HuntingBytes

Eduardo Alexandre de Amorim
Gabriel André Camargo Pimentel de Barros
João Pedro Freire de Andrade
João Pedro Oliveira da Silva
Lucas Lins Pereira da Silva
Moésio Wenceslau da Silva Filho

The Cooks' Books

Repositório GitHub: https://github.com/HuntingBytes/The-Cooks-Books

Diagrama de classes:

Abaixo existem 2 versões do diagrama de classes, uma representação mais simples que abstrai os métodos e atributos das classes focando principalmente nas associações, e uma representação mais detalhada que apresenta os atributos das classes e seus métodos principais (Não incluindo todos os getters/setters).

As imagens na resolução original podem ser encontradas no repositório do projeto no ***Github***. Uma descrição detalhada das classes e suas associações pode ser encontrada ao fim do documento, tratando algumas classes separadamente.

Resumo das mudanças em relação aos diagramas iniciais:

Após rever e iniciar a implementação do diagrama e das demais classes do sistema, percebemos que centralizar tudo na classe *Usuario* por meio de composições tornava difícil implementar repositórios diferentes para algumas classes. Por exemplo, manter uma lista de **CadernoReceitas** em **Perfil** tornava difícil manter um repositório somente com **CadernoReceitas** (Já que **CadernoReceitas** não possuía referência para o **Perfil** que o possui).

Por conta disso, alteramos algumas composições do diagrama. No lugar de fazer parte diretamente de uma outra classe, agora mantemos um *identificador* que permite relacionar tais classes.

Também percebemos que algumas classes não precisavam existir, ao menos no momento, então fizemos a remoção (posteriormente podemos adicionar novamente) e pequenas alterações nas demais classes que as utilizavam.

Lista de mudanças:

- **Usuario** e **Perfil** agora são uma única classe com o nome **Usuario**;
- **Imagem** foi removida e mantemos apenas uma String com o caminho para uma imagem nas classes que a utilizavam;
- **Usuario** não possui mais uma lista de **CadernoReceitas**;
- **CadernoReceitas** não possui mais uma lista de **Receita**, e agora possui um *id* que identifica o **Usuario** que possui esse caderno e um *id* que identifica o caderno;
- **Receita** possui um *id* que identifica qual **CadernoReceitas** a possui e um *id* que identifica a receita;
- **MinhasReceitas** foi removida, não faz-se mais necessário manter essas informações em uma classe própria.
- Outras mudanças menores em atributos e métodos de algumas classes;

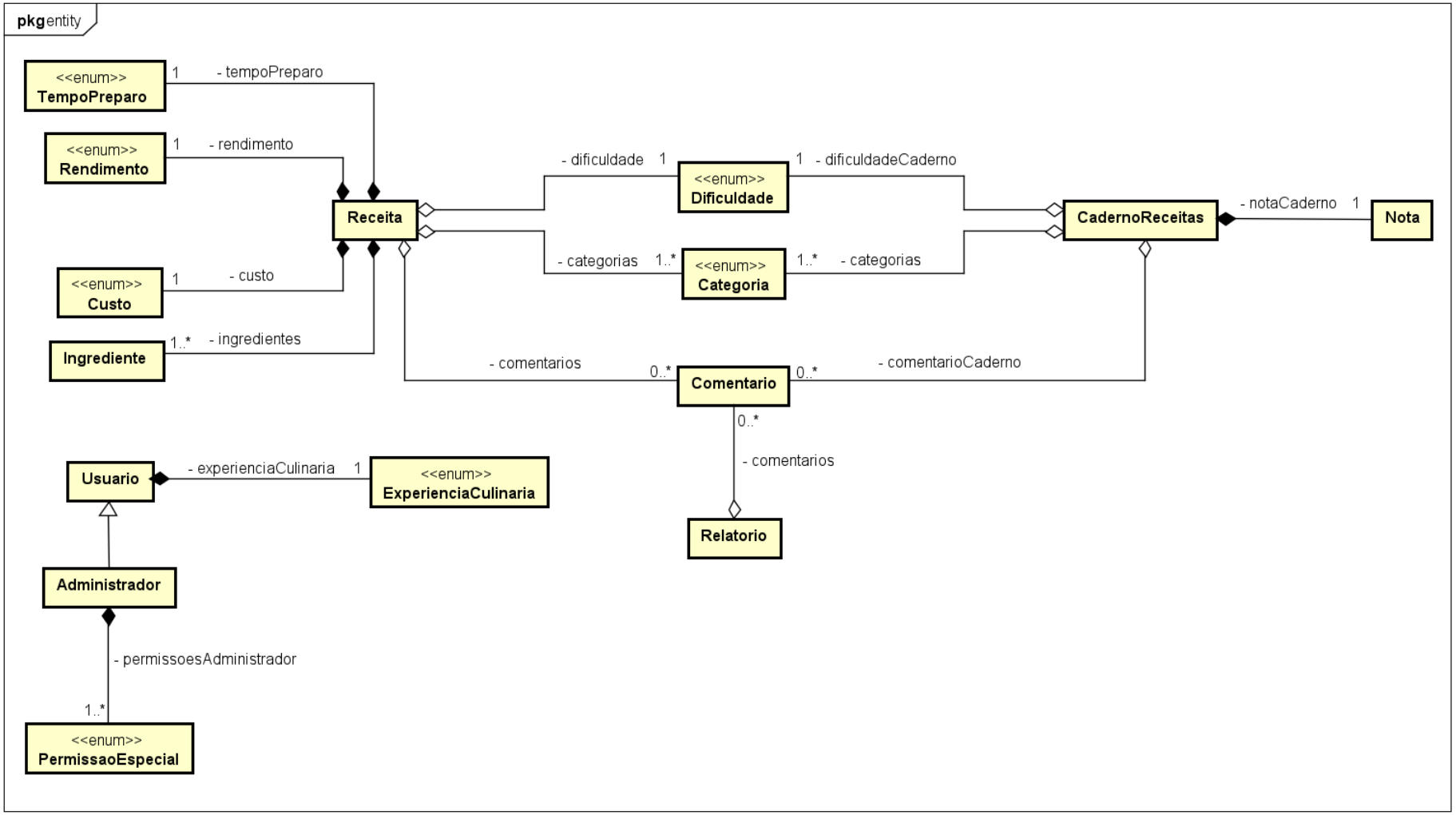


Diagrama de classes simples. Principais associações entre as classes de domínio.

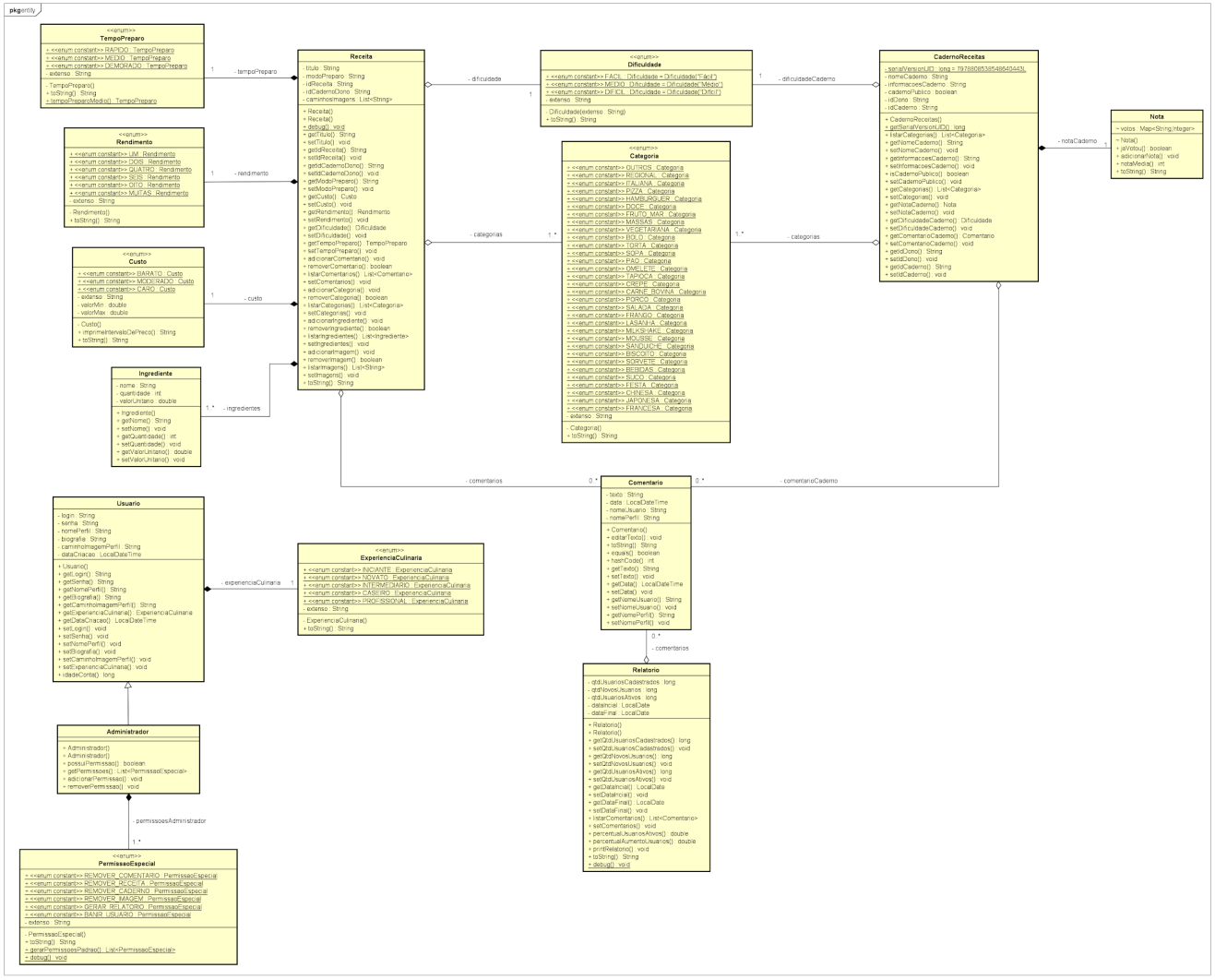
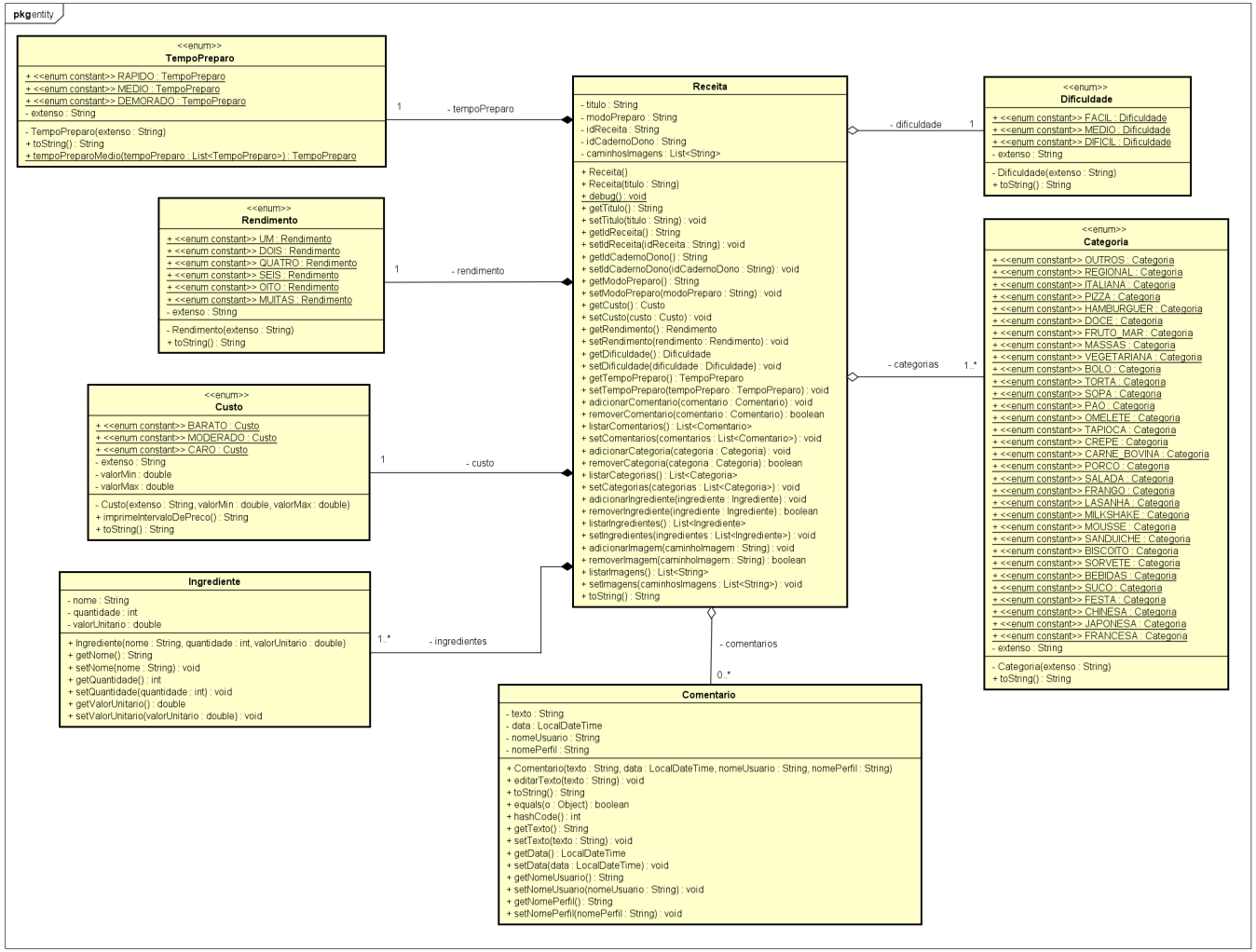


Diagrama de classes detalhado. Principais associações, atributos e métodos (parâmetros omitidos).

A Classe Receita e suas associações

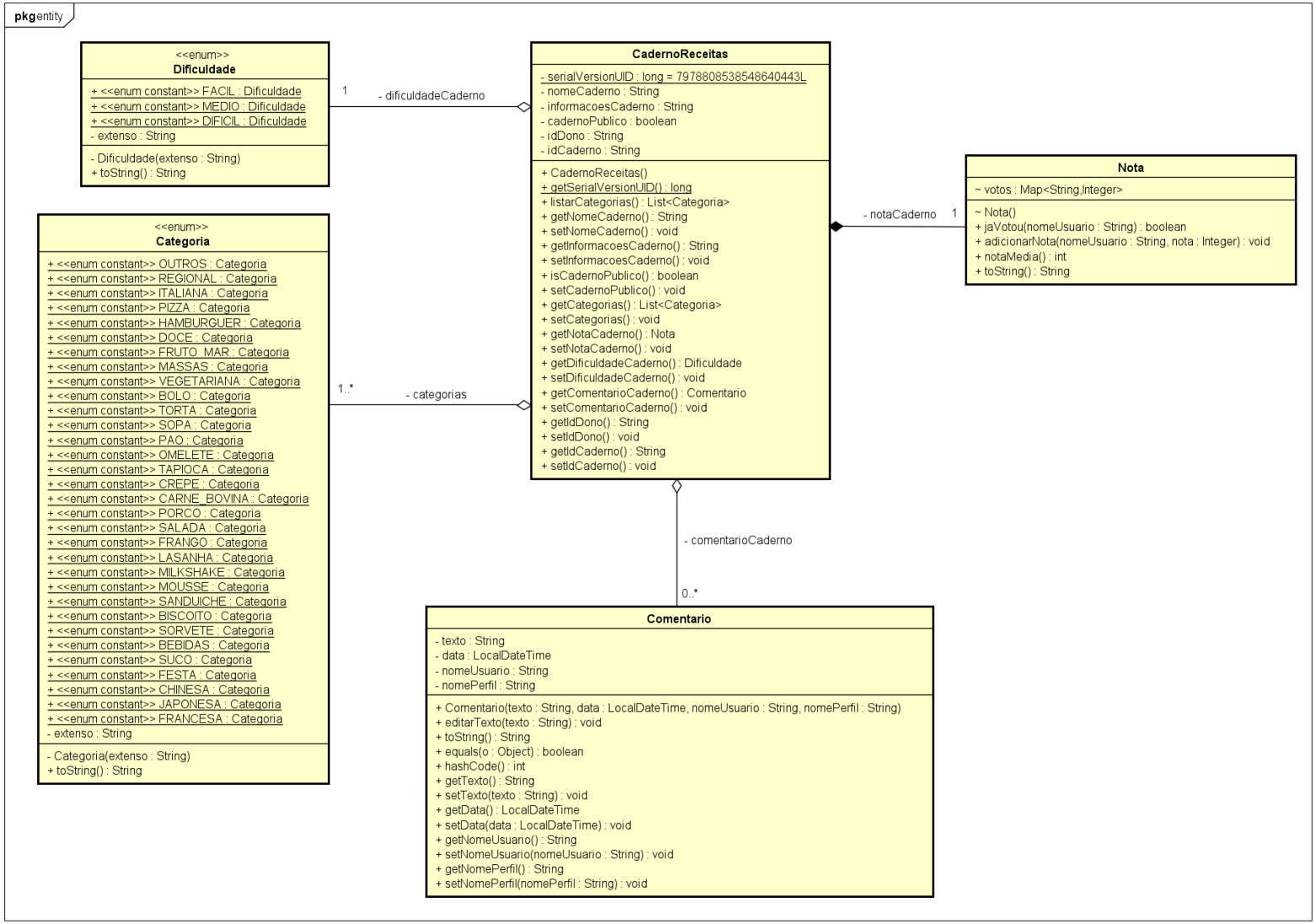


A classe **Receita** é uma das principais classes do sistema. Ela é composta pelas classes **Custo**, **Rendimento**, **TempoPreparo** e **Ingrediente**. Ela também possui **Categoria**, **Dificuldade** e **Comentario**. Essa classe é responsável por abstrair a noção de uma receita real, dessa forma possui uma lista de *ingredientes* (List<**Ingrediente**>), um *rendimento* (**Rendimento**), um *tempo de preparo* (**TempoPreparo**), um *custo* (**Custo**), uma lista de *categorias* (List<**Categoria**>), uma *dificuldade* (**Dificuldade**), uma lista de *comentários* (List<**Comentario**>), uma lista de *imagens* (representada pelo caminho para o arquivo da imagem List<String>), um *nome* (String), um *modo de preparo* (String), um *identificador único* da receita (String), um *identificador* do caderno (**CadernoReceitas**) que ela faz parte.

As classes **Rendimento**, **Dificuldade**, **Custo**, **TempoPreparo** e **Categoria** são classes enumeradas (pelo fato delas possuírem instâncias discretas e específicas) responsáveis por abstrair alguns conceitos usados no dia a dia. Elas apresentam métodos que permitem a fácil manipulação (Cálculo do valor médio, representação como String) durante as operações realizadas pelo sistema. Essas classes possuem uma relação de composição (**Rendimento**, **Custo**, **TempoPreparo**) e agregação (**Categoria, Dificuldade**) com **Receita**.

A classe **Comentario** é responsável por agregar mais informações a uma **Receita**. Os comentários são notas extras, adicionadas pelo autor, que não se encaixam em nenhum outro atributo de uma receita. Ela possui uma relação de agregação com **Receita**.

A Classe CadernoReceitas e suas associações



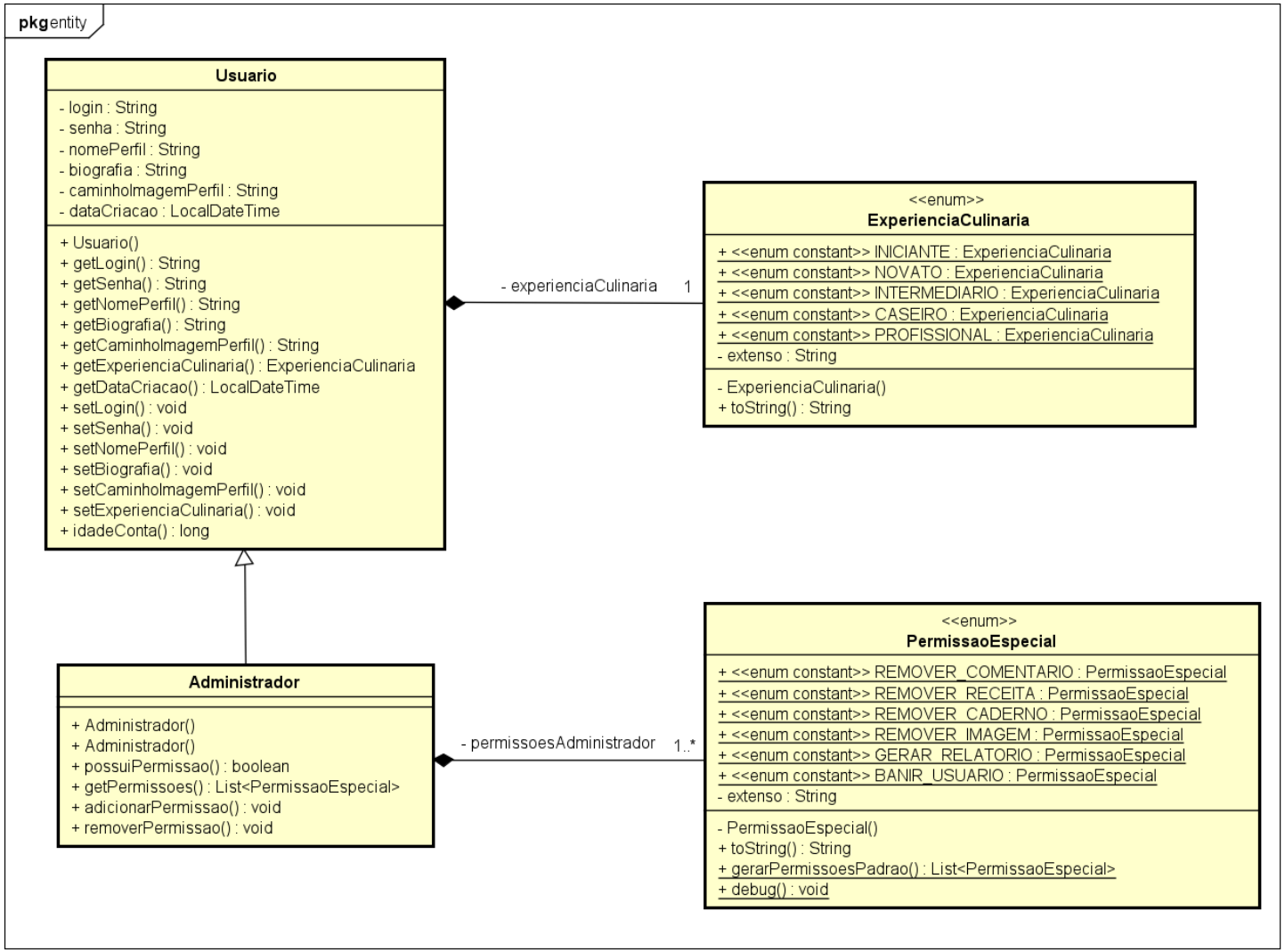
A classe **CadernoReceitas** também é uma das principais classes do sistema. Ela é composta pela classe **Nota** e possui relação de agregação com **Dificuldade** e **Comentario**. Essa classe é responsável por abstrair a ideia de um conjunto de receitas criadas pelo usuário. Ela também possui: comentários; notas (o usuário poderá avaliar o caderno de receita como um todo e dar sua opinião através de uma nota); dificuldade média do caderno (média aritmética da dificuldade das receitas agrupadas nele); identificador único do caderno no sistema; identificador do usuários que possui o caderno.

A classe possui métodos que permitem alterar seus atributos e obter os valores desses atributos.

A classe **Dificuldade** é uma classe enumerada (pelo fato dela possuir instâncias discretas e específicas) responsável por representar a média geral das dificuldades das receitas contidas no caderno. Ela apresenta métodos que permitem a fácil manipulação (Cálculo do valor médio, Representação por String) durante as operações realizadas pelo sistema. Essa classe possui uma relação de agregação com **CadernoReceitas**.

As classes **Nota** e **Comentario** são responsáveis por agregar mais informações ao **CadernoReceitas**. As notas permitem representar a avaliação média dada pelos usuários, já os comentários são um artifício disponível para que outros usuários possam avaliar o caderno do autor e expressar suas opiniões de maneira mais extensa, funcionando como uma extensão para a nota.

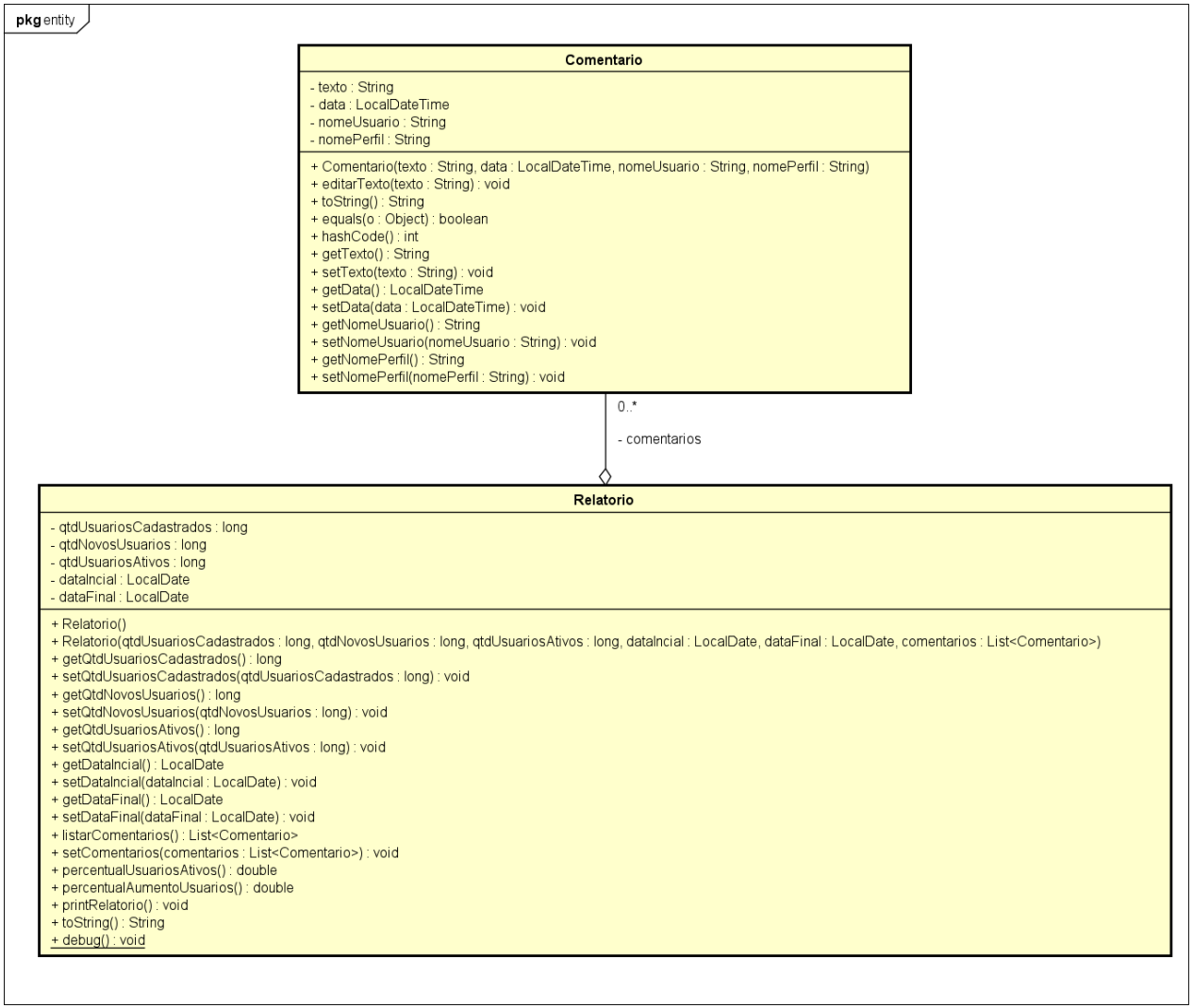
A Classe Usuario e suas associações



A classe **Usuario** é a classe mãe de Administrador. Ela possui os campos: *login*, *senha*, *dataCriacao* e *nome de perfil*, *biografia*, *caminhoImagemPerfil*, caso seja uma conta de usuário padrão. *Login* e *senha* são usados para acessar a plataforma, *dataCriacao* guarda a data em que o usuário foi criado, *nomePerfil* é o nome exibido para outros usuários, *biografia* contém informações sobre o usuário e *caminhoImagemPerfil* indica onde está o arquivo com a imagem do usuário. Dos métodos da classe, o *idadeConta()* retorna há quanto tempo existe a conta. Ela também possui composição com **ExperienciaCulinaria**, que representa a experiência do usuário com culinária.

Administrador é a classe filha de **Usuario**, portanto, herda os campos e métodos, além dos próprios métodos e o seu único campo, um enum de permissões, que dá a possibilidade ao administrador realizar ações no sistema que um usuário comum não pode, como remover comentários e remover uma receita. **Administrador** possui dois construtores, um caso já lhe seja passada uma List de permissões e um caso não seja passada, o que acarreta na necessidade de acessar o método *adicionarPermissao()* (que possui em contraparte, o *removerPermissao()*). Além desses métodos, há o *possuiPermissao()* para checar se o administrador possui a permissão necessária para efetuar uma ação (como as descritas acima).

A Classe Relatorio e suas associações



Vide o que foi colocado no *REQ6* da entrega 1 do projeto, ***a classe Relatorio existe para fornecer algumas informações cruciais acerca do sistema ao administrador em um determinado intervalo de tempo que ele escolher, quando solicitado***.

Ela possui os seguintes campos: *quantidadeNovosUsuarios*, *quantidadeUsusariosCadastrados*, *quantidadeUsuariosAtivos*, *dataInicial*, *dataFinal* e a lista dos comentários feito pelos usuários sobre o sistema.

Tem uma relação de agregação (ou seja, estamos enfatizando que se um relatório que possuía um certo comentario deixar de existir, esse comentário continua existindo) com a classe **Comentario**, que vai fornecer os feedbacks necessários para construir o relatório (detalhe é que como mostrado no UML acima, o relatório pode também não conter nenhum feedback fornecido e ainda assim ser construído). Usuários administradores podem solicitar a criação de um **Relatorio** entre duas datas para o sistema (que será o responsável por criar um objeto **Relatorio**).

Possui um construtor e os seguintes métodos: *percentualAumentoUsuarios(), percentualUsuarioAtivos()*, *listarComentarios()* e o *toString()*.