de interface entre o front end, ou seja, as interfaces com o usuário do modo Desktop em Swing, e o backend construído puramente em JAVA por requisição do cliente. Em futuras versões, como por exemplo, uma versão WEB JEE, esta camada poderá ser substituída por servlets ou frameworks como Struts, JSF, etc
* Decidimos utilizar a persistência através de puro JDBC e suas APIs JTA e JPA, dado que, para um banco de dados tão pequeno e muito poucas chances de modificação, os processos de configuração de frameworks de ORM poderia tomar mais tempo do que a implementação direta. Consideramos que a utilização de um padrão de Objetos de Acesso de Dados consistente (DAOs) já atende a uma boa estrutura de acesso ao banco e permite fácil incorporação de outros frameworks de persistência
* A Classe “Pessoa”, requisitada pelo usuário como sendo classe abstrata não vai ser considerada entidade do modelo de dados, uma vez que decidimos usar o modelo relacional puro, que não contempla herança, e não o modelo Objeto-Relacional, desta forma, permitindo que o sistema possa ser implementado em bancos de dados mais simples que não suportem o modelo Objeto-Relacional.

# Arquitetura

## Modelo de Componentes

A arquitetura do projeto utiliza o padrão MVC de construção, desta forma, constituído de 3 blocos principais, a parte de “Modelo” onde se encontram o banco de dados, os objetos de dados “POJO”, também utilizados como objetos de transferência de dados por todo o sistema, no padrão TO (transfer Object), já que neste caso simples, ambos tem a mesma estrutura e funcionamento. A seção de Modelo também inclui os gerenciadores do banco de dados, neste caso, escolhidos para acesso ao banco “Derby” do Apache (JavaDB).

O bloco de visualização, interface de entrada e saída de dados irá compartilhar um componente com o controlador, componente este isolado da interface gráfica (GUI) e do controlador principal. Este componente age como um conector entre o controlador e a view na intenção de futura reposição para outras arquiteturas, como WEB ou Plataformas de Aplicativos móveis. Iremos esclarecer detalhes mais adiante.



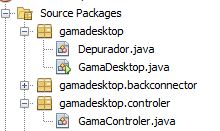
## Modelo de Classes

### Controlador Principal

O controlador principal (GamaControler), seguindo o padrão MVC, neste caso, sem dependência direta da view e apenas do Modelo, contém os métodos de negócios CRUD para manipulação do cadastro dos usuários, cargos e perfils, podendo ser usado com outras interfaces da VIEW sem nenhuma modificação direta.

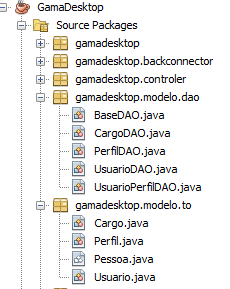
Ele é instanciado no ponto de entrada de programa, a classe de execução principal (GamaDesktop)e passado para o conector FrontEnd-BackEnd que inicializa propriamente o sistema.

Construimos um SINGLETON de depuração simples (Depurador), dado que as configurações e capacidade mais avançadas dos frameworks de depuração, como Log4j2 iriam requerer mais tempo de instruções e configurações tanto para o desenvolvimento como para o cliente implementar.



### Modelo - Classes de Dados e DAOs

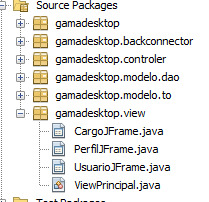
As classes de dados, usadas também como objetos de transferências de dados assim como seus componentes de acesso a dados estão abaixo organizados e com as devidas heranças de compartilhamento requeridas na especificação do projeto



### Objetos de Visualização

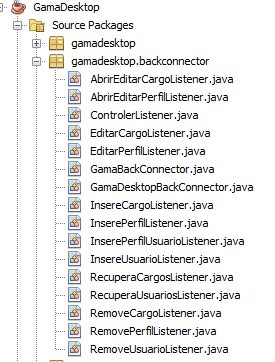
Os objetos de visualização (Frames em Swing), cujos layouts podem ser vistos na seção seguinte, estão encapsulados em um único componente (ViewPrincipal), responsável pelo direcionamento das diversas funcionalidades de acesso as componentes visuais, possibilitando uma modificação ou incorporação das 3 views de cadastro em uma só, se necessário em subsequentes versões.

Estes objetos foram construídos com ajuda do editor de design visual da ferramenta NetBeans, POREM, nenhuma das funcionalidades de incorporação direta de resposta a eventos de botões foi usada para manter o isolamento da VIEW e o CONTROLADOR do padrão MVC. Note, por observação do código, que o padrão e componentes de Listeners foram criados externamente e adicionados aos componentes de resposta de eventos através de novos métodos nestas views. Desta forma, apenas os Listeners acessam o controlador, ou neste caso, o conector front-backEnd, mantendo estes objetos sem nenhuma dependecia do controlador ou dos DAOs.



### Objetos de interface Front-BackEnd

Como explicado na subseção anterior, este componente (GamaDesktopConnector), recebe o controlador e as views como referencias e após, cria e conecta os listeners que vao receber as ações das views e redirecionar para o controlador principal. Ele especializa um conector geral (GamaBackConnnector), de forma que múltiplos tipos de conector podem ser especializados mudando os tipos de view, por exemplo, para uma versão WEB.

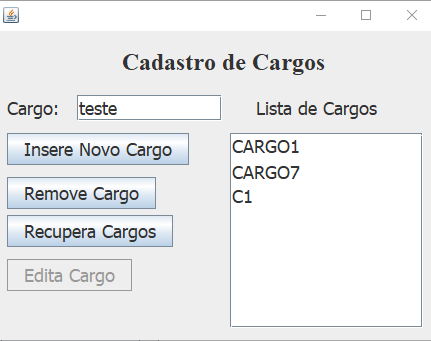


# Interface com o Usuário

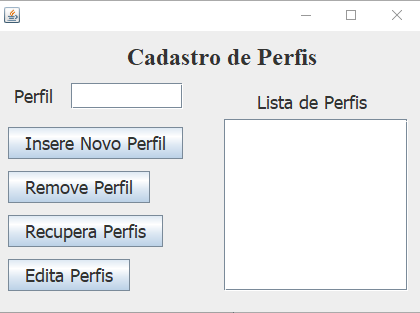
As figuras abaixo mostram as três Views da interface gráfica com o usuário. Podemos acrescentar, remover ou recuperar os usuários já inseridos no banco, porem, ao inicializarmos o sistema, ele já buscará e mostrará a lista de usuários existentes na lista lateral.



O cargo do usuário é adicionado automaticamente quando o usuário é inserido, contudo, é também possível abrir uma tela adicional para editar estes cargos



A interface principal adiciona usuários com múltiplos perfis que são adicionados (Acrescenta Perfil). Se os perfils não existirem, são adicionados automaticamente, contudo, é possível editar os perfils separadamente através do botão de “Editar Perfis” que abre um novo painel



# Modelo de Dados

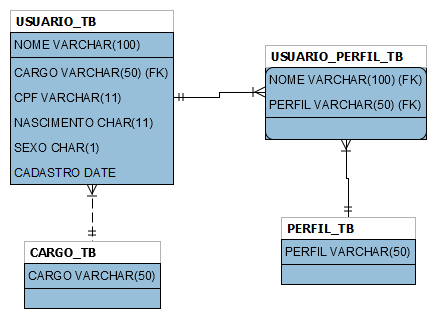
## Modelo Conceitual

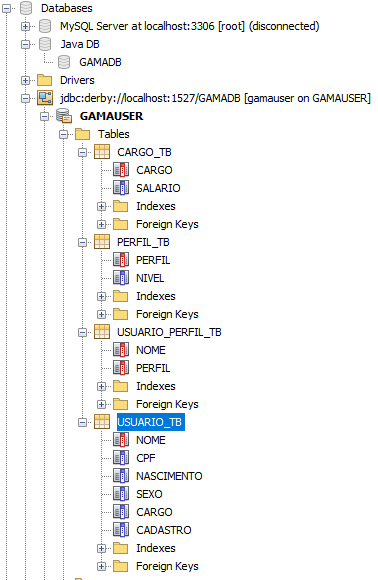
O Sistema se constitui de um cadastro para usuários, os quais possuem alguns atributos básicos e um relacionamento com duas entidades simples, Cargo e Perfil. Tais entidades, que de outro modo seriam atributos do Usuário, foram destacadas como entidades, dado a sua necessidade de funções de edição e provavelmente, em futuras versões, possuírem elas mesmas atributos, como por exemplo, salário do cargo. O sistema descreve a possibilidade de um usuário poder ter mais de um perfil, e desta forma, formando uma relação M:N a ser modelada com uma relação (tabela) além das entidades descritas:

* **Entidade Pessoa** – Requisitada dentro da modelagem de classes como uma entidade pai abstrata e de outro modo não sendo indicada como tal, nesse caso, indicamos como entidade se olharmos do ponto de vista de um modelo de dados Objeto-Relaciona. Contudo, dado a simplicidade do modelo e a premissa de um banco de dados que não suporte o modelo Objeto-Relacional, não iremos levar essa entidade ao modelo lógico
* **Entidade Usuário** – Uma herança da entidade pessoa e irá conter os campos obrigatórios de Nome, CPF e cargo. Sendo este último, um relacionamento com uma outra entidade, uma vez que através de requisitos do usuário, ele poderá vir a ter mais atributos.
* **Entidade Cargo** – Apesar de ter apenas um atributo, sua própria chave, deve permitir edição, sendo, portanto, uma entidade a parte e relacionada de 1:N com os usuários (todo usuário deve ter um cargo e mais de um usuário pode ter o mesmo cargo)
* **Entidade Perfil** – Um usuário pode ter mais de um perfil e tal perfil pode ser atribuído a mais de um usuário, desta forma, teremos uma relação M:N, o qual vai levar a criação de uma tabela intermediária.

## Modelo Lógico/Fisico

O modelo de dados é bastante simples e genérico. Utilizamos o JavaDB (Derby) para implementação mas o mesmo modelo pode ser simplesmente aplicado ao MySQL. Testes foram feitos mas não ficaram prontos a tempo para esta versão.





# Ferramentas

Utilizamos as seguintes ferramentas e pacotes de recursos para desenvolvimento do sistema. Para este protótipo inicial, recomendamos a instalação da Ferramenta NetBeans para execução do projeto, assim como os drivers do Derby, que podem ser obtidos diretamente através da própria ferramenta. O Banco de Dados Derby pode ser incorporado utilizando a copia do diretório GAMADB no diretório do GitHub do Projeto

* Java SE 16.1
* Netbeans V.12
* JavaDB – Banco de Dados Derby do Apache
* Git / GitHub – GitHubDesktop com disponibilidade publica.

# Operação e Recursos

Referencias de disponibilidade de código e recursos:

* Código postado no Github como público: <https://github.com/andrelb2000/GAMAUSUARIOS>
* Projeto a ser aberto utilizando NetBeans(no GitHub): GAMAUSUARIOS/GamaDesktop
* Banco de Dados disponível no GitHub em: GAMAUSUARIOS/GAMADB
* Este documento: GAMAUSUARIOS/Projeto GAMA - André Luiz Braga