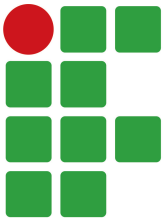


INSTITUTO FEDERAL
Goiás
Câmpus Anápolis

Filipe Beserra Maia

Modelo para controle de acessos baseado em função utilizando segurança em nível de linha

Anápolis-GO

2022

Filipe Beserra Maia

Modelo para controle de acessos baseado em função utilizando segurança em nível de linha

Trabalho de Curso submetido ao Instituto Federal de Goiás - Câmpus Anápolis como parte dos requisitos necessários para a obtenção do Título de Bacharel em Ciência da Computação.

Instituto Federal de Goiás - Câmpus Anápolis

Orientador: Prof. Me. Alessandro Rodrigues

Anápolis-GO

2022

FICHA CATALOGRÁFICA

M217m	MAIA, Filipe Beserra Modelo para controle de acessos baseado em função utilizando segurança em nível de linha / Filipe Beserra Maia – – Anápolis: IFG, 2022. 64 p. : il. color. Orientador: Prof. Me. Alessandro Rodrigues Trabalho de Conclusão do Curso de Ciência da Computação, Instituto Federal de Goiás, Campus Anápolis, 2022. 1. Controle de acesso granular aos dados. 2. Controle de acesso baseado em funções. 3. Segurança em nível de linha. 4. RBAC. 5. RLS, 6. Informática. I. RODRIGUES, Alessandro orien.. II. Título.
	CDD 005.8

Ficha catalográfica elaborada pelo Bibliotecário Matheus Rocha Piacenti CRB1/2992
Biblioteca Clarice Lispector, Campus Anápolis
Instituto Federal de Educação, Ciência e Tecnologia de Goiás

INSTITUTO FEDERAL
Goiás

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
SISTEMA INTEGRADO DE BIBLIOTECAS

TERMO DE AUTORIZAÇÃO PARA DISPONIBILIZAÇÃO NO REPOSITÓRIO DIGITAL DO IFG - ReDi IFG

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia de Goiás, a disponibilizar gratuitamente o documento no Repositório Digital (ReDi IFG), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IFG.

Identificação da Produção Técnico-Científica

[ ]	Tese	[ ]	Artigo Científico
[ ]	Dissertação	[ ]	Capítulo de Livro
[ ]	Monografia – Especialização	[ ]	Livro
[x]	TCC - Graduação	[ ]	Trabalho Apresentado em Evento
[ ]	Produto Técnico e Educacional - Tipo: _____		

Nome Completo do Autor: Filipe Beserra Maia
Matrícula: 20191060140401
Título do Trabalho: **Modelo para controle de acessos baseado em função utilizando segurança em nível de linha**

Autorização - Marque uma das opções

1. (x) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso aberto);
2. ( ) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG somente após a data ___/___/_____ (Embargo);
3. ( ) Não autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso restrito).

Ao indicar a opção **2 ou 3**, marque a justificativa:

( ) O documento está sujeito a registro de patente.
( ) O documento pode vir a ser publicado como livro, capítulo de livro ou artigo.
( ) Outra justificativa: _____

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a autor/a declara que:

i. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
ii. obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;
iii. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

Anápolis, 01/02/2023.
Local Data

Filipe B. Maia
_____

Assinatura do Autor e/ou Detentor dos Direitos Autorais

ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO II

Aos 16 dias do mês de dezembro de 2022, às 11:07 horas, em sessão pública na sala multimeios III do Câmpus Anápolis / IFG, na presença da Banca Examinadora presidida pelo(a) Professor Orientador Alessandro Rodrigues e Silva e composta pelos examinadores:

1. Alexandre Belezzi José
2. Daniel Xavier de Souza,

o(a) Discente Filipe Beserra Maia , matrícula 20191060140401
apresentou o Trabalho de Conclusão de Curso intitulado:
Modelo para controle de acessos baseado em função utilizando segurança em nível de linha
como requisito curricular indispensável para a integralização do Curso de Graduação de Bacharelado em Ciência da Computação.
Após reunião em sessão reservada, a Banca Examinadora deliberou e decidiu pela aprovação do referido trabalho, divulgando o resultado formalmente ao discente e demais presentes e eu, na qualidade de Presidente da Banca, lavrei a presente ata que será assinada por mim, pelos demais examinadores e pelo discente.

_____
Presidente da Banca Examinadora

_____
Examinador 01

_____
Examinador 02

_____
Discente

Documento assinado eletronicamente por:

- **Daniel Xavier de Sousa, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 01/02/2023 14:46:53.
- **Alexandre Bellezi Jose, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 01/02/2023 13:40:50.
- **Filipe Beserra Maia, 20191060140401 - Discente**, em 01/02/2023 11:41:33.
- **Alessandro Rodrigues e Silva, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 01/02/2023 11:40:34.

Este documento foi emitido pelo SUAP em 01/02/2023. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifg.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 369453
Código de Autenticação: af60b49185

Lista de ilustrações

Lista de tabelas

Lista de algoritmos

Lista de abreviaturas e siglas

RBAC	Role Based Acess Control
RLS	Row Level Security
SGBD	Sistemas Gerenciadores de Banco de Dados
HTML	Linguagem de Marcação de Hipertexto
UML	Linguagem de Modelagem Unificada
MER	Modelo Entidade Relacionamento
DER	Diagrama de Entidade Relacionamento
SQL	Structured Query Language
SI	Segurança da Informação
HTTP	Hypertext Transfer Protocol
SSD	Solid State Drive

Sumário

Resumo

O controle de acesso granular aos dados é um problema em aberto, para o qual há poucos estudos que o resolvem a partir de um único método ou solução. Em um ambiente corporativo, a tecnologia da informação é um recurso estratégico que contribui para a sobrevivência e permanência das corporações no mercado, fazendo com que essas corporações utilizem softwares que objetivam não só centralizar dados e informações, como também auxiliar nos processos administrativos e organizacionais.

Sabendo dessas informações, é possível entender que seja de preocupação dessas organizações implantar e manter ambientes computacionais seguros, visto que proteger os dados contra o acesso não autorizado é de suma importância: pesquisas mostram que uma considerável parcela de ataques são conduzidos por indivíduos que já possuem acesso ao sistema, o que prova que negligenciar o controle de acesso aos dados permitindo o acesso indevido pode ocasionar em alteração ou destruição de informações importantes.

Este trabalho visa propor a implementação de um framework que une o Controle de Acesso de Dados Baseado em Papéis (RBAC) à Segurança em Nível de Linha já implementada por alguns sistemas gerenciadores de banco de dados. Essa implementação permite que as informações acessadas ou gravadas por usuários de aplicações sejam controladas diretamente no banco de dados, propiciando assim uma melhor garantia do controle de acesso a leituras e escritas de cada registro, tanto de maneira geral quanto de maneira específica.

Esse modelo define uma estrutura utilizada para implementar políticas de segurança atreladas diretamente a objetos do banco de dados, possibilitando assim a existência de regras de acesso que delimitem as informações disponíveis a cada usuário. Além disso, esse procedimento contribui para a transparência e visibilidade organizacional quanto ao acesso aos dados, uma vez que todos os procedimentos, regras e papeis responsáveis por determinadas instâncias serão explicitamente definidos e facilmente gerenciados. Finalmente, o modelo possibilita uma considerável redução da produção manual de código-fonte destinado ao controle de acesso de dados, propiciando uma maior facilidade de manutenção e organização do sistema como um todo.

- **Palavras-chave:** Controle de acesso granular aos dados, Controle de acesso baseado em funções, segurança em nível de linha, RBAC, RLS

Abstract

Fine-grained access control is an open problem, for which there are few studies that solve it based on a single method. In a corporative environment, information technology is a strategic resource that contributes for survival and permanence of corporations in business, causing these corporations to use softwares that have not only the centralization of data and information as an objective, but also the easing of administrative and organizational processes. Knowing this, it's understandable that those organizations care that their computational environment is as safe as possible, since protecting data against unauthorized access is of the utmost importance: studies show that a considerable portion of attacks are conducted by individuals that already had access to the system, which proves that neglecting to control data access, allowing improper access, can cause alteration or destruction of important information.

This work intends to propose the implementation of a framework that joins Role-Based Access Control (RBAC) to the Row Level Security that is already present by some database management systems. This implementation allows for information accessed or written by application users to be controlled directly in the database, which in turn makes it easier to guarantee that writings and readings of each entry is controlled, both in general and specific ways. This model defines a structure used to implement security policies attached directly to database objects, making the existence of access rules that limit the info available to each user possible. In addition, this procedure contributes to the transparency and organizational visibility related to data access, since every procedure, rule and role responsible for each instance is explicitly defined and easily managed. Finally, this model reduces considerably the manual writing of code destined to data access control, allowing for an easier upkeep and organization of the system as a whole.

- **Key words:** Fine-grained access control, Role-Based Access Control, Row-Level Security, RBAC, RLS

1 Introdução

Desde o seu surgimento, a tecnologia da informação no ambiente corporativo passou a ser vista como um recurso estratégico que contribui para a sobrevivência e a permanência das organizações no mercado (HAMMER MICHAEL E CHAMPY, 1994). Tais corporações utilizam softwares que objetivam centralizar dados e informações, bem como, auxiliar nos processos administrativos e organizacionais.

É de preocupação das organizações implantar e manter ambientes computacionais seguros, visto que proteger os dados contra o acesso não autorizado é de suma importância. Assim, negligenciar o controle de acesso a informação pode ter como consequência o acesso indevido, alteração ou destruição de dados, conduzido, inclusive, por indivíduos que já possuem credenciais de acesso ao sistema. Pesquisas mostram que os ataques internos, ou seja, aqueles conduzidos por usuários autenticados, chegam a 34% dos casos estudados (VERIZON, 2019).

Ao usar uma aplicação, normalmente, cada usuário precisa passar por um processo de autenticação. Tal processo é utilizado para garantir a legitimidade da tentativa de acesso ao sistema. Apenas esta etapa não é suficiente para garantir quais informações estarão disponíveis para um determinado usuário, desse modo, faz-se necessário a coexistencia deste método com outros mecanismos de segurança como o controle de acesso, o qual é responsável pela disponibilização e controle das informações (SANDHU; SAMARATI, 1994).

Um dos métodos de controle de acesso é o RBAC (Role Based Access Control). Nele as decisões são baseadas no tipo de função exercida pelo usuário como integrante de uma organização, isto é, a segurança é organizada em um formato bastante semelhante à estrutura da organização. Além disso, determinados bancos de dados possuem recursos que possibilitam o controle de acesso aos dados de forma granular, permitindo assim estabelecer certas regras para cada tabela, determinando quem pode ter acesso a cada dado contido na mesma, esse recurso é o RLS (Row Level Security).

Tendo em vista a importância de controlar o acesso à informação, o objetivo deste trabalho é propor um modelo que implemente um framework unindo ao RBAC a funcionalidade de controle de acesso aos dados utilizando os recursos do RLS. Deste modo, uma melhor maneira de garantir o direito de leitura e escrita de cada registro de forma tanto generalista quanto específica será implementada. Assim, as informações que serão acessadas ou gravadas terão seu controle feito diretamente no banco de dados, antes de chegar a aplicação.

Para a implementação do framework as tecnologias selecionadas foram a linguagem

de programação Ruby e o SGBD (Sistema Gerenciador de Banco de Dados) PostgreSQL, sendo assim, a partir da especificação completa do modelo, será possível implementá-lo fazendo o uso de tecnologias análogas.

1.1 Justificativa

O crescente uso dos softwares pelas organizações com intuito de auxiliar nos processos organizacionais e administrativos, originou a necessidade de se utilizar melhores instrumentos para gerir a segurança dos dados, visto a possibilidade de que informações confidenciais sejam expostas ou modificadas por pessoas não autorizadas. Considerando estas preocupações, a proteção dos dados é um fator relevante nas companhias.

Nesse sentido, nota-se a necessidade de regular o acesso às informações através da criação e implantação de mecanismos de controle de acesso. O modelo proposto busca prover recursos capazes de auxiliar a restrição ao acesso aos dados. A sua utilidade está na redução do tempo de desenvolvimento relacionado a segurança e controle de acesso dos dados, isso ocorre pois a ferramenta contruida utilizando os conceitos abordados neste trabalho possibilita a geração de políticas de segurança de forma automática, bem como, facilita a manutenibilidade das configurações abordadas anteriormente.

Um dos recursos utilizados na criação desta ferramenta será a linguagem Ruby junto do framework Rails. Para gerir o controle de acesso, comumente uma biblioteca que possui uma estrutura semelhante ao RBAC é utilizada, porém, esta não provê uma forma de controle de acesso aos dados de forma granular. Para realizar tal processo, é necessário criar regras específicas para cada papel que se deseja restringir o acesso às informações.

A figura 1 demonstra um exemplo de como é normalmente feita a verificação via **aplicação** para garantir que um determinado tipo de usuário consiga realizar a alteração de informações de um registro. Neste exemplo um usuário com a função de professor solicita a alteração de informações de um registro contido em uma tabela nomeada curso, portanto, além das consultas acerca das informações do curso e do próprio usuário, a aplicação também verifica o papel do usuário logado e se o mesmo faz parte dos professores do curso em questão, caso seja confirmado,o registro então será alterado.

Neste trabalho, o modelo de controle de acessos proposto unindo RBAC e RLS, permite que a aplicação controle o acesso à leitura e escrita de dados de forma simplificada, uma vez que o banco de dados será responsável por realizar as principais verificações. Isto contribui para uma redução drástica na quantidade de código escrito na aplicação.

Considerando o mesmo cenário utilizado na figura 1, nota-se a drástica redução de código na camada da aplicação exposta na figura 2, propiciada pelo framework Granar. Tal redução é justificada pela realização do controle de acesso realizado diretamente pelo

Figura 1 – Exemplo de controle de acesso feito via aplicação para modificação de dados
Fonte: Elaborado pelo autor (2022)

```
atualiza_tabela_curso("Ciência Da Computação", novos_atributos)

def atualiza_tabela_curso(nome_do_curso, novos_atributos)
  #Busca Informações no Banco sobre o curso
  curso = Course.find_by_name(nome_do_curso)

  #Busca Informações no Banco sobre o usuário
  usuario = User.find_by_id(sessao[:id_usuario_autenticado])
  papel_do_usuario = usuario.find_user_role
  if papel_do_usuario == "Professor"
    #Busca Informações Quem são os professores do curso
    professores_do_curso = curso.professors_ids

    #verifica se o usuário autenticado é professor do curso
    if professores_do_curso.include?(usuario.id)
      curso.update(novos_atributos)
    end
  end
end
```

Figura 2 – Exemplo de código na aplicação quando o controle de acesso é feito utilizando o modelo Granar
Fonte: Elaborado pelo autor (2022)

```
def atualiza_tabela_curso(nome_do_curso,novos_atributos)
  curso = Course.find_by_name(nome_do_curso)
  curso.update(novos_atributos)
end
```

SGDB, sem a necessidade de fazer verificações adicionais na aplicação.

Ademais, geralmente ferramentas que realizam o controle de acesso granular aos dados através da reescrita de consultas possuem a necessidade de alterar o ORM (Object Relational Mapper), isto é, o código SQL gerado automaticamente pela aplicação. Já o modelo Granar elimina esta etapa adicional através da criação de migrations. Tal processo otimiza a manutenção da aplicação, bem como, evita trabalhos extras de modificação e permite o versionamento das alterações realizadas na ferramenta.

Em geral, não há confirmações de que existam métodos de segurança que funcionem sem falhas, vale ressaltar também que a segurança depende das pessoas que utilizam a aplicação. Contudo, utilizar artifícios que ajudam a melhorar os métodos de desenvolvimento inserindo novas camadas de segurança em um ambiente computacional é algo

fundamental.

Garantir a integridade das informações dispostas no banco de dados é algo imprescindível, impedir o acesso aos dados por pessoas não autorizadas é algo essencial para a confiabilidade das informações. Como a finalidade de um sistema gerenciador de banco de dados é servir como repositório central para o armazenamento de registros, os usuários devem poder confiar na informação armazenada (PFLEEGER, 2012).

1.2 Objetivos

1.2.1 Objetivos gerais

A finalidade deste trabalho é propor um modelo para controle de acesso granular aos dados, contribuindo para a segurança através de verificações de acesso à informação. Este modelo terá como base os conceitos do recurso de controle de acesso RBAC aplicados juntamente aos métodos de RLS.

O framework proposto tem por objetivo facilitar a implantação da funcionalidade de segurança, auxiliando o programador na implementação das políticas de restrições no banco de dados. Sendo possível que o processo seja realizado com o mínimo de intervenção possível do profissional, pois através de uma política padrão pré-definida é possível a situações específicas de controle de acesso.

1.2.2 Objetivos Específicos

- Descrever conceitos sobre segurança da informação e controle de acesso aos dados, relacionando-os com a segurança em Banco de Dados.
- Fazer pesquisas e analises de soluções semelhantes.
- Descrever o funcionamento do controle de acesso em nível de linha em SGBD e suas políticas de funcionamento.
- Especificar os requisitos do sistema.
- Criar o modelo conceitual do sistema.
- Compreender os fundamentos e metodologias de projetos de desenvolvimento de software.
- Desenvolver o framework, utilizando os recursos de segurança propostos.
- Avaliar a eficácia da ferramenta através de estudos experimentais, para avaliar o modelo de software proposto.

- Apresentar a análise estatística dos resultados experimentais.

2 Referencial Teórico

2.1 Segurança da informação

Segurança da informação (SI) pode ser conceituada como um conjunto de boas práticas e ações que têm como finalidade resguardar um conjunto de informações. SI, está diretamente relacionada à proteção de um conjunto de dados, com o propósito de garantir a confidencialidade e a integridade dos elementos. Segundo os padrões internacionais (ISO/IEC:17799, 2005), as propriedades básicas de SI, podem ser elencadas como:

- **Autenticidade**: propriedade que garante que a informação é proveniente da fonte anunciada e que não foi alvo de mutações ao longo do processo de armazenamento ou transferência, isto é, não sofreu modificações não permitidas durante tais etapas.
- **Confidencialidade**: propriedade que limita o acesso à informação tão somente às entidades legítimas, ou seja, garante a privacidade dos dados, restringindo seu acesso apenas ao conjunto de pessoas autorizadas.
- **Disponibilidade**: propriedade que garante que a informação esteja sempre disponível, isto é, os dados se encontram sempre a disposição quando solicitados por pessoas autorizadas.
- **Integridade**: propriedade que garante que a informação manipulada mantenha todas as características originais estabelecidas pelo seu proprietário, incluindo controle de mudanças e garantia do seu ciclo de vida.

Desse modo, percebe-se que a segurança da informação não é restrita somente aos sistemas de computadores ou armazenamento, o conceito é bastante amplo com uma vasta aplicabilidade. Assim sendo, nota-se a grande importância de garantir sua aplicação correta e o mínimo de falhas possíveis.

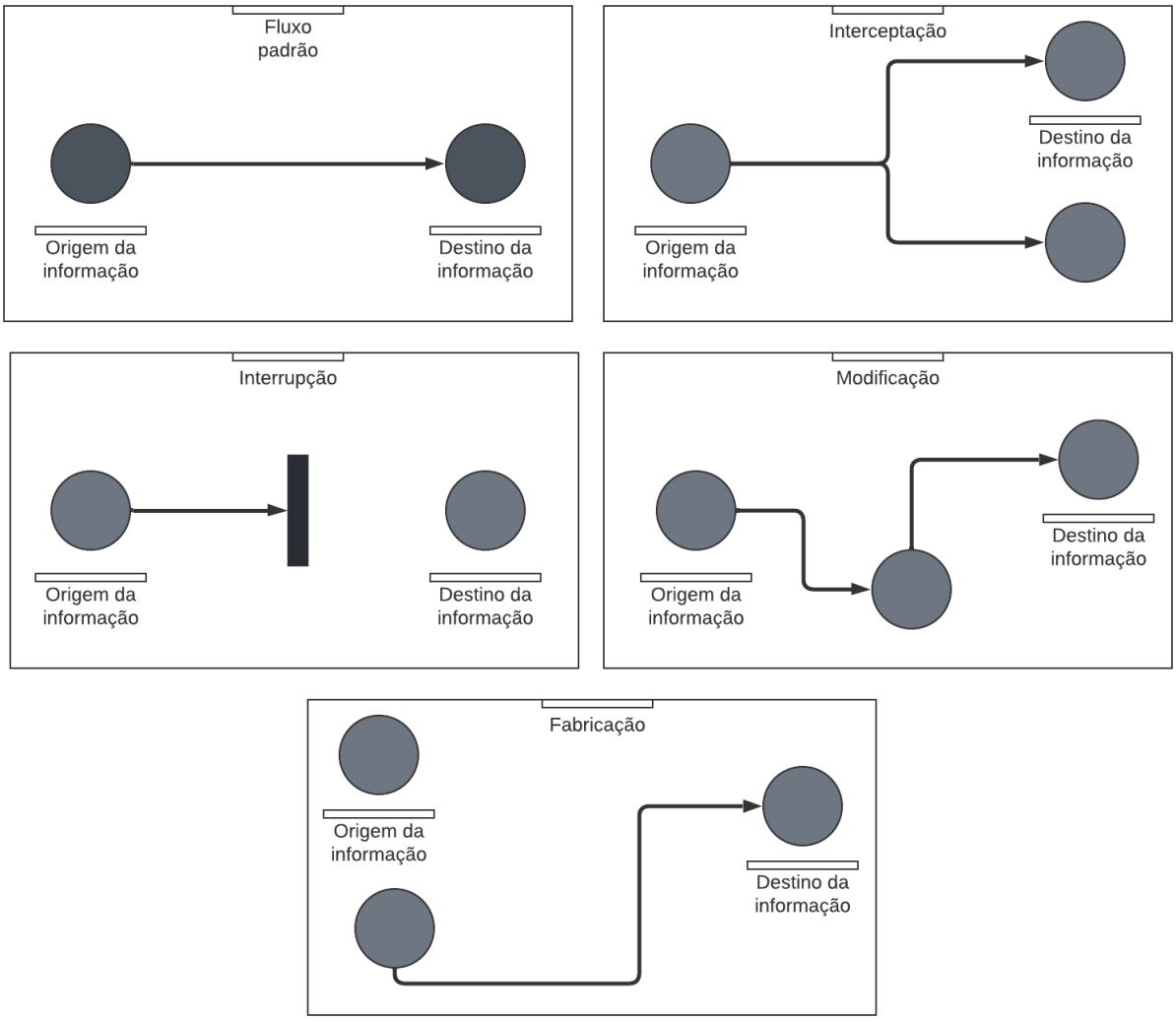
Atualmente o conceito de Segurança da Informação segue o padrão instituído pela norma (ISO/IEC:17799, 2005) e tem como norma técnica a (ISO/IEC:27002, 2013) que objetiva estabelecer um conjunto de instruções e princípios gerais para se iniciar, implantar e manter a gestão de segurança da informação em uma organização.

A (ISO/IEC:27002, 2013) afirma que a informação é um ativo essencial para uma organização e o cumprimento de seus objetivos, sendo assim, necessita ser protegida. Desse modo, nota-se a grande importância dos dados no contexto organizacional, ou seja, a manipulação da informação é percebida como um fator decisivo para impulsionar o alcance dos objetivos, permanência e bons resultados.

Contudo, não basta apenas ter acesso à informação, é necessário garantir que as propriedades de integridade, confiabilidade, disponibilidade e autenticidade estejam sendo aplicadas. Nesse contexto, nota-se um obstáculo, os ataques cibernéticos. Assim, conhecer os principais tipos de ataques em ambientes virtuais é fundamental.

Tais ataques comprometem as propriedades básicas de segurança. Segundo (GUIMARÃES; LINS; OLIVEIRA, 2006) os ataques de segurança podem ser divididos em quatro categorias: interrupção, interceptação, modificação e fabricação.

Figura 3 – Tipos de ataques de segurança.
Fonte: (COELHO, 2014) (Modificado)



- **Interrupção :** Os ataques de interrupção objetivam destruir ou interromper um serviço oferecido. Um exemplo desse tipo ataque é o Denial of Service (negação de serviço), que utiliza do envio de um grande número de requisições para determinado servidor, de maneira que este não consiga atender a todas elas, ocasionando assim,

uma má performance do sistema ou até mesmo, uma interrupção total do serviço fornecido.

- **Interceptação:** Já os ataques de interceptação, objetivam capturar os dados que estão sendo trafegados entre a origem e o destino sem que o sistema perceba, ou seja, rompe a privacidade das informações. A partir desta estratégia, são geradas cópias dos dados furtados para utilização não autorizada.

- **Modificação:** Os ataques de modificação acontecem quando há alteração da informação que está sendo transmitida entre a origem e o destino por alguém que não tem autorização de acesso, ou seja, ataca-se a integridade da informação.

- **Fabricação:** Os ataques de fabricação ocorrem quando alguém externo à organização adentra os canais privados e injeta dados falsificados no tráfego privado de forma não autorizada e anônima, isto é, sem que a origem ou destino, se deem conta de tal inserção.

O foco deste trabalho encontra-se na criação de um mecanismo que molde um fluxo de implementação, tornando viável o impedimento de ataques de modificação e fabricação internos, isto é, com origem em membros já conectados ao sistema, contudo, com autorizações de acesso limitadas.

Uma possível tentativa de acesso indevido, seria um cenário em que um usuário malicioso que possui acesso a uma determinada instância da aplicação, tente acessar ou modificar informações que não tem acesso via aplicação através de um ataque denominado Parameter Tampering Attack. Este ataque consiste na manipulação de parâmetros usados na comunicação web entre cliente e servidor (BISHT et al., 2014), assim, caso não haja uma verificação específica por parte da aplicação, o banco de dados poderá retornar dados ou modificar informações que este usuário mal-intencionado não possui autorização de acessar ou modificar.

Um possível exemplo de Parameter Tampering Attack é a alteração de parâmetros em campos de um formulário. Quando um usuário realiza seleções em campos de uma página HTML, o seu conteúdo é geralmente armazenado como valores em campos de um formulário e ao salvar, são enviadas ao aplicativo Web como uma solicitação HTTP. Esses valores podem ser de diversos tipos, sendo eles, pré-selecionados, texto livre ou ocultos. Todos esses valores são passíveis de manipulação por parte de um invasor, considerando que todo código HTML é processado no computador do cliente.

Campos ocultos são parâmetros invisíveis para o usuário final, normalmente utilizados para prover informações de status para o aplicativo Web. Por exemplo, considere um formulário de pedido de produtos que inclui o campo oculto exemplificado na listagem 2.1.

Listing 2.1 – Exemplo de campo adulterado em um ataque de adulteração de parametros

```
1 <input type="hidden" name="preco" value="84.88">
```

A modificação desse valor de campo oculto fará com que ao submeter o formulário para o aplicativo da Web, este utilize o valor adulterado para seus processos.

Tendo em vista a importância de controlar o acesso à informação, o objetivo deste trabalho é propor um modelo que implemente um framework unindo ao RBAC a funcionalidade de controle de acesso aos dados utilizando os recursos do RLS.

2.2 Role Based Access Control

O RBAC (Role Based Access Control - Controle de acesso baseado em funções) é um tipo de controle de acesso aos dados utilizado por aplicações que restringem o acesso com base na função, ou papel, do usuário com base em privilégios específicos que ele possua.(MASON, 2000)

Quando aplicado, este método garante que cada usuário de uma aplicação seja designado para uma ou várias funções, sendo que elas possuem diferentes privilégios de acesso tanto a dados quanto a funções da aplicação. Neste modelo de controle de acesso, todos os papeis definidos para um usuário podem ser removidos ou alterados facilmente a medida que o sistema evolui ou sofre alterações, aplicando assim, a garantia de flexibilidade para a manutenção do controle de acesso.

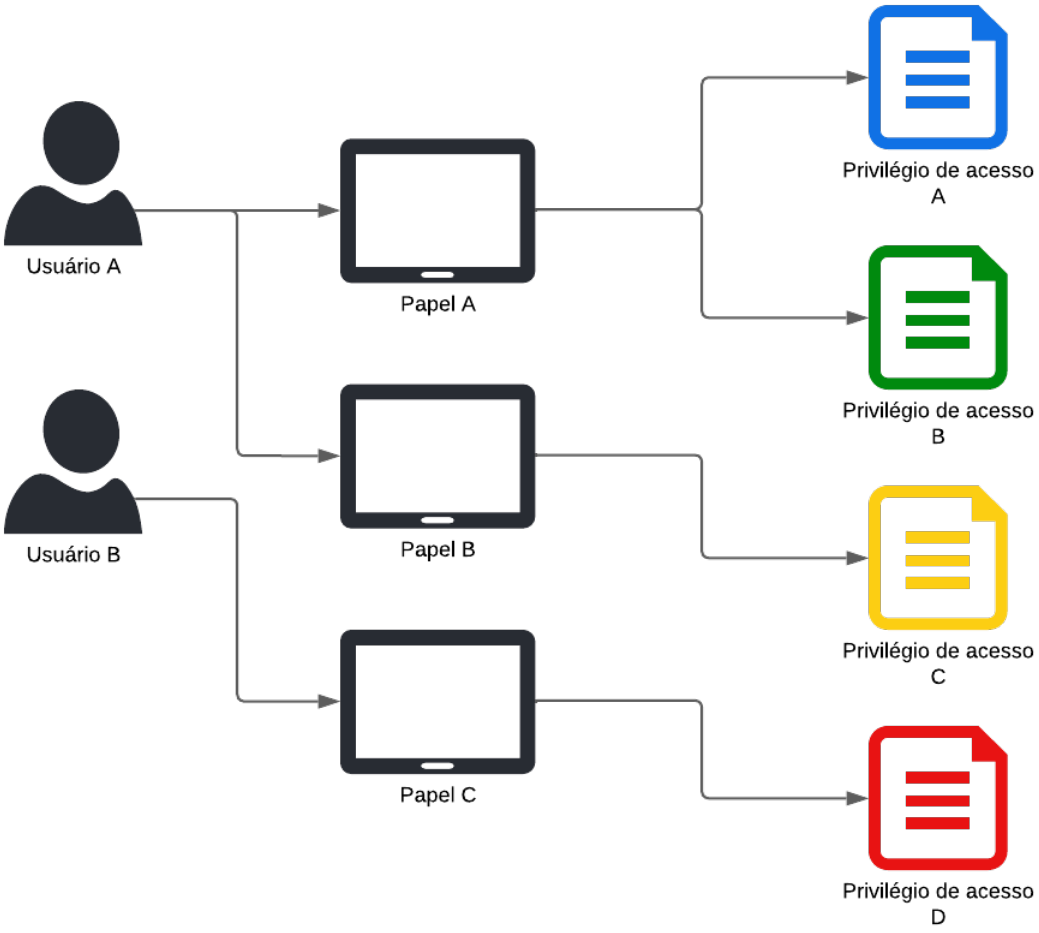
O RBAC é comumente utilizado em sistemas comerciais. Seu grande uso é motivado por sua premissa, isto é, a utilização de uma abordagem que faz a aplicação de conceitos do mundo real de hierarquia para estruturar as permissões e funções do controle de acesso de uma aplicação. (ELMASRI RAMEZ; NAVATHE, 2018)

Na prática, em uma organização, as pessoas possuem diferentes cargos, papeis e responsabilidades. Seguindo esse modelo, na aplicação, é possível atribuir ao usuário uma função com base em seu cargo e responsabilidades. Dessa forma, um usuário pode ser relacionado a vários papéis e cada um com diversos privilégios específicos. Observa-se a na figura 4 um exemplo da estruturação do RBAC.

A definição de papéis é baseada na análise integral do funcionamento de uma organização podendo abranger uma grande variedade de usuários (MASON, 2000). No modelo RBAC, cada utilizador poderá ser associado a um ou vários papéis que podem ou não possuir hierarquia, ademais, cada um destes está associado a um ou mais privilégios de acesso (ELMASRI RAMEZ; NAVATHE, 2018). Os usuários que possuem um determinado perfil estão limitados a interagir com o programa conforme os seus limites de acesso.

Em um sistema que utiliza o modelo RBAC como forma de restringir o seu uso, a filtragem de dados ocorre da seguinte maneira, primeiramente a aplicação assegura que

Figura 4 – Role Based Access Control.
Fonte: Elaborado pelo autor (2022)



o usuário tem direito de acesso a rota que será acessada pela requisição, então busca a informação no banco de dados, realiza o processo de filtragem e devolve os dados buscados para o usuário. De maneira semelhante ocorre o processo de escrita na base de dados, neste processo, antes de inserir um novo registro, a aplicação verifica o direito do usuário de executar tal ação.

Sendo assim, o RBAC tem sua estratégia baseada em funções hierárquicas que espelham o funcionamento organizacional utilizado atualmente. Ademais, as etapas de filtragem acontecem diretamente na aplicação depois de realizada a consulta ao banco de dados.

2.3 Row Level Security

Como visto na seção 2.2, métodos como o RBAC restringem os acessos de maneira generalista, filtrando os dados na aplicação, não contemplando artifícios que garantam

o controle aos dados de maneira granular. Sendo assim, não é possível impedir o acesso indevido a informação em casos onde é necessário controlar o acesso a linhas específicas do banco de dados. Isto é, apenas a utilização do RBAC não é suficiente para garantir o controle de acesso granular aos dados, ou seja, é interessante uni-lo a outros recursos.

Um possível cenário sujeito a melhoria pode acontecer quando um usuário é associado a um papel cujas permissões garantem o acesso a um conjunto de informações, contudo, esse colaborador em específico não deve acessar alguns dados contidos nessa coleção, seja para visualizar ou editar.

Determinados bancos de dados possuem recursos que possibilitam o controle de acesso aos dados de forma granular, permitindo assim estabelecer regras específicas para tabelas estabelecidas, determinando assim, quem pode ter acesso a cada dado armazenado, esse recurso é denominado Row Level Security - segurança em nível de linha (RLS).

O Row Level Security aplica um processo de filtragem nos dados consultados no banco de dados. Tal recurso tem por objetivo limitar os dados que serão acessados, impedindo o acesso indevido à informação a nível de banco de dados por usuários que não possuam direito de acesso.

O RLS é formado por um conjunto de políticas de segurança, anexadas a uma tabela ou view de banco de dados que executam sempre que há uma tentativa de consulta ou alteração das informações (CARTER, 2018). O recurso possibilita o uso do contexto de execução de uma consulta para controlar o acesso às linhas em uma tabela, simplificando a forma como a segurança é gerenciada em uma aplicação.

A restrição de acesso é feita na camada do banco de dados, onde os dados estão armazenados, ao invés de ser na camada da aplicação como acontece no RBAC. Quando o Row Level Security está habilitado, o sistema gerenciador de banco de dados verifica a legitimidade do acesso a cada tentativa de consulta às informações, independente de qual seja a origem da tentativa. Isso torna o sistema de segurança mais robusto e confiável.

A aplicação deste recurso é feita em duas etapas. Primeiro, um usuário administrador deve habilitar o recurso, especificando cada objeto que receberá esta verificação. Segundo, deve ser criado uma política de segurança para este objeto, de modo que esta determine quais dados o usuário beneficiado pela política de segurança pode acessar.

2.3.1 Políticas de segurança

Como sugere (ZHANG et al., 2016), o Row Level Security é composto por políticas de segurança, que podem ser definidas como regras que determinam o modo de acesso a um subconjunto dos dados que o usuário do banco de dados pode acessar. Tais políticas podem ser definidas para cada um dos comandos de acesso aos dados, sendo eles, Insert, Update, Delete e Select. Para cada política de segurança criada, as seguintes informações

são registradas:

- **Tabela**: identificação da tabela no banco de dados em que a política de segurança será aplicada.
- **Tipo de política de restrição**: Refere-se ao comando SQL que será usado para acessar os dados, estes comandos são:

  SELECT: Utilizado para listar os dados de uma tabela do banco de dados.

  INSERT: Utilizado para inserção de um ou mais registros em uma tabela.

  UPDATE: Utilizado para efetuar a alteração de um ou mais registros armazenados em uma tabela.

  DELETE: Utilizado para a exclusão de um ou múltiplos registros do banco de dados.

  ALL: Refere-se a um comando generalista que indica a aplicação das políticas para cada uma das quatro possibilidades listadas anteriormente.
- **Criador da política de segurança**: usuário do banco de dados responsável pela criação da política de restrição.
- **Beneficiário**: Usuário do banco de dados para qual a política de segurança foi criada.
- **Política de segurança** : verificação que retorna se o usuário beneficiário tem acesso a um registro específico, sendo esta, uma verificação boleana, retornando assim verdadeiro ou falso.

É possível criar diversas políticas de segurança para a mesma tabela do banco de dados, porém, elas devem ser únicas. Ademais, ao receber uma requisição de consulta, todas as políticas criadas para uma determinada tabela serão verificadas.

2.4 Banco de dados

Para o melhor entendimento deste trabalho, torna-se interessante expor a descrição de conceitos e estruturas relacionadas diretamente ao sistema de gerenciamento do banco de dados e a aspectos que se referem a sua arquitetura geral, sendo eles:

- **Comando SQL** – Um comando SQL é uma instrução de texto usada para se comunicar com um SGBD com o objetivo de executar tarefas, funções, obter dados contidos em tabelas, criar ou modificar estruturas e dados no banco de dados.

- **Objeto** – Um objeto no contexto de banco de dados é um conceito que identifica estruturas como, tabelas, visões, clusters, *sequences* e índices, estes são utilizados para armazenar ou referenciar dados.

- **Usuário** – Um usuário é uma entidade fortemente relacionada a segurança em nível do banco de dados. Todos os acessos feitos devem ser mapeados e atrelados a um usuário de banco de dados, logo, para se conectar a um banco de dados é necessário informar o usuário que estará acessando naquele momento através de uma autenticação.

- **Esquema (Schema)** – Schemas são coleções de objetos que contribuem para uma organização melhor do banco de dados em varios aspectos, estes são importantes para segmentar a segurança e possibilitar um melhor gerenciamento dos objetos e dados contidos no banco.

- **Filtro** – Em banco de dados, um filtro pode ser definido como um recurso que permite buscar um intervalo de dados com base em critérios definidos manualmente, podendo utilizar operadores lógicos de comparação.

- **Predicado** - Um predicado pode ser descrito como um resultado da avaliação de uma expressão tendo como definição TRUE, FALSE ou UNKNOWN. Este é usado como critério de pesquisa de filtros aplicados a consultas e outras expressões onde seja necessária uma definição boleana

- **Perfis de Acesso** – Este é um conceito utilizado para definir um conjunto de permissões atribuídas a cada usuário, sendo possível agrupar usuários com as mesmas características de acesso aos dados;

- **Política de Segurança** – Uma política de segurança é um conceito que identifica um conjunto de padrões que será aplicado/associado a um objeto do banco de dados e a um perfil de acesso ou usuário afim de garantir a proteção das informações contidas em uma instância dos dados possibilitando uma melhor proteção contra possíveis ameaças que coloquem em risco a integridade do objeto.

2.5 Linguagem unificada UML

Na linguagem unificada UML (Unified Modeling Language), um diagrama nada mais é do que uma representação visual de elementos de um sistema. Eles são utilizados para visualizar o software sob diferentes perspectivas. Segundo a UML, os diagramas permitem observar de forma independente aspectos diversos de um mesmo sistema(BOOCH, 2006). O principal objetivo de um diagrama se encontra em facilitar a forma como a aplicação que está sendo desenvolvida é compreendida.

Um modelo entidade relacionamento (MER), é um modelo usado para descrever dados e aspectos da informação de uma maneira abstrata no que se refere a um determinado domínio e seus respectivos requisitos. Neste processo, esta representação descreve como as entidades pertencentes ao projeto relacionam-se entre si e pode ser implementado em diferentes sistemas gerenciadores de banco de dados (CHEN, 1976). No Modelo Entidade-Relacionamento, seus principais componentes são as entidades e suas relações.

O Diagrama Entidade-Relacionamento (DER), é um diagrama utilizado como representação gráfica do que foi descrito no MER. Ele se faz importante no que tange a descrição dos conceitos que serão utilizados para projetar o banco de dados(FRANCK; PEREIRA; FILHO, 2021).

Um Diagrama de fluxo é um modelo que utiliza um conjunto de símbolos para representar visualmente as etapas pertencentes a um determinado processo. Isto é, funciona analogamente a uma sequência de passos, descrevendo do início ao fim a execução de um determinado procedimento. Ele é responsável por mapear um fluxo de informações para que um processo seja executado(PRESSMAN, 2005).

2.6 Trabalhos relacionados

Sendo o controle de acesso granular aos dados uma técnica que possibilita a filtragem de informações através da reescrita de consultas ou aplicação de uma subconsulta junto a consulta original, é esperado que a aplicação deste método, possibilite que uma mesma operação solicitada ao SGBD por usuários distintos com perfis de acessos distintos tenha resultados diferentes.

A granularidade de acesso aos dados pode ser definida como uma modificação dinâmica de uma consulta com o objetivo de promover o controle de acesso aos objetos que estão associados a uma ou mais políticas de segurança no banco de dados (SPENDOLINI, 2013). Este método pode ser aplicado de várias maneiras diferentes, sendo possível elencar vantagens e desvantagens nelas.

O ideal é encontrar uma solução onde possa haver uma boa correlação entre manutenibilidade e escalabilidade do sistema utilizador de tal método, (CHAUDHURI; DUTTA; SUDARSHAN, 2007) mostram algumas desvantagens ao se utilizar a implementação de um controle de acesso granular através da aplicação, os principais malefícios de se utilizar este método, se dá pelo motivo de que geralmente o controle de acesso é aplicado através da implementação de diversas linhas de código, o que requer uma manutenção constante e possui o risco de ocasionar problemas de segurança causados por erro humano.

O primeiro trabalho relacionado ao contexto de controle de acesso granular dos dados explora a reescrita de uma consulta, (STONEBRAKER; WONG, 1974). A técnica

aplicada consiste em um algoritmo que modifica a consulta original reescrevendo-a para uma nova forma onde é aplicado restrições de controle de acesso para cada elemento advindo de uma interação entre usuário e os dados consultados. A consulta modificada por sua vêz recebe uma nova estrutura e ainda pode ser executada normalmente.

Posteriormente, diversas metodologias com o mesmo intuito foram criadas, sendo que as mais recentes também usam da reescrita de consultas como uma forma de aplicar o controle de acesso granular dos dados (MARQUES, 2013)

O estudo de (MARQUES, 2013) apresenta uma metodologia criada para garantir o controle de acesso aos dados de forma granular para sitemas gerenciadores de banco de dados *open-source*. O método consiste em efetuar uma interceptação dos comandos SQL enviados da aplicação para o banco de dados diretamente na camada de conexão com o banco de dados, após interceptar a consulta, analisa-se os privilégios de acesso e reescreve-se os comandos SQL enviando para o banco de dados a consulta reescrita.

O método apresentado não sofre alterações significativas de desempenho, entretanto possui algumas limitações, pois necessita de um ambiente específico para seu uso, sendo que sua aplicação é condicionada a utilização do mecanismo de conexão ao banco de dados denominado JDBC.

Em contraste dos exemplos vistos até então, atualmente, existem outros métodos para garantir o controle granular de acesso diretamente definidos no banco de dados. No trabalho de HOHENSTEIN, (HOHENSTEIN; ZILLNER; BIESDORF, 2018) é apresentada uma ferramenta para oferecer dados como um serviço, basicamente uma ferramenta para gerenciar o controle de acesso aos dados explicitados por API'S.

Tal ferramenta possibilita aos administradores entregarem os dados de maneira customizada para os consumidores, de acordo com a configuração desejada, possibilitando acessos a atributos específicos e linhas específicas dos dados contidos nos bancos de dados acessados pelas API's do *marketplace*. Esta abordagem faz o uso da segurança em nível de linha para tratar o acesso aos dados em vários níveis.

No modelo proposto por (ZHU et al., 2020) é apresentado como foi a utilização de Row Level Security, RBAC e LBAC (Label Based Access Control) para garantir o controle granular de acesso aos dados a uma plataforma digital privada, esta plataforma visa oferecer recursos para desenvolvimento de software, análise de dados, gerenciamento de tarefas para os desenvolvedores e outros serviços que viabilizam a criação e *deploy* de aplicativos, sendo então limitada a aplicabilidade destes métodos de controle de acesso apenas a estrutura já estabelecida desta plataforma, o qual inviabiliza a aplicação da solução criada para outras estruturas e aplicações.

A proposta do artigo é baseada em dividir o controle de acesso em duas camadas, na primeira restringe as ações que o usuário pode executar baseado em seu perfil de

acesso e na segunda camada cada consulta é analisada e filtrada com base em seus rótulos previamente configurados na aplicação.

Existe para o framework Ruby on Rails uma biblioteca que implementa o RLS, porém ela não trabalha de maneira semelhante à biblioteca proposta, pois trata a segurança de acesso aos dados diferentemente do modelo sugerido (SUUS, 2019). No modelo existente todas as políticas são definidas em um único arquivo que deve ser totalmente implementado de forma manual e não há integração com o RBAC ou outra ferramenta que provenha controle de acesso.

O presente trabalho busca oferecer um modelo para a aplicação dos conceitos de controle de acesso granular através da adição de subconsultas utilizando RBAC e RLS como principais recursos de controle de acesso, de forma que possibilite aplicar estes métodos de forma genérica a qualquer a bancos de dados que já esteja estruturado, não sendo limitado a um ambiente específico, sendo aplicável a outras linguagens e diferentes sistemas gerenciadores de bancos de dados que possuam o recurso de segurança em nível de linha.

3 Granar - Modelo para controle de acesso granular ao banco de dados

O objetivo deste capítulo é expor os detalhes referentes a metodologia estudada no desenvolvimento deste trabalho. Desse modo, serão expostos os passos necessários para a adoção do modelo proposto, bem como, sua exemplificação. O foco do método abordado se encontra na possibilidade de realizar o controle ao acesso granular aos dados. Ademais, objetiva-se que sua aplicação seja compatível a qualquer sistema gerenciador de banco de dados que contemple o recurso de segurança em nível de linha.

Este modelo define uma estrutura utilizada para implementar políticas de segurança atreladas diretamente a objetos do banco de dados, bem como, possibilita a existência de regras de acesso utilizadas para delimitar a porção de informações que estarão disponíveis a usuários específicos. Ademais, tal procedimento contribui para a transparência e visibilidade organizacional no que tange o acesso aos dados, uma vez que todos os procedimentos, regras e responsáveis por determinadas instâncias serão explicitamente definidos e facilmente gerenciados.

Sendo assim, este fluxo permite que uma aplicação limite o acesso às informações através da inserção de políticas de restrição no banco de dados, sendo este, o responsável por filtrar os dados que serão retornados a aplicação. Assim, o código SQL denominado "Política de Segurança Row Level Security"é gerado pelo programador/administrador da aplicação a partir da inserção dos parâmetros desejados. Este código habilita a aplicação de filtros customizados às consultas realizadas ao banco de dados. Enfim, tal processo se dá através da união dos recursos disponibilizados pelo Role Based Acess Control e do Row Level Security, de forma a potencializar seus usos individuais.

O fluxo de funcionamento adotado nesta metodologia é exposto a seguir:

- Geração das políticas de restrição através da aplicação
- Inserção dos scripts SQL gerados no banco de dados
- Fluxo de consulta: ao ocorrer uma solicitação de leitura ou escrita ao banco de dados, o contexto da aplicação é considerado e ao executar a consulta, um filtro (política de segurança) é verificado. Assim, caso o usuário em questão tenha permissão para realizar a ação, ela é concluída em sua totalidade. Contudo, se autorização não existir, o banco de dados realizará a interceptação e só realizará as alterações ou retorná os dados de acordo com o nível de acesso determinado pelo administrador do sistema.

3.1 Arquitetura do modelo Granar

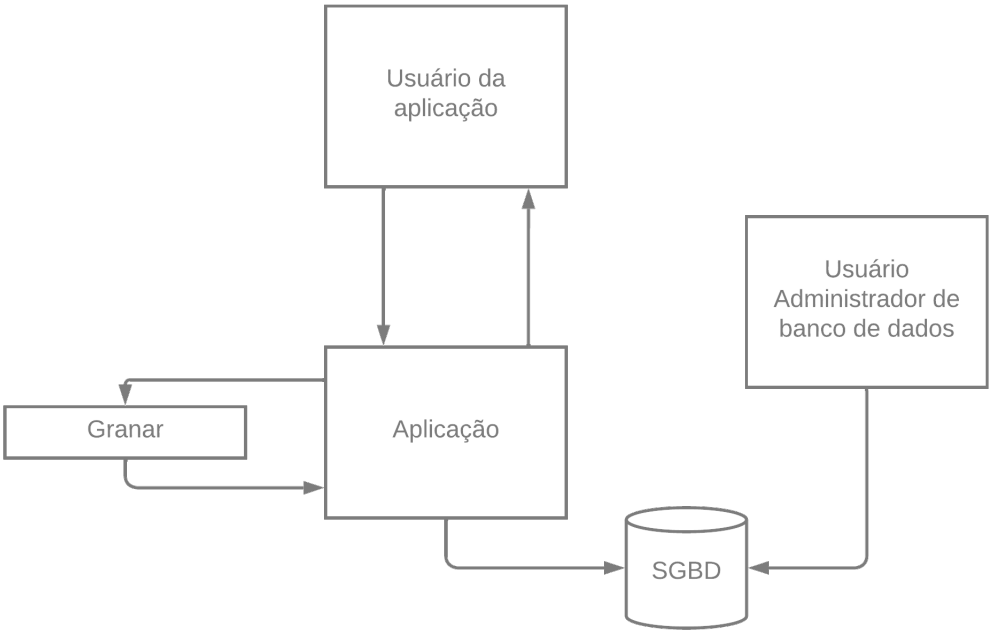
A arquitetura deste modelo detalha e delimita a forma como são organizadas as estruturas de um projeto seguindo tal abordagem. Sendo assim, é importante considerar que cada passo neste processo influencia aspectos como a performance, qualidade, facilidade de manutenção e escalabilidade do modelo. Enfim, este fluxo possibilita a replicação do sistema desenvolvido usando os conceitos abordados anteriormente.

A figura 5 expõe a arquitetura adotada por este modelo, a partir da representação abstraida, é possivel inferir as interações de cada componente nas interações entre aplicação e sistema gerenciador de banco de dados.

Apartir desta representação é possível simplificar a compreensão das relações dos usuários existentes no modelo ao acessar os dados contidos no SGBD. Já Granar é representada como uma camada complementar a aplicação, não interferindo diretamente nas consultas enviadas para o SGBD, sendo então o objetivo de Granar prover recursos extras para a aplicação de forma que seja possivel simplificar a criação de rotinas que moldam o comportamentos padrão do SGBD.

Por fim, nesta imagem é demonstrado uma abstração de como o sistema funciona no que tange as interações entre cada componente especifico. Esta figura é baseada nos conceitos supracitados nos capítulos 2.2 e 2.3 e será detalhada a seguir.

Figura 5 – Arquitetura base do Modelo Granar
Fonte: Elaborado pelo autor (2022)



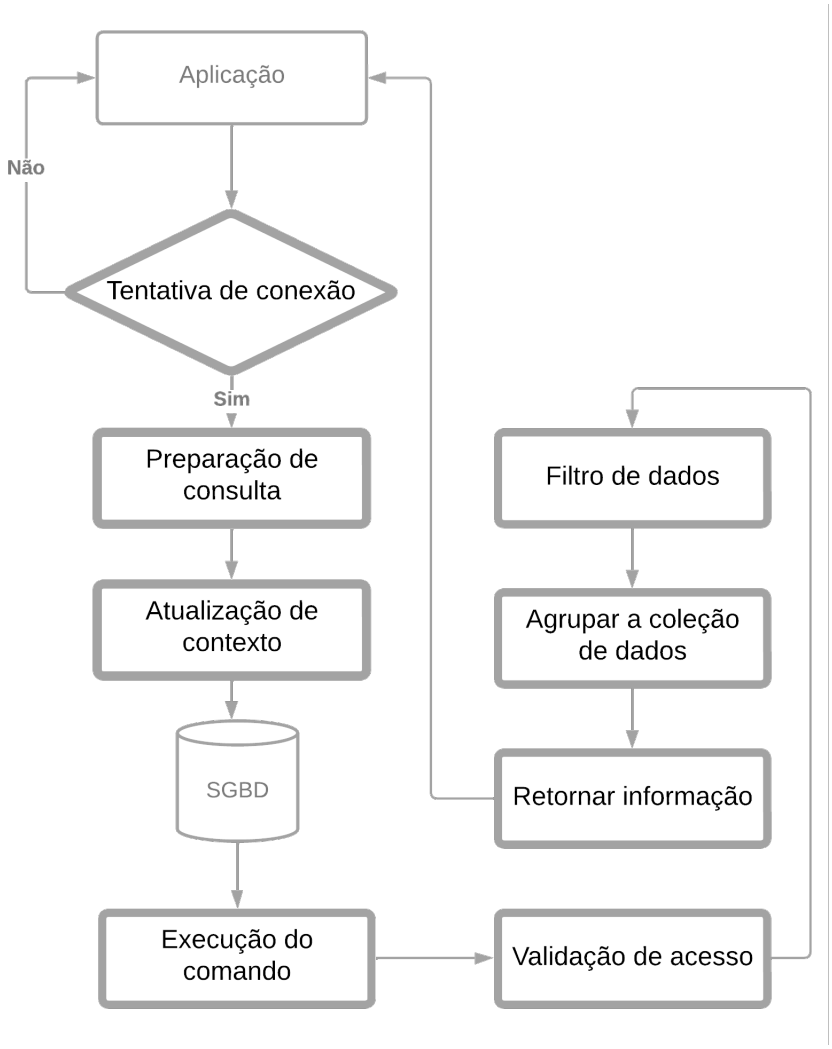
3.1.1 Usuários

Neste modelo é papel da aplicação possibilitar ao programador a estrutura necessária para gerar código através da passagem de parâmetros. Sendo assim, a partir da implantação das restrições de acesso, será informado ao banco de dados o contexto necessário para que ocorra a limitação de acesso, seja para casos de leitura ou escrita de dados.

Cabe apenas aos usuários administradores do banco de dados a função de aplicar as políticas de segurança geradas pela aplicação. Diferentemente da aplicação, este tem acesso total a todas as instâncias do banco de dados e é responsável por gerir a estrutura e organização do mesmo.

3.1.2 Fluxo adotado pelo Modelo

Figura 6 – Modelo de fluxo da interação entre aplicação e banco de dados
Fonte: Elaborado pelo autor (2022)



Após a aplicação das políticas de segurança definidas pelo administrador do sistema, o fluxo adotado pelo modelo é semelhante ao exposto a seguir quando houver operações

de escrita ou leitura executadas pelo sistema gerenciador de banco de dados:

- **Tentativa de conexão:**

  Usualmente, quando existe interação entre a aplicação e o banco de dados, é necessário que a aplicação antes de realizar qualquer operação inicie uma conexão específica com o banco de dados. Neste processo, ela envia suas credenciais de acesso únicas para que ocorra a validação por parte do banco de dados, caso a conexão seja recusada, um erro é retornado. Se a solicitação for aceita, uma conexão direta é estabelecida entre os dois lados.

- **Preparação de consulta:**

  Após confirmada a existência de uma conexão entre a aplicação e o banco de dados, a aplicação será responsável por converter os parâmetros enviados pelo cliente em um script SQL que fará a operação de leitura ou escrita na base de dados.

- **Atualização de contexto:**

  O processo de atualização de contexto consiste em permitir que a aplicação comunique ao banco de dados informações importantes do contexto em que se encontra. Estas informações geralmente são identificadores de seus usuários, bem como, dados adicionais considerados opcionais que serão utilizados no momento em que a execução dos comandos SQL ocorrerem. Esta etapa deve necessariamente ocorrer antes da execução do comando de leitura ou escrita no banco de dados.

- **Execução do comando:**

  Esta etapa se inicia com o envio de um comando SQL oriundo da aplicação sem quaisquer modificações, o qual é interpretado pelo SGBD em etapas posteriores. Antes da execução ocorrer, é verificada a estrutura do script SQL, caso a consulta esteja livre de erros, o fluxo prossegue para a próxima etapa.

  Sempre que uma operação é submetida ao SGBD, é necessário que haja uma tradução e uma transformação dos comandos para uma sequência de operações que possa ser executada pelo sistema computacional. Após estes processos, a etapa de processamento da consulta é iniciada, esta consiste em extrair, alterar ou inserir informações de um banco de dados.

- **Validação de acesso:**

  Na etapa de execução, para cada política de segurança aplicada ao objeto do banco de dados é feita uma validação de acesso. Tal processo consiste em analisar todas as cláusulas de todas as políticas de segurança aplicadas ao objeto específico até que se encontre uma válida.

Nesta etapa, para cada componente da consulta é retornado um valor de verdadeiro ou falso. Neste contexto, verdadeiro significa que o usuário tem acesso àquele registro e falso representando que ele não tem direitos de acesso àquele registro.

- **Filtro de dados:**

  A filtragem de dados consiste em separar os dados em duas categorias, sendo elas, verdadeiro para as informações onde o usuário possui direito de acesso e falso para os casos onde não existe a permissão.

- **Agrupar a coleção de dados:**

  Após a conclusão das etapas anteriores, é necessário que o banco de dados separe as informações que serão retornadas para a aplicação daquelas que não serão. Neste passo, uma tupla com as informações validadas e marcadas como verdadeiro para direito de acesso é formada. Já as informações marcadas como falso são descartadas, pois o usuário não possui acesso.

- **Retornar informação:**

  Após a realização do agrupamento de toda a coleção de dados, o banco de dados envia o resultado da consulta realizada para a aplicação.

Todas as etapas explicadas são aplicáveis a operações de leitura ou escrita. Por fim, cabe a aplicação confirmar a finalização da operação solicitada ao banco de dados, encerrar a conexão estabelecida e enviar os dados recebidos na consulta para o cliente da aplicação. Ademais, é válido ressaltar que as etapas de validação de acesso e filtragem dos dados ocorrem em conjunto, embora possuam papéis distintos.

3.1.3 Detalhamento das etapas de criação de uma política de segurança

Um dos objetivos deste trabalho é permitir que exista um desacoplamento do banco de dados à aplicação, possibilitando assim que o mesmo realize o controle de acesso separadamente da aplicação.

Para atingir tal objetivo, é necessário que toda a estrutura do Role Based Acess Control seja aplicada diretamente no banco de dados. Desse modo, esta metodologia implementa essa estrutura através da criação de tabelas para o controle, identificação e armazenamento das informações relativas a usuários, grupos, permissões e funções dos usuários.

Estas informações serão utilizadas nas etapas de criação de políticas de segurança e também nas etapas descritas na seção 3.1.2, sendo então uma estrutura fundamental para uma atuação efetiva do modelo.

O processo de geração de uma nova política de segurança ocorre através dos seguintes passos:

- Identificação dos usuários da aplicação associados a função
- Identificação da função ou grupo que será associado a uma permissão
- Identificação do tipo de permissão (básica ou acesso individual)

Ademais, é necessário que estas informações estejam persistidas no banco de dados antes da etapa de geração de código ocorrer. Sendo assim, cabe à aplicação possibilitar meios simples pelos quais será possível persistir cada uma destas informações.

3.1.3.1 Criação de papeis

A etapa de criação dos papéis tem por objetivo a geração de perfis de acesso para que seja possível que o usuário utilizador crie as funções a partir da hierarquia organizacional experimentada por ele.

Deste modo, cada papel criado poderá ser associado a múltiplos usuários, algo análogo ao que é representado na figura 4. Permitindo, como é sugerido pelo RBAC, que haja uma hierarquia entre os perfis de acesso e seja permitido a associação de uma ou várias permissões a um perfil específico, isto é, o que cada membro de um determinado grupo poderá ou não acessar.

3.1.3.2 Definição de Politicas de segurança

Na etapa de definição de políticas de segurança, cada usuário que possui acesso à aplicação será identificado e cada consulta feita ao banco de dados retornará apenas as informações ao qual o mesmo possuir o direito de acesso. Todas as restrições serão feitas com base no atributo identificador, determinando um tratamento específico para cada usuário com base em suas funções e permissões definidas.

Como visto na seção 3.1.3.1, toda função criada poderá ter múltiplas permissões de acesso associadas a ela, logo, cada permissão definida na aplicação será atendida como uma política de segurança do Row Level Security que fará o controle de acesso por parte do banco de dados. Até o momento foram elencadas três formas de restringir os dados, sendo elas, permissões básicas, permissões de acesso individual e permissões de acesso em grupo.

3.1.3.2.1 Permissões básicas

Como descrito na seção 2.3.1, cada política de segurança está relacionada a uma única tabela, logo, as permissões básicas procuram controlar o acesso para as interações do sistema de forma específica.

Ao criar uma nova permissão, o usuário criador deverá especificar quais tipos de interações poderão ser realizadas pelos papéis associados a ela, estas restrições básicas compreendem leitura, escrita, atualização ou exclusão de informação dos dados contidos na tabela do banco de dados.

Um exemplo desta situação é perceptível quando um usuário possui permissão apenas de leitura e gravação de dados para um papel específico, não permitindo atualização ou exclusão.

3.1.3.2.2 Permissões de acesso individual

Com o objetivo de fornecer um controle de acesso mais específico e modular, o modelo conta com permissões de acesso individual a uma informação. Sendo assim, é possível que ao criar uma permissão seja aplicada uma restrição para que apenas quem criou o registro tenha acesso de visualizar, atualizar, ou apagar o mesmo. Este controle é feito utilizando um campo previamente existente na estrutura da tabela.

3.1.3.2.3 Permissões de acesso em grupo

As permissões de acesso em grupo objetivam fornecer um controle de acesso direcionado a um conjunto de indivíduos pertencentes a um mesmo nível hierárquico de acesso, diferentemente de outras permissões, nesta os indivíduos podem ter papéis distintos no sistema e pertencerem a um mesmo grupo de acesso.

Estas permissões visam restringir o acesso acerca de visualização, atualização, exclusão ou até mesmo inserção de novas informações no banco de dados. Este tipo de permissão tem como finalidade aferir ou negar o acesso um recurso para todos os membros, sendo então uma forma mais generalista de tratar os acessos dentro de uma hierarquia, isso possibilita que haja acesso de diferentes indivíduos com papéis distintos a um recurso comum, desta forma as permissões de acesso em grupo centralizam o gerenciamento de acesso.

3.1.4 Geração de código automatico

A partir da implementação do modelo, será possível gerar o código necessário de forma automática, em suma, esta etapa é responsável por gerar códigos de comandos SQL

específicos de RLS para o banco de dados. O objetivo desta fase é facilitar o desenvolvimento das políticas, bem como diminuir a quantidade de código escrito pelos programadores.

O gerador recebe como entrada parâmetros que serão utilizados para a geração de código e execução dos seguintes passos no banco de dados:

- **Conceder privilégios:** Criar um comando para autorizar o controle de acesso aos dados para um determinado usuário do banco de dados, sendo que este segue a estrutura exemplificada na figura 7:

Figura 7 – Exemplo de comando para conceder privilégios
Fonte: Elaborado pelo autor (2022)

```
GRANT [access type]
ON [object name]
TO [user]
```

- **Habilitar o Row Level Security:** Criação de um comando que habilita o RLS para uma tabela específica, sendo que ao habilitá-lo, a tabela passará a controlar o acesso baseada nas políticas definidas para a mesma. O comando gerado segue a estrutura demonstrada na figura 8:

Figura 8 – Exemplo de comando para habilitar o Row-level security
Fonte: Elaborado pelo autor (2022)

```
ALTER TABLE [table name]
ENABLE ROW LEVEL SECURITY
```

- **Criar uma política especifica - Row-level security:** Criação da política restritiva, comando que pode assumir a seguinte estrutura demonstrada na figura 9 :

Figura 9 – Exemplo de criação de uma política do Row-level security
Fonte: Elaborado pelo autor (2022)

```
CREATE POLICY [policy name]
ON [table name] for [policy type]
TO [user] [validation type ] [boolean validation]
```

3.1.4.1 Algoritmo para geração de políticas do Row Level Security

A implementação das etapas de geração de código descritas anteriormente, podem ser ilustradas a partir do algoritmo 1. Ademais, após sua execução, não se faz necessário a realização de nenhum passo adicional para inclusão das modificações no banco de dados.

Algorithm 1 Algoritmo de geração de política RLS

```
procedure RlsPolicyGenerator(typesPolicies, user,policyName,table)
    for type in typesPolicies do
        enableSql ← GenerateEnableRlsSql(table)
        grantSql ← GenerateGrantAccessSql(type, table, user)
        policySql ← GenerateRlsPolicySql(type, user, policyName, table)
        ExecuteSqlInDataBase(enableSql)
        ExecuteSqlInDataBase(grantSql)
        ExecuteSqlInDataBase(policySql)
    end for
end procedure
procedure ExecuteSqlInDataBase(sql)
    connection ← DBConnection()
    if connection == True then
        Execute(sql)
    end if
end procedure
```

3.2 Considerações Finais

Neste capitulo foi descrito a metodologia proposta e suas etapas necessárias para garantir o controle de acesso granular em SGBDs relacionais utilizando RBAC e RLS. No capítulo 4 é descrito o desenvolvimento de uma ferramenta seguindo tal estrutura. Esta será utilizada como prova de conceito para avaliar a metodologia criada.

É possível concluir que um dos pontos positivos ao utilizar este modelo é que o mesmo possibilita a execução de todas as verificações de acesso à informação diretamente ao nível de banco de dados, impedindo que dados que serão descartados cheguem à aplicação ou que seja extraído indevidamente informação do banco de dados.

Contudo, aumentar a responsabilidade do banco de dados acerca do controle de acesso às informações acarreta uma maior responsabilidade no que tange sua manutenção, uma vez que sua estrutura torna-se consideravelmente mais complexa.

4 Granar

Neste capítulo serão descritos os aspectos técnicos acerca da ferramenta baseada no modelo proposto, nomeada como Granar. Serão apresentadas as especificações conceituais, diagramas, descrição de toda a arquitetura e tecnologias adotadas para a construção da ferramenta. Ademais, existem sessões específicas para exemplificar o uso prático da mesma, realizar a análise dos benefícios e possíveis limitações. Por fim, uma análise de desempenho e dos testes realizados com usuários será exposta.

4.1 Descrição geral da ferramenta Granar

A fim de validar a metodologia proposta, construiu-se uma ferramenta que tem como objetivo controlar o acesso granular aos dados em bancos de dados. Esta funcionalidade se torna possível a partir da aplicação de políticas de segurança. Tal processo advém da utilização de recursos de segurança em nível de linha gerados pela ferramenta via linha de comando ou interface gráfica integrada a aplicação principal.

A ferramenta Granar entrega ao programador recursos que possibilitam que o tratamento do controle de acesso seja realizado diretamente a nível de banco de dados, ou seja, ele se torna responsável por limitar a alteração e consulta de suas informações armazenadas.

Assim, a implementação deste modelo permite que haja uma redução significativa na quantidade de código na aplicação, isso se torna possível, pois toda a tratativa de controle de acesso será migrada para o banco de dados através da inserção de scripts gerados pela aplicação através de parâmetros selecionados pelo programador. Sendo assim, não existe a necessidade de se adicionar código extra para realizar tal procedimento à rotina principal do sistema.

A seguir realiza-se uma comparação acerca de diferentes maneiras de implementar o controle de acesso, sendo elas, via aplicação e utilizando o modelo Granar, ou seja, via banco de dados.

4.2 Formas de controle de acesso

O controle de acesso pode ser implementado a partir de diversas estruturas. A seguir serão abordadas duas maneiras, sendo elas, via aplicação, utilizando a inserção de código extra para realizar as verificações e contingências referentes ao controle as informações. Ademais, via banco de dados, utilizando a definição de políticas de segurança

na aplicação e geração dos scripts SQL necessários para realizar tal procedimento através da metologia Granar.

Com o intuito de melhorar o entendimento a respeito da forma como a Granar possibilita o controle de acesso, um exemplo teórico será detalhado a seguir, destrinchando como seria a realização do controle de acesso para o mesmo cenário utilizando as duas maneiras citadas anteriormente.

4.2.1 Controle de acesso via aplicação

Considerando um cenário onde o usuário solicita à aplicação que a tabela denominada "Curso"tenha um de seus registros alterado, a mesma deverá realizar a verificação de três possíveis possibilidades, sendo elas, o usuário pode possuir a função de chefe de departamento, professor ou pró-reitor. Assim, para cada caso, a aplicação terá que realizar uma sequência de rotinas e consultas ao banco de dados adequada ao contexto em questão.

Para os três casos de exemplo, nota-se que é necessário realizar no mínimo duas consultas, buscando informações mais detalhadas acerca do usuário, bem como, do registro que será atualizado. Nesse sentido, as diferenças de verificações são notadas a partir das verificações de direito de acesso adicionais.

- **Chefe de departamento:**

Figura 10 – Código de controle de acesso feito via aplicação para modificação de dados em uma tabela para o cargo de chefe de departamento
Fonte: Elaborado pelo autor (2022)

```
atualiza_tabela_curso("Ciência Da Computação", novos_atributos)

def atualiza_tabela_curso(nome_do_curso, novos_atributos)
  #Busca Informações no Banco sobre o curso
  curso = Course.find_by_name(nome_do_curso)

  #Busca Informações no Banco sobre o usuário
  usuario = User.find_by_id(sessao[:id_usuario_autenticado])
  papel_do_usuario = usuario.find_user_role
  #verifica se o usuário autenticado é um chefe de departamento
  if papel_do_usuario == "Chefe de departamento"
    #verifica se o campus onde o usuário é chefe é o mesmo
    # campus do curso a ser atualizado
    if curso.nome_campus == usuario.departamento.nome_campus
      curso.update(novos_atributos)
    end
  end
end
```

O primeiro caso, exposto na figura 10, exprime a forma como deve ser feita a verificação via aplicação para garantir que apenas o chefe de departamento do campus vinculado ao curso de uma instituição consiga realizar a alteração de informações do curso em questão. Isto é, além das consultas básicas citadas anteriormente. Faz-se necessário verificar se o papel do usuário logado condiz com a solicitação, bem como, se o curso em questão faz parte do campus em que o usuário logado é chefe de departamento.

- **Professor:**

Figura 11 – Código de controle de acesso feito via aplicação para modificação de dados em uma tabela para o cargo de professor
Fonte: Elaborado pelo autor (2022)

```ruby
atualiza_tabela_curso("Ciência Da Computação", novos_atributos)

def atualiza_tabela_curso(nome_do_curso, novos_atributos)
  #Busca Informações no Banco sobre o curso
  curso = Course.find_by_name(nome_do_curso)

  #Busca Informações no Banco sobre o usuário
  usuario = User.find_by_id(sessao[:id_usuario_autenticado])
  papel_do_usuario = usuario.find_user_role
  if papel_do_usuario == "Professor"
    #Busca Informações Quem são os professores do curso
    professores_do_curso = curso.professors_ids

    #verifica se o usuário autenticado é professor do curso
    if professores_do_curso.include?(usuario.id)
      curso.update(novos_atributos)
    end
  end
end
```

No caso exposto na figura 11, a forma como deve ser feita a verificação via aplicação para garantir que apenas os professores vinculados ao curso de uma instituição consiga realizar a alteração de informações deste curso é exposta. Portanto, além das consultas acerca das informações do curso e do próprio usuário, a aplicação também verifica o papel do usuário logado e se o mesmo faz parte dos professores do curso em questão.

- **Pró-reitor:**

Para este caso, pouca validação é necessária, tendo em vista que o papel de pró-reitor é privilegiado, como mostrado na figura 12. Há apenas uma validação necessária,

Figura 12 – Código de controle de acesso feito via aplicação para modificação de dados em uma tabela para o cargo de pró-reitor
Fonte: Elaborado pelo autor (2022)

```
atualiza_tabela_curso("Ciência Da Computação", novos_atributos)

def atualiza_tabela_curso(nome_do_curso, novos_atributos)
  #Busca Informações no Banco sobre o curso
  curso = Course.find_by_name(nome_do_curso)

  #Busca Informações no Banco sobre o usuário
  usuario = User.find_by_id(sessao[:id_usuario_autenticado])
  papel_do_usuario = usuario.find_user_role

  #verifica se o usuário autenticado é um chefe de departamento
  if papel_do_usuario == "Proreitor"
    curso.update(novos_atributos)
  end
end
```

isto é, verificar se o usuário logado é o pro-reitor da instituição e então é aferido o direito de atualizar as informações.

4.2.2 Controle de acesso via Banco de dados

Considerando o mesmo cenário utilizado anteriormente, na sessão 4.2.1, ao utilizar a metodologia Granar não se faz necessário inserir qualquer código extra ao já existente na aplicação para realizar o controle de acesso aos dados. Isto ocorre pois toda a tratativa será feita pelo próprio banco de dados, através do cadastro de papéis e permissões para geração de políticas de segurança. Este procedimento é abordado em detalhes na sessão 4.3.3.

A figura 13, exemplifica uma política de restrição gerada pela ferramenta Granar. Esta, é capaz de realizar a mesma função executada pelos três exemplos demonstrados na sessão anteriormente, ela utiliza-se de uma sub consulta feita ao banco de dados para verificar se é possível realizar a operação solicitada pela aplicação.

4.3 Arquitetura do Sistema

Esta seção tem por objetivo fornecer a descrição de alto nível do software, explicando sua estrutura e comportamento.

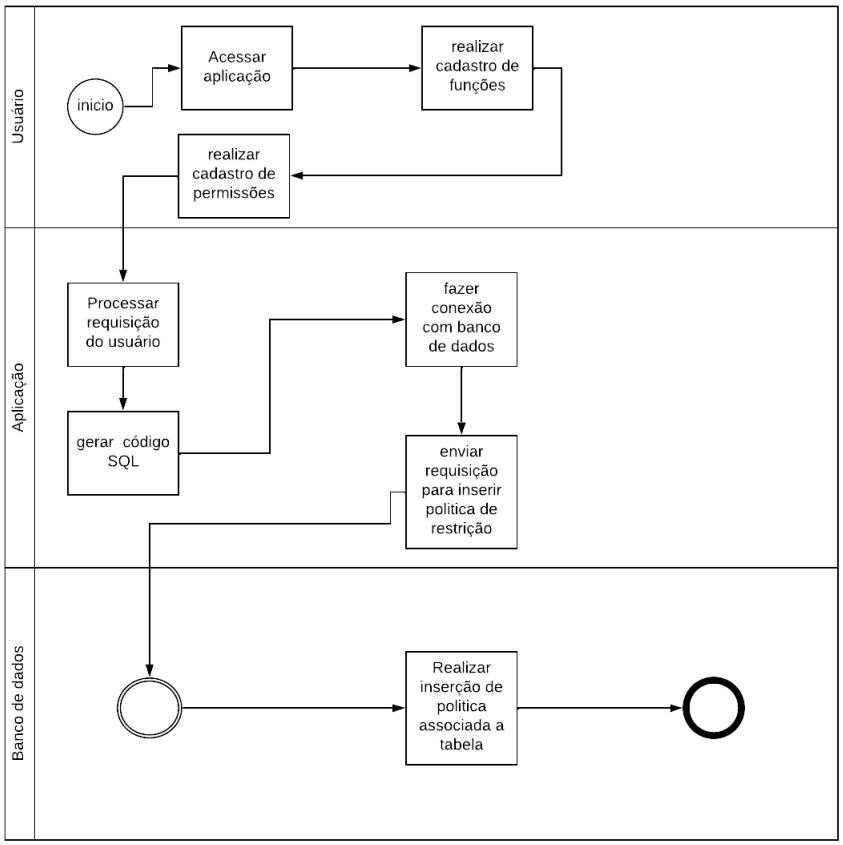
O fluxo básico de funcionamento do framework Granar é representado na figura 14, este, consiste na geração de código por parte da aplicação após o recebimento dos dados

Figura 13 – Política de segurança para controle de acesso via banco de dados para modificação de dados em uma tabela
Fonte: Elaborado pelo autor (2022)

```
CREATE POLICY politica_update_curso ON curso for  select TO app_user using (
            NULLIF(current_setting('rls.user_id', TRUE), ''):bigint in
       (SELECT
           ur.user_id
         FROM permissions p
           INNER JOIN role_permissions rp  on rp.permission_id  = p.id
             INNER JOIN user_roles ur  on rp.role_id  = ur.role_id
         WHERE (p."change")
           AND p.table_name = 'curso')

 or
         (NULLIF(current_setting('rls.user_id', TRUE), ''):bigint in
   (
     SELECT
       ur.user_id
     FROM permissions p
       INNER JOIN role_permissions rp  on rp.permission_id  = p.id
         INNER JOIN user_roles ur  on rp.role_id  = ur.role_id
     WHERE (p."owner_change")
       AND p.table_name = 'curso'
   )  and  owner_id = NULLIF(current_setting('rls.user_id', TRUE), ''):bigint  ))
```

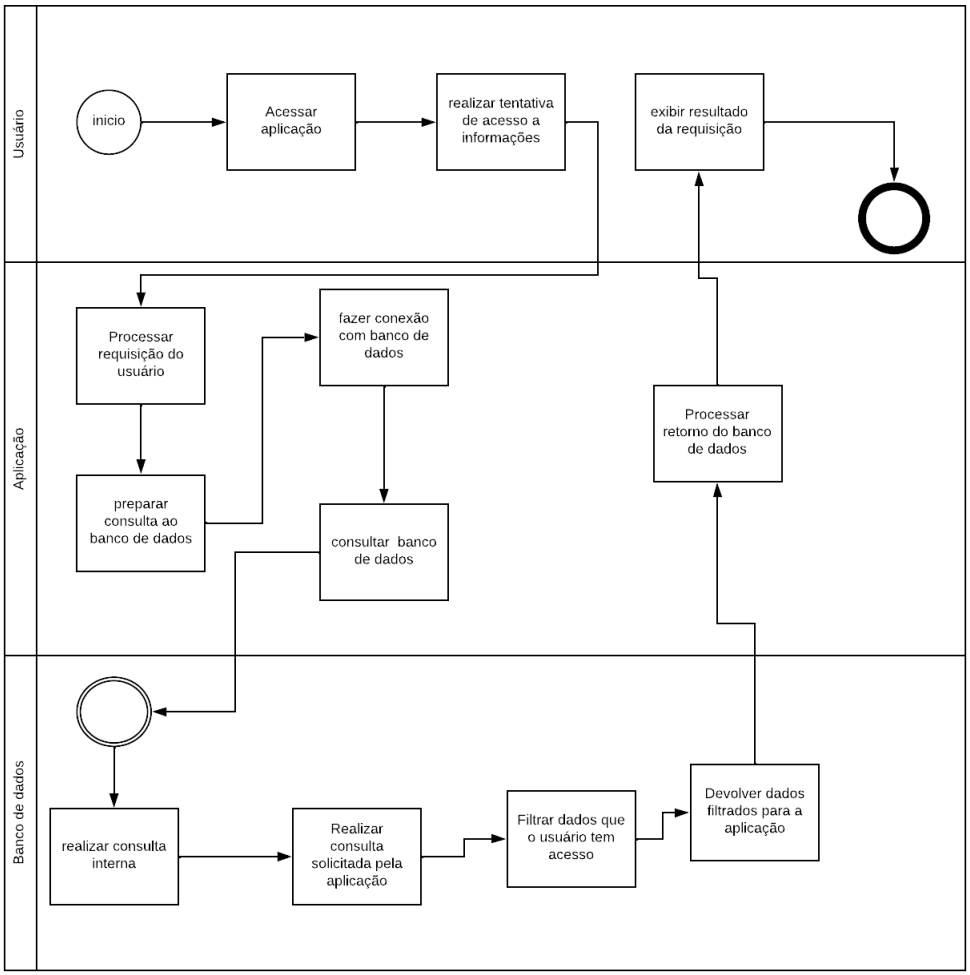
Figura 14 – Fluxo de para inserção de novas políticas de restrição
Fonte: Elaborado pelo autor (2022)



enviados pelo usuário e por fim, a conexão com o banco de dados, onde ocorre a inserção da nova política criada associando-a a uma tabela.

Sendo assim, para cada objeto do banco de dados, beneficiado por uma política de segurança em nível de linha, as informações serão filtradas a cada tentativa de acesso, como exemplificado no fluxo de consulta da figura 15.

Figura 15 – Fluxo de consulta em objetos beneficiados pelo RLS
Fonte: Elaborado pelo autor (2022)

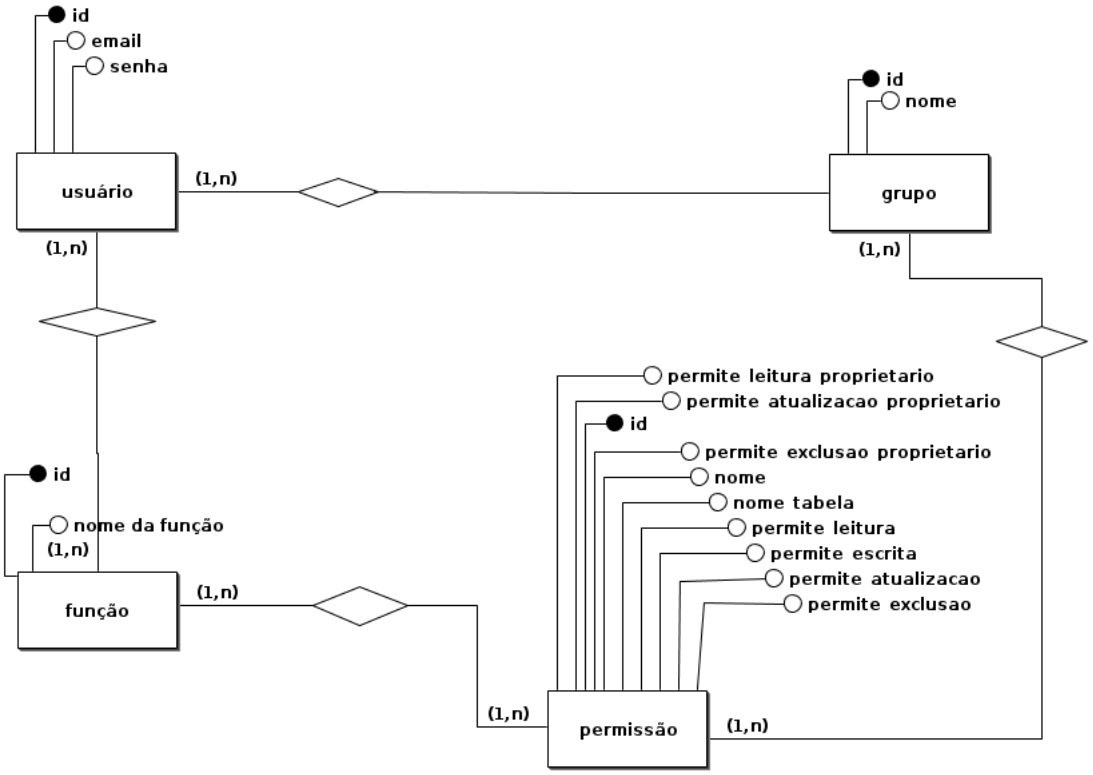


4.3.1 Diagramas

4.3.1.1 Modelo Entidade Relacionamento

A figura 16 representa especificamente a etapa de estruturação do banco de dados de forma conceitual. Esta representação é de suma importância, pois exemplifica sua estrutura e entidades. Seu objetivo é representar como estes conceitos se relacionam, bem como, definir o planejamento dos objetos que irão compor o banco de dados.

Figura 16 – Modelo Entidade relacionamento
Fonte: Elaborado pelo autor (2022)



4.3.1.2 Diagrama Entidade Relacionamento

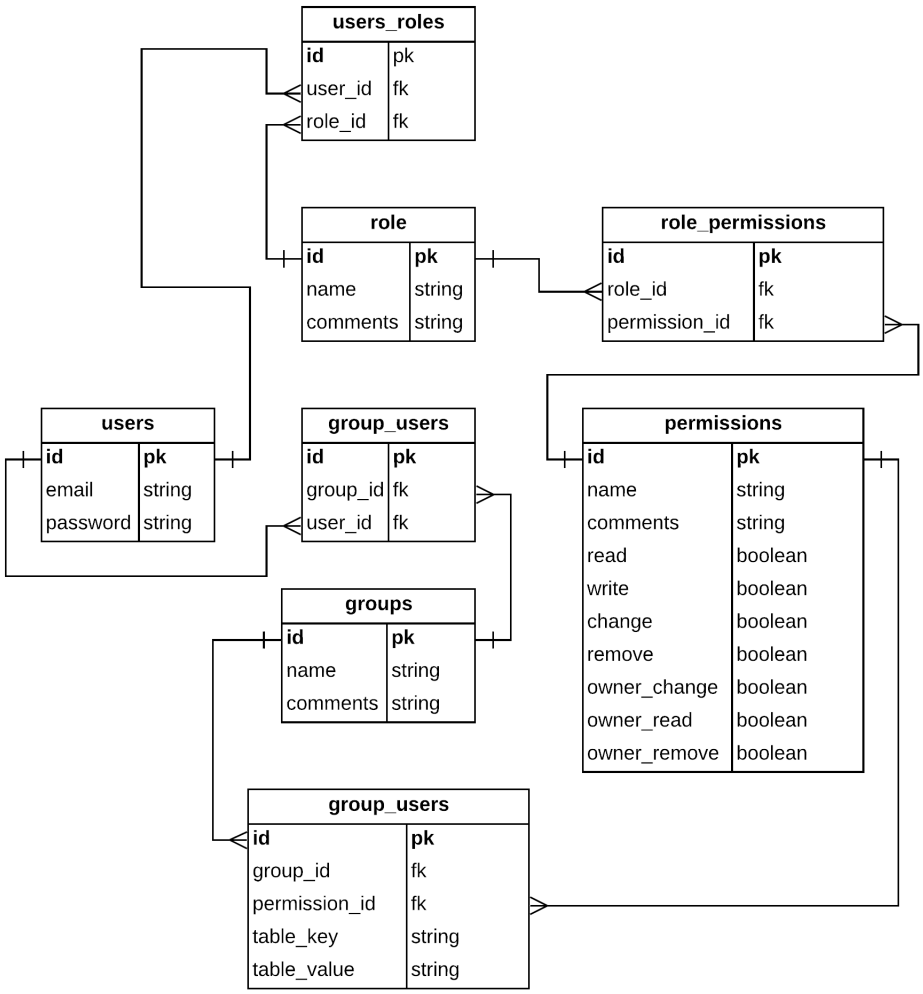
A figura 17 representa a notação associada a figura 16, sendo então a estrutura física básica das tabelas necessárias para a criação da ferramenta Granar.

4.3.1.3 Diagrama de Fluxo

A figura 4.3.1.3 representa o fluxo de utilização da ferramenta Granar para criar uma nova política de segurança, a seguir, suas etapas são detalhadas:

- **Criar Papeis/Criar Grupos:** Esta etapa consiste em informar identificadores e informações sobre o novo papel que está sendo criado, sendo então futuramente atribuído a uma posição hierárquica específica no sistema.
- **Cadastrar Permissões:** Uma permissão detalha quais são as formas permitidas para se manipular as informações para um dado objeto selecionado, ou seja, como cada usuário associado a esta permissão através de seus papeis podem manipular o conteúdo da tabela.
- **Associar permissões a papéis:** Esta etapa consiste em criar um registro associativo permitindo que haja uma relação entre os papéis e as permissões.

Figura 17 – Diagrama Entidade Relacionamento
Fonte: Elaborado pelo autor (2022)



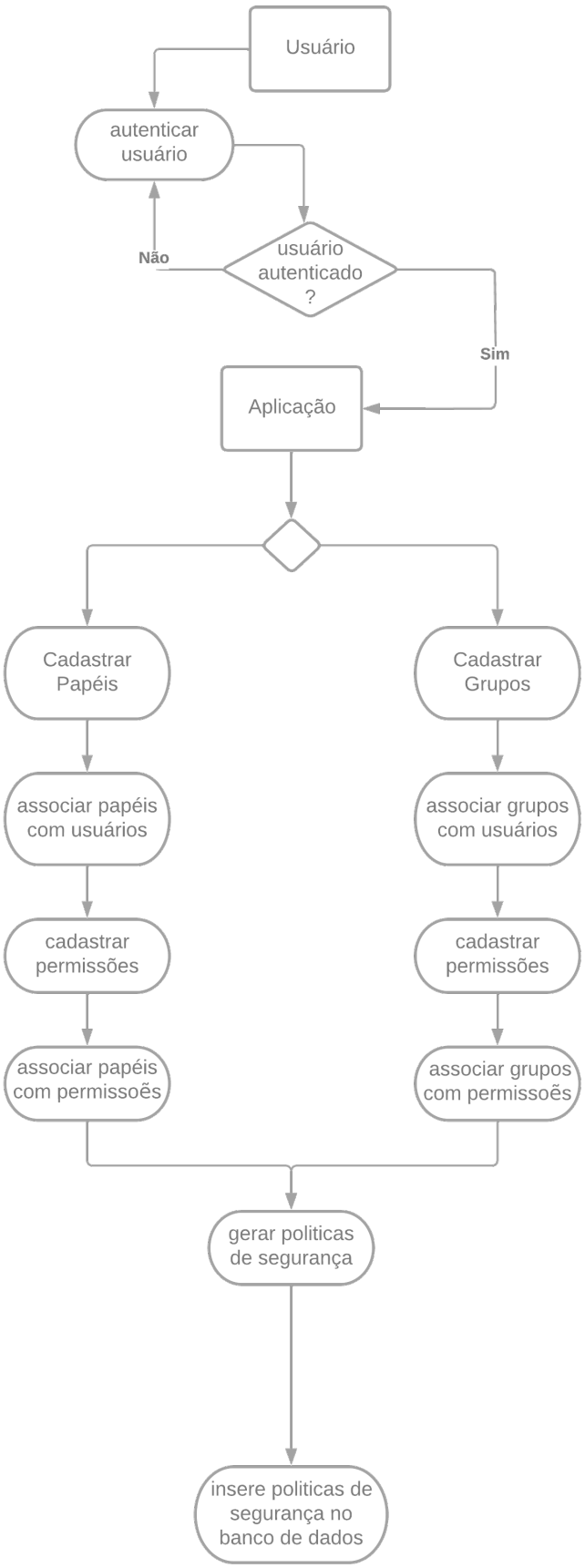
- **Associar usuários a papéis:** Esta etapa consistem em criar um registro associativo permitindo que haja uma relação entre os papéis e os usuários.

Após a realização das etapas exemplificadas anteriormente, a ferramente utiliza-se destas informações para gerar código, originando assim, uma nova política de restrição para a tabela selecionada no cadastro.

4.3.2 Tecnologias Utilizadas

Para a execução de etapas posteriores foi necessário utilizar-se de recursos providos por sistemas gerenciadores de bancos de dados e linguagens de programação de alto nível, o objetivo desta sessão é prover uma descrição das ferramentas e quais seus papéis na associação com os conceitos teóricos.

Figura 18 – Representação do fluxo de criação de políticas de segurança
Fonte: Elaborado pelo autor (2022)



4.3.2.1 **Ruby**

Ruby é uma linguagem de programação **interpretada**, multiparadigma, de tipagem dinâmica e forte, que conta com o gerenciamento automático de memória. Este trabalho utiliza a versão 2.7.4 da linguagem.

A ferramenta Granar foi desenvolvida com o foco em aplicações web, dessa forma, Ruby foi utilizada para a construção de todo o back-end, contando com recursos do framework Rails para o auxílio e estruturação do projeto, bem como, a construção da interface gráfica da biblioteca.

4.3.2.2 **PostgreSQL**

PostgreSQL é um sistema gerenciador de banco de dados (SGBD), desenvolvido originalmente como um projeto de código aberto, sendo então a ferramenta que atua como sistema de gerenciamento de bancos de dados deste trabalho, implementando toda a estrutura descrita na sessão 4.3.1.2. PostgreSQL permite a execução dos comandos SQL que serão amplamente explorados por este trabalho.

4.3.3 Fluxo de utilização da ferramenta

Com o intuito de melhorar a experiência de usuário no que tange a utilização do framework, uma interface gráfica é fornecida, ela possui a função de controlar todo o ciclo de vida de políticas do RLS e também dos papéis do RBAC, compreendendo criação, alteração e exclusão das mesmas. Sendo assim, o usuário administrador da aplicação que consumirá a biblioteca poderá acessar uma tela simples e intuitiva para fazer o gerenciamento das funções e permissões que irão gerar código de maneira, automática. Desse modo, o principal objetivo da interface gráfica é fornecer uma maneira simplificada e de fácil entendimento para o consumidor da biblioteca, de forma a potencializar seu uso.

A figura 19 apresenta o menu principal, isto é, a tela em que é feita a administração dos usuários, permitindo o acesso aos módulos responsáveis por criar perfis de acesso e políticas de segurança.

4.3.3.1 Criação de papeis

Esta é a etapa inicial do uso da interface gráfica, esta contempla a criação de perfis de acesso, ou seja, será possível que o usuário crie os papeis e para cada papel criado seja viável associar múltiplos usuários. Como é sugerido pelo RBAC, é esperado que haja uma hierarquia entre os perfis de acesso, logo, cada perfil poderá ser também associado a uma ou várias permissões.

A figura 20 apresenta a tela onde o usuário poderá criar novos perfis para a aplicação. Ao entrar nesta tela é exibido um formulário onde é possível nomear uma função,

Figura 19 – Tela principal da ferramenta Granar
Fonte: Elaborado pelo autor (2022)

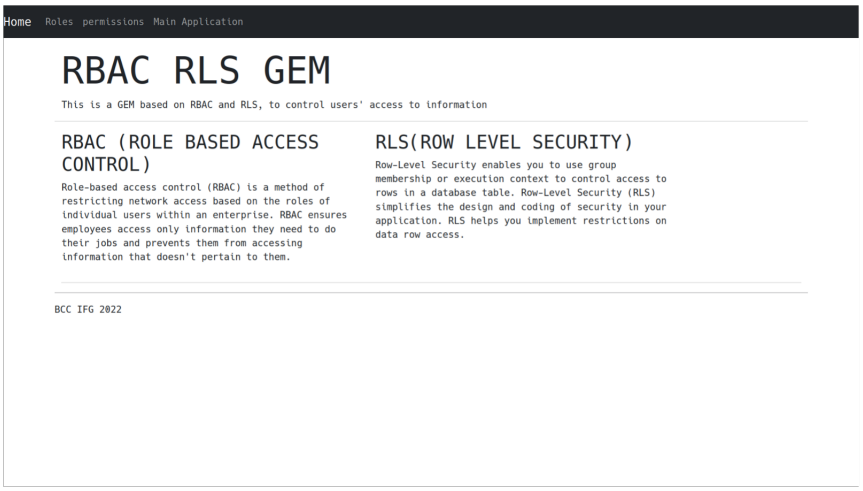


Figura 20 – Tela de criação de funções
Fonte: Elaborado pelo autor (2022)

bem como adicionar uma breve explicação sobre o seu papel na organização. Ademais, é possível associar usuários a determinadas funções e também relacionar permissões a funções específicas.

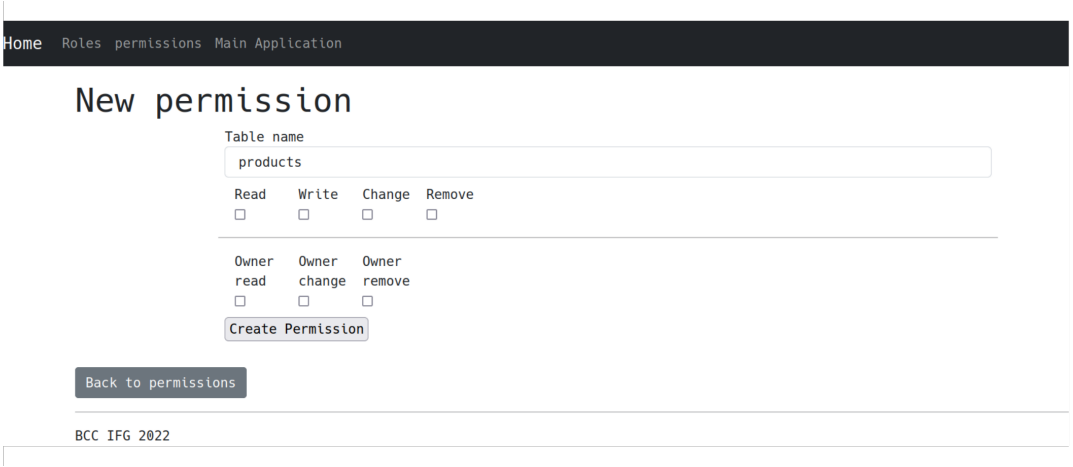
4.3.3.2 Definição de Politicas de segurança

Durante o uso da aplicação, cada usuário autenticado possuirá um código identificador atrelado ao seu perfil, este será utilizado em todas as consultas feitas pela aplicação no banco de dados. Assim, todas as restrições de acesso serão feitas com base neste atributo identificador, determinando um tratamento específico para cada usuário com base em suas funções e permissões definidas.

Toda função criada poderá ter múltiplas permissões de acesso associadas a ela,

cada permissão definida na aplicação será atendida como uma política de segurança do Row Level security que fará o controle de acesso por parte do banco de dados.

Figura 21 – Tela de criação de Funções da ferramenta Granar
Fonte: Elaborado pelo autor (2022)



A figura 21 apresenta a tela onde o usuário poderá criar novas permissões de acesso. Ao entrar nesta tela é exibido um formulário onde é possível criar uma nova permissão que será associada a uma política de restrição do Row Level Security.

Para criar uma nova permissão é necessário informar a qual tabela do banco de dados a permissão será associada e quais são os níveis de acesso que ela concederá, compreendendo apenas escrita, leitura, atualização e remoção de informações ou apenas alterações do próprio usuário, seguindo a especificação de criação de permissões dos tópicos 3.1.3.2.1 e 3.1.3.2.3.

4.3.3.3 Geração de código automatico

A geração de codigo é uma etapa fundamental pois o gerador de código do framework é responsável por criar migrates específicas de RLS para o banco de dados. Uma **migrate** (migração) é um recurso oferecido por diversos frameworks de várias linguagens de programação. Seu objetivo é auxiliar o gerenciamento da estrutura do banco de dados de uma aplicação, possibilitando a inclusão, alteração e exclusão em diversos mecanismos do banco de dados como tabelas, campos e permissões de forma organizada, deixando todas essas alterações documentadas e diminuindo a quantidade de código SQL feito manualmente.

Deste modo, existem duas maneiras em que o usuário da biblioteca poderá criar políticas de segurança associadas às tabelas. O primeiro modo acontece via interface gráfica, onde o usuário submete para a aplicação um conjunto de parâmetros selecionados dentre todas as opções disponíveis de criação como citado no tópico 3.1.3.2.1. E a outra opção é que o programador gere as políticas de segurança diretamente via linha de comando, caso

necessário. Ambos os processos incluem escrever um comando de entrada para o gerador de políticas de RLS, que tem como resultado a geração automática do SQL referente a solicitação realizada.

Após receber uma entrada específica, o gerador de código iniciará a geração de comandos SQL, como exemplificados abaixo, ilustrando a criação de uma política para a tabela de produtos com o tipo de política de escrita (*(Insert)*:

Figura 22 – Política de restição Row Level Security gerada pela ferramenta Granar
Fonte: Elaborado pelo autor (2022)

```
CREATE POLICY p1653563218productsinsert ON products for  insert TO app_user with check  (
              NULLIF(current_setting('rls.user_id', TRUE), '')::bigint in
         (
           SELECT
             ur.user_id
           FROM permissions p
             INNER JOIN role_permissions rp  on rp.permission_id  = p.id
               INNER JOIN user_roles ur  on rp.role_id  = ur.role_id
           WHERE (p."write")
             AND p.table_name = 'products'
         )
       )
```

A figura 22 apresenta um comando SQL para criar uma política de restrição de escrita no banco de dados, validando se os papeis do usuário logado na aplicação possuem o direito de escrita na tabela de produtos do banco de dados.

A figura 23 apresenta uma migração feita para o banco de dados contendo todo conteúdo necessário para a efetivação da política gerada. Este é o resultado da etapa final de geração de código da biblioteca, a qual será inserida pela biblioteca automaticamente no banco de dados.

4.4 Otimização do banco de dados

Para a otimização de um SGBD é necessário eliminar os possíveis problemas que possam impactar o desempenho, como, consultas mal estruturadas, sendo assim, é necessário considerar tratar consultas lentas a fim de permitir que haja uma melhor utilização das políticas de segurança.

Toda a estrutura da ferramenta Granar foi pensada para reduzir a quantidade de políticas de seguranças inseridas no banco de dados, logo, o modelo de código gerado permite que as políticas de segurança utilizem informações contidas nas tabelas de permissões, usuários, grupos e papéis para realizar a filtragem de dados, logo as permissões de restrição básicas, individuais e permissões de grupos são tratadas de forma genéricas pelo sistema gerenciador de banco de dados, conforme gerado pelo Framework. Tal procedimento

Figura 23 – Exemplo de Migrate gerada pela ferramenta Granar
Fonte: Elaborado pelo autor (2022)

```
class CreateProductsPolicy < ActiveRecord::Migration[7.0]
  def change
    execute 'GRANT  insert ON products TO app_user'
    execute 'ALTER TABLE products ENABLE ROW LEVEL SECURITY'
    reversible do |dir|
      dir.up do
        execute <<-SQL
          CREATE POLICY p1653563218productsinsert ON products for  insert TO app_user with check  (
               NULLIF(current_setting('rls.user_id', TRUE), '')::bigint in
          (
            SELECT
              ur.user_id
            FROM permissions p
              INNER JOIN role_permissions rp  on rp.permission_id  = p.id
                INNER JOIN user_roles ur  on rp.role_id  = ur.role_id
            WHERE (p."write")
              AND p.table_name = 'products'
          )
        )
        SQL
      end
    end
  end
end
```

possibilita que existam menos politicas por tabelas, contribuindo então de forma direta para ocorrer menos verificações de direito de acesso e de forma indireta para uma manutenção mais simples de toda a estrutura.

É possível ver na imagem 23 uma política de segurança gerada pela ferramenta Granar, esta possui uma estrutura genérica que utiliza informações contidas em outras tabelas do banco de dados para realizar o controle de acesso. Um exemplo deste comportamento é percebido quando no lugar de criar uma política explicitando que um usuário específico da aplicação não pode acessar um recurso de leitura ou escrita nesta tabela, a ferramenta gera uma política que utiliza informações do contexto da aplicação, e com isso faz apenas uma única sub consulta para verificar os direitos de acesso para o registro da tabela consultada.

Para contribuir com o desempenho, para todas as colunas de tabelas referenciadas em políticas de segurança geradas felo Framework, foram criados de forma automática índices seguindo o padrão imposto pelo SGBD, isto tem por objetivo de melhorar o desempenho das consultas afetadas pelas políticas de segurança. Índices são estruturas auxiliares que permitem a associação a uma tabela ou coleção de dados com a finalidade de acelerar o tempo de acesso às linhas de uma tabela, criando ponteiros para os dados.

4.5 Estudos experimentais

4.5.1 Teste de desempenho

Com a finalidade de determinar o impacto do uso do modelo Granar no que se refere ao desempenho da aplicação, foram realizados diversos experimentos. Como descrito anteriormente, esta metologia utiliza o método denominado Row Level Security para realizar o controle de acesso granular aos dados, este recurso não se utiliza da reescrita de consultas e sim da aplicação de sub consultas como filtro da consulta original. Sendo assim, nesta sessão busca-se determinar o impacto causado pela utilização de tal comportamento, uma vez que para todo objeto associado a uma política, existe uma etapa adicional no curso natural da execução da consulta SQL feita ao banco de dados.

4.5.1.1 Métrica e componentes utilizados

Considerando o objetivo de se descobrir o custo da utilização desta metodologia que faz uso do RLS, a métrica escolhida para avaliar o desempenho do modelo foi a análise do tempo de execução de consultas em conjunto de dados de diversos tamanhos.

Ademais, fatores como ambiente com grande concorrência por tempo de execução devem ser desconsiderados, pois o cenário utilizado para a simulação conta com apenas um usuário solicitando diversas consultas ao banco de dados.

As consultas utilizadas para os testes feitos, foram geradas por rotinas automatizadas implementadas via aplicação, sendo então uma interação apenas entre aplicação e banco de dados.

Para todos os testes o tempo de execução começa a ser computado assim que é iniciada uma nova conexão com o banco de dados e então é finalizada a contagem assim que a aplicação recebe a resposta do banco de dados acerca da requisição feita, compreendendo o início e o fim de uma requisição.

Também é importante ressaltar que os comandos utilizados envolvem poucas tabelas para junções.

A tabela 4.5.1.1 exibe os principais componentes utilizados para a realização dos testes. É válido ressaltar que os resultados estão fortemente ligados a plataforma e ao hardware que foi utilizado. Deste modo, pode haver variações nos resultados ao se adotar componentes de hardware ou softwares com características distintas das expostas a seguir.

4.5.1.2 Testes de performance

Após definidos os parâmetros e métricas de avaliação utilizados, no tópico 4.5.1.1, realizou-se uma série de experimentos para coletar informações acerca do tempo de execução de consultas. Em sua totalidade, 6400 experimentos foram realizados, dentre eles consultas

Tabela 1 – Parâmetros dos experimentos realizados

Componente de teste	**Estado**
Processador	Core i7 10700
Memoria Ram	16GB DDR4 Modelo AD4S26668G19-SGN
SDD	256GB Modelo MZ-V7S250B/AM
SGBD	PostgreSQL 14
Comandos Utilizados	Select,Insert
Utilização de índices	Sim

de leitura e escrita, sendo então 3200 sem a utilização da ferramenta na aplicação e 3200 utilizando a metodologia Granar.

As listagens 4.1 e 4.2, mostram exemplos de SQL utilizadas nos testes de desempenho, sendo estes os exemplos de leitura e escrita.

Listing 4.1 – Exemplo de SQL de inserção via aplicação

```
1
2  INSERT
3          INTO
4          "ecommerce_benchmarks" ("status",
5          "sku",
6          "price",
7          "qty_ordered",
8          "grand_total",
9          "increment_id",
10         "category_name_1",
11         "sales_commission_code",
12         "discount_amount",
13         "payment_method",
14         "working_date",
15         "bi_status",
16         "mv",
17         "year",
18         "month",
19         "customer_since",
20         "m_y",
21         "fy",
22         "customer_id",
23         "created_at",
24         "updated_at")
```

```
25 VALUES('211131',
26 'complete',
27 7.0,
28 0.0,
29 1950.0,
30 1,
31 '1950',
32 '100147443',
33 0.0,
34 '\\N',
35 '23/12/2022',
36 '7/1/2016',
37 '#REF!12',
38 '1',
39 'rEF99',
40 '950␣\',
41 '2016',
42 '---',
43 '7',
44 now(),
45 now())
```

Listing 4.2 – Exemplo de SQL

```
1
2 SELECT
3         id,
4         status,
5         sku,
6         price,
7         qty_ordered,
8         grand_total,
9         increment_id,
10        category_name_1,
11        sales_commission_code,
12        discount_amount,
13        payment_method,
14        working_date,
15        bi_status,
16        mv,
```

```
17          "year",
18          "month",
19          customer_since,
20          m_y,
21          fy,
22          customer_id
23  FROM
24          ecommerce_benchmarks eb
25  INNER JOIN ecommerce_benchmark_associations eba
26  ON
27          eba.customer_id = eb.customer_id
```

Na tabela 2 encontra-se a descrição dos experimentos realizados para cada respectivo tamanho da coleção. Tais experimentos estão divididos em dois casos, sendo eles, implementação do controle acesso feito diretamente via aplicação e com a utilização da metodologia Granar, isto é, controle de acesso realizado via banco de dados.

Tabela 2 – Condições aplicadas no experimento

Tipo	Comando	Nº de experimentos	Tamanho da coleção	Politicas aplicadas
Leitura	SELECT	200	1000	Sim
leitura	SELECT	200	10.000	Sim
leitura	SELECT	200	10.000	Sim
Leitura	SELECT	200	100.000	Sim
Leitura	SELECT	200	1.000.000	Sim
Escrita	INSERT	200	1	Sim
Escrita	INSERT	200	10	Sim
Escrita	INSERT	200	100	Sim
Leitura	SELECT	200	1000	Não
leitura	SELECT	200	10.000	Não
leitura	SELECT	200	10.000	Não
Leitura	SELECT	200	100.000	Não
Leitura	SELECT	200	1.000.000	Não
Escrita	INSERT	200	1	Não
Escrita	INSERT	200	10	Não
Escrita	INSERT	200	100	Não

4.5.1.2.1 Resultados

Os resultados obtidos nos testes de desempenhos estão expostos na tabela 4.5.1.2.1. A partir deles, é possível observar que há uma diferença de comportamento de acordo com a quantidade de dados.

Tabela 3 – Resultados dos experimentos realizados

Tempo médio de execução	Comando	Tamanho da base	Políticas de segurança
0.413 ms	SELECT	1000	Sim
9.787 ms	SELECT	10.000	Sim
54.395 ms	SELECT	100.000	Sim
452.132 ms	SELECT	1.000.000	Sim
1.5 ms	INSERT	1	Sim
9.8 ms	INSERT	10	Sim
10.2 ms	INSERT	100	Sim
0.236 ms	SELECT	1000	Não
2.509 ms	SELECT	10.000	Não
28.292 ms	SELECT	100.000	Não
302.723 ms	SELECT	1.000.000	Não
1.5 ms	INSERT	1	Não
8.5 ms	INSERT	10	Não
9.9 ms	INSERT	100	Não

A partir dos gráficos expostos na figura 25 é possível notar o custo temporal advindo do investimento em segurança entre consultas onde objetos que possuem associações a políticas de restrições geradas por granar e tabelas que não possuem restrição alguma, isto é, a adição da etapa de conferência das políticas adotada pela estrutura Granar ocasiona um tempo adicional no resultado, sendo este, de aproximadamente 42% para registros menores e 33% para maiores. Contudo, este fato já era esperado e está diferença não é considerada tão significativa quando os recursos computacionais atuais são analisados.

Ao analisar os resultados obtidos, nota-se que para as operações de leitura, no geral, em situações onde o número de registros é de até mil, como demonstrado na figura 24, existe um aumento ínfimo de cerca de 0.177**ms**. Já para operações maiores, nota-se que houve um aumento significativo para consultas que buscam de 10 mil até 100 mil registros, é possível visualizar melhor estes resultados ao observar a figura 25.

Já para casos em que o número de registros é de cerca de 1 milhão, observa-se que o aumento é de aproximadamente 49%, ou seja, cerca de 150 **ms** em relação a um modelo em que não se utiliza políticas de segurança. Ao observar a figura 25, conclui-se também que o aumento no tempo de execução é proporcionalmente menor para casos onde a base de dados possui um maior número de informações.

A figura 26 mostra o gráfico contendo as projeções para testes de escrita, nestes, em todos os casos testados, exceto para a inclusão de um único registro, houve um aumento de cerca de 1.3**ms** para múltiplas inserções de registros em uma tabela.

Figura 24 – Resultado teste de leitura com 1.000 registros
Fonte: Elaborado pelo autor (2022)

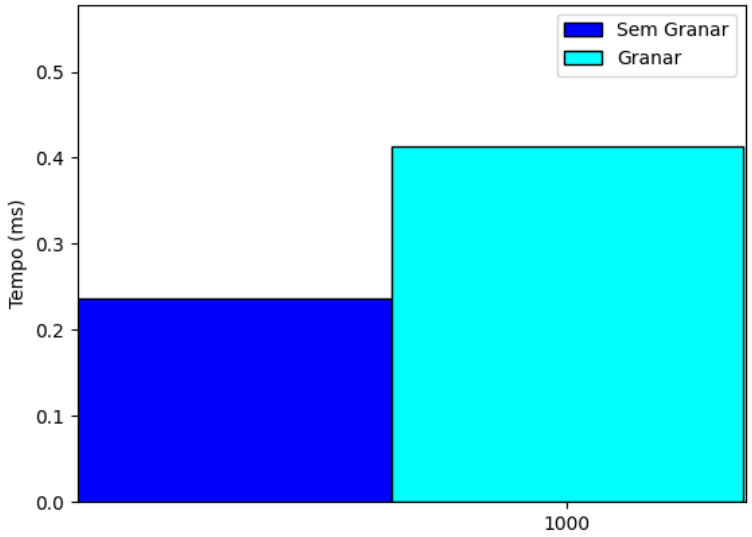


Figura 25 – Gráfico geral de resultados para consultas de leitura de até 1.000.000 de registros
Fonte: Elaborado pelo autor (2022)

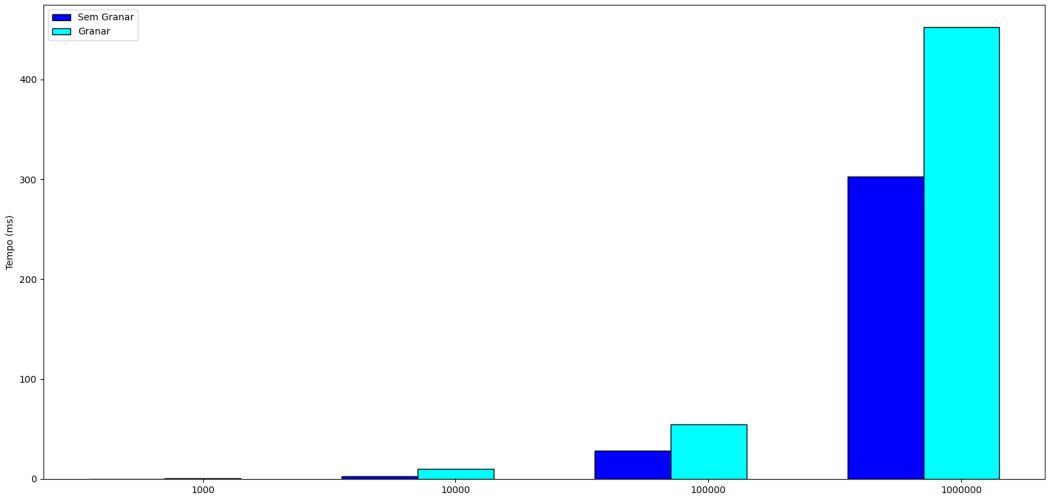
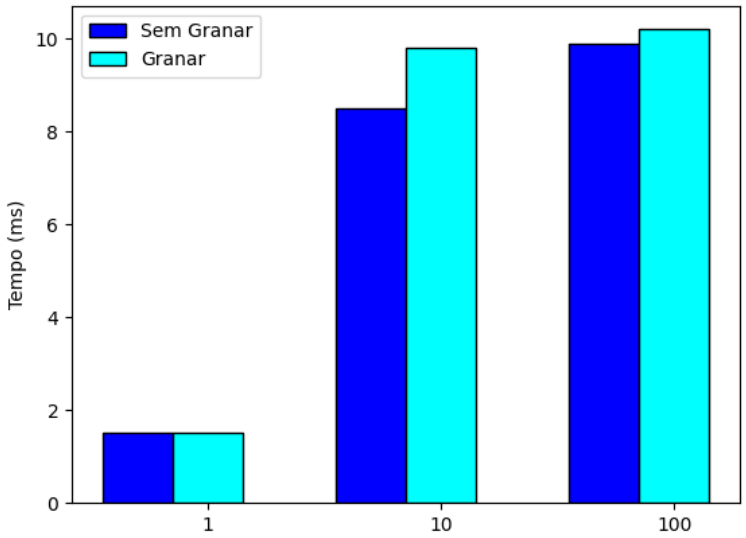


Figura 26 – Resultado das operações de escrita
Fonte: Elaborado pelo autor (2022)



4.6 Testes com voluntários

Com a finalidade de determinar a efetividade prática da metodologia implementada na ferramenta Granar, uma pesquisa com trabalhadores da área de tecnologia com diferentes níveis de experiência e áreas de atuação foi realizada. Seu objetivo está centrado na avaliação do funcionamento, eficácia e praticidade de utilização da ferramenta implementada. Ademais, objetiva-se avaliar seu processo de integração com a aplicação, bem como, colher informações acerca de possíveis melhorias.

O experimento foi realizado a partir da disponibilização da ferramenta para download, bem como, um tutorial exemplificando desde a etapa de instalação, utilização e remoção da estrutura. Após a etapa de utilização, foi solicitado que respondessem a um formulário de forma **anônima**. Encontra-se a seguir os resultados obtidos através do teste feito com o total de dez voluntários, dentre eles todos da área te Tecnologia da informação, profissionais experientes e iniciantes.

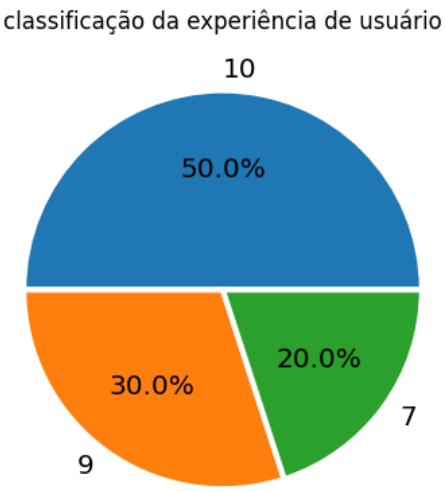
4.6.1 Resultados

O experimento buscou avaliar a ferramenta desenvolvida sob diversos fatores, sendo eles, eficácia, experiência de usuário, processo de instalação e desinstalação e a possibilidade de se indicar o uso da estrutura em projetos futuros.

A figura 27 expõe a opinião geral dos usuários no que se refere à sua experiência geral utilizando a ferramenta, isto é, desde a instalação, utilização, manutenção e exclusão. A partir dela é possível notar que 50% dos participantes classificam a experiência como 10

em uma escala de 0 - 10, sendo 0 muito ruim e 10 ótimo. Ademais, 30% dos utilizadores a classifica como 9 e 20% como 7.

Figura 27 – Classificação da experiência de usuário
Fonte: Elaborado pelo autor (2022)



A figura 28 detalha a opinião dos usuários a respeito da eficácia da metodologia Granar, ou seja, considerando que a finalidade deste trabalho é propor um modelo para controle de acesso granular aos dados, contribuindo para a segurança através de verificações de acesso à informação, quão bem a estrutura alcança tal meta. Sendo assim, 70% dos utilizadores classifica a eficácia em 10, 20% em 9 e 10% em 8.

Já a figura 29 relata a opinião dos usuários acerca da facilidade e intuitividade para se instalar e desinstalar a biblioteca Granar. Apenas 30% dos usuários classifica o processo como nota 10, 40% como 9, 10% como 8 e 20% como 6. Tal resultado revela a necessidade de se aprimorar, tal etapa, bem como, elaborar manuais concisos e detalhadas sobre o procedimento a ser realizado.

Por fim, a figura 30 revela a probabilidade de o usuário em questão indicar o uso da biblioteca Granar em seu próximo projeto pessoal ou ambiente de trabalho. Os resultados mostram que 30% dos usuários definiram como nota 10 a possibilidade de indicação, 30% com nota 9, 20% considera como 8 essa chance e 20% como 7.

Enfim, este teste é de bastante utilidade e demonstra altos níveis de eficácia segundo a opinião dos utilizadores, bem como, evidencia a necessidade de melhorar procedimentos como instalação e desinstalação.

Figura 28 – Classificação da eficácia da ferramenta
Fonte: Elaborado pelo autor (2022)

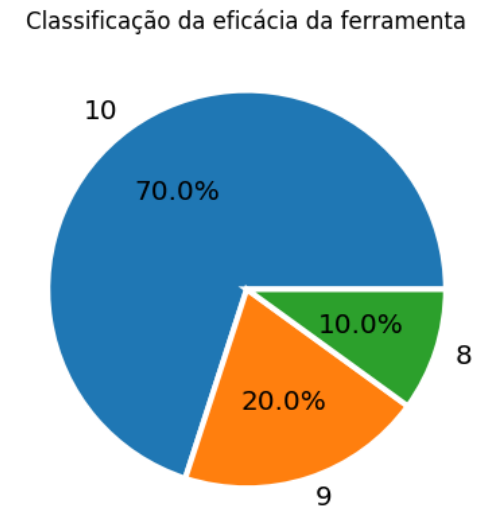


Figura 29 – Intuitividade do processo de instalação e desinstalação
Fonte: Elaborado pelo autor (2022)

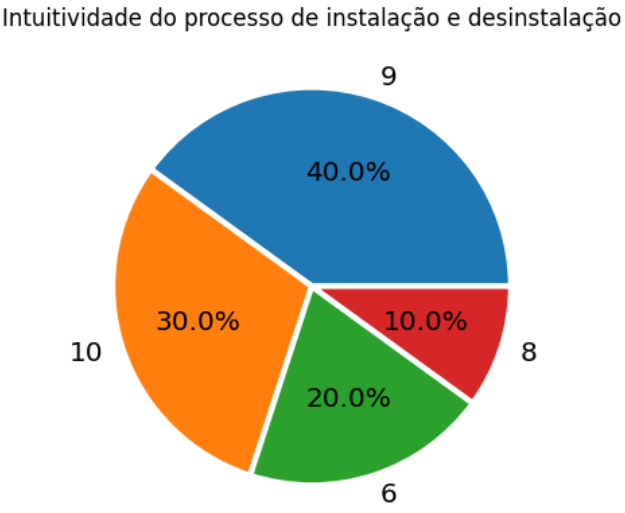
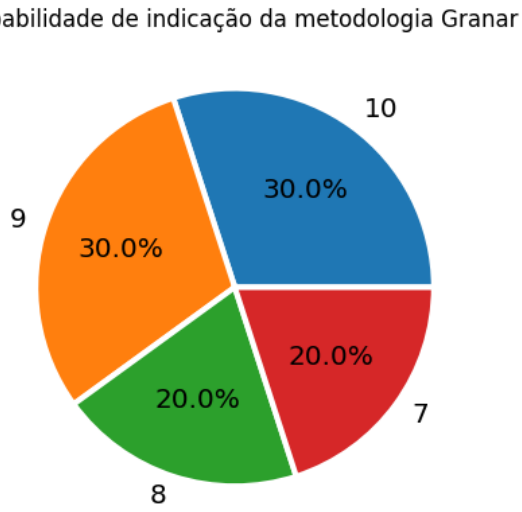


Figura 30 – Probabilidade de indicação da modelo Granar
Fonte: Elaborado pelo autor (2022)



4.7 Trabalhos futuros

A partir do teste com os usuários, conclui-se que a primeira avaliação da ferramenta foi positiva. Contudo, faz-se necessário entender e expor alguns pontos de melhoria a serem tratados em trabalhos futuros.

Um tópico importante a ser abordado, é permitir que o usuário da biblioteca possa realizar o controle de acesso a rotas juntamente com o controle de acesso granular aos dados disponibilizado pelo framework Granar. Implementar esta funcionalidade geraria um recurso que possibilitaria uma estrutura auxiliar para contribuir com a segurança da aplicação.

Algo interessante, também, é a possibilidade de se criar políticas mais customizadas, estas podendo ser gerada pelo próprio programador e serem definidas juntamente com as permissões que já são criadas via interface, possibilitando assim, uma maior customização das políticas de segurança.

Até o momento, a ferramenta possibilita a integração apenas com o SGBD PostgreSQL, considerando as diversas opções disponíveis no mercado, seria interessante haver a possibilidade de integração com outros sistemas gerenciadores de banco de dados como Oracle, Mysql e SQL Server. Tal ação possibilitaria uma maior gama de usos para a ferramenta, bem como, deixaria transparente ao programador os detalhes de implementação de políticas de segurança em ambientes onde existe uma diversidade de tipos de bancos de dados utilizados pela aplicação.

Por fim, realizar experimentos com uma quantidade maior de participantes seria

interessante, uma vez que tal procedimento potencializaria a precisão dos resultados acerca do processo adotado pela metodologia Granar.

4.7.1 Disponibilização da ferramenta

Atualmente a ferramenta gerada seguindo os procedimentos citados neste capitulo, está disponivel no repositório RubyGems (MAIA, 2022)

5 Conclusão

O desenvolvimento deste trabalho iniciou-se com o detalhamento e a especificação acerca do controle de acesso aos dados, mais especificamente, aspectos referentes a controle de acesso granular aos dados, sendo este, um método utilizado na camada de banco de dados, utilizando-se de sub consultas associadas a objetos de banco de dados para limitar os acessos de leituras e escritas.

Este trabalho buscou unir os recursos do Role-based Access Control, isto é, a restrição de acesso com base na função ou papel do usuário fundamentando-se em privilégios específicos que ele possua ao Row-level security, ou seja, aplicação de um processo de filtragem nos dados consultados no banco de dados. Sendo assim, o modelo Granar utiliza a geração de papéis e funções para criação de políticas de segurança que serão aplicadas diretamente no banco de dados e farão a filtragem dos dados que estarão envolvidos nas operações solicitadas pelo usuário.

Para validar a tratativa de controle de acesso proposta pelo trabalho, houve a implementação de uma ferramenta baseada nos conceitos descritos, e através de estudos experimentais analisou-se a eficácia do método proposto.

Considerando que a ferramenta é baseada no método proposto e foi implementada utilizando o PostgreSQl como SGBD, é válido ressaltar que há a possibilidade da aplicação da mesma estrutura em outros sistemas gerenciadores de banco de dados que comportem o recurso de segurança em nível de linha.

Este modelo define uma estrutura utilizada para implementar políticas de segurança atreladas diretamente a objetos do banco de dados, bem como, possibilita a existência de regras de acesso utilizadas para delimitar a porção de informações que estarão disponíveis a usuários específicos. Ademais, tal procedimento contribui para a transparência e visibilidade organizacional no que tange o acesso aos dados, uma vez que todos os procedimentos, regras e responsáveis por determinadas instâncias serão explicitamente definidos e facilmente gerenciados.

Ao analisar os resultados de desempenho obtidos no teste de desempenho, conclui-se que ocorre um aumento em relação ao tempo de execução das consultas quando o modelo Granar é utilizado. Contudo, tal fato já era esperado, pois a carga de processamento que antes seria da aplicação, é então direcionada ao banco de dados.

Ademais, há uma considerável redução da produção manual de código-fonte destinado ao controle de acesso aos dados. Como consequência, nota-se uma maior facilidade de manutenção e organização do sistema e da hierarquia de cargos, papéis e funções.

A partir do teste com usuários, infere-se o grande potencial da metodologia proposta por este trabalho, isto é, seu nível de aceitação e eficiência segundo os utilizadores foi bastante satisfatório e promissor.

Referências

BISHT, P. et al. Automated detection of parameter tampering opportunities and vulnerabilities in web applications. *Journal of computer security*, IOS Press, v. 22, n. 3, p. 415–465, 2014. Citado na página 21.

BOOCH, G. *UML: guia do usuário*. [S.l.]: Elsevier Brasil, 2006. Citado na página 26.

CARTER, P. A. *Data-Level Security*. [S.l.]: Apress, Berkeley, CA, 2018. Citado na página 24.

CHAUDHURI, S.; DUTTA, T.; SUDARSHAN, S. Fine grained authorization through predicated grants. In: IEEE. *2007 IEEE 23rd International Conference on Data Engineering*. [S.l.], 2007. p. 1174–1183. Citado na página 27.

CHEN, P. P.-S. The entity-relationship model—toward a unified view of data. *ACM Trans. Database Syst.*, Association for Computing Machinery, New York, NY, USA, v. 1, n. 1, p. 9–36, mar 1976. ISSN 0362-5915. Disponível em: <https://doi.org/10.1145/320434.320440>. Citado na página 27.

COELHO, F. E. S. *Gestão da Segurança da Informação – NBR 27001 e NBR 27002*. [S.l.]: Escola Superior de Redes – RNP, 2014. Citado 2 vezes nas páginas 5 e 20.

ELMASRI RAMEZ; NAVATHE, S. B. *Sistemas de banco de dados. 7ª ed.* [S.l.]: Editora Pearson, São Paulo, 2018. Citado na página 22.

FRANCK, K. M.; PEREIRA, R. F.; FILHO, J. V. D. Ratio-entity diagram: a tool for conceptual data modeling in software engineering. *Research, Society and Development*, v. 10, n. 8, p. e49510817776, Jul. 2021. Disponível em: <https://rsdjournal.org/index.php/rsd/article/view/17776>. Citado na página 27.

GUIMARÃES, A. G.; LINS, R. D.; OLIVEIRA, R. C. de. *Segurança em Redes Privadas Virtuais-VPNs*. [S.l.]: Brasport, 2006. Citado na página 20.

HAMMER MICHAEL E CHAMPY, J. *revolucionando a empresa em função dos clientes, da concorrência e das grandes mudanças da gerência. 17. ed.* [S.l.]: Campus Rio de Janeiro, 1994. Citado na página 14.

HOHENSTEIN, U.; ZILLNER, S.; BIESDORF, A. Architectural considerations for a data access marketplace. In: *DATA*. [S.l.: s.n.], 2018. p. 322–333. Citado na página 28.

ISO/IEC:17799. *Information technology — Security techniques — Code of practice for information security management*. [S.l.]: Multiple. Distributed through American National Standards Institute, 2005. Citado na página 19.

ISO/IEC:27002, A. N. *Tecnologia da informação — Técnicas de segurança — Código de prática para controles de segurança da informação*. [S.l.]: ABNT - Associação Brasileira de Normas Técnicas, 2013. Citado na página 19.

MAIA, F. B. *2022 Granar*. [S.l.], 2022. Disponível em <https://rubygems.org/gems/rbac_rls>, Acessado em: 24.01.2023. Citado na página 63.

MARQUES, M. G. *Uma metodologia para controle de acesso granular em sistemas de banco de dados relacionais open source*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2013. Citado na página 28.

MASON, M. E. S. . G. UML-Based Representation of Role-Based Access Control. *IEEE*, p. 195–196, 2000. Citado na página 22.

PFLEEGER, S. L. P. C. P. *Security in computing. 4ª ed.* [S.l.]: Editora Prentice Hall,Massachusetts, 2012. Citado na página 17.

PRESSMAN, R. S. *Software engineering: a practitioner's approach*. [S.l.]: Palgrave macmillan, 2005. Citado na página 27.

SANDHU, R.; SAMARATI, P. Access control: principle and practice. *IEEE Communications Magazine*, v. 32, n. 9, p. 40–48, 1994. Citado na página 14.

SPENDOLINI, S. Virtual private database. In: *Expert Oracle Application Express Security*. [S.l.]: Springer, 2013. p. 211–223. Citado na página 27.

STONEBRAKER, M.; WONG, E. Access control in a relational data base management system by query modification. In: *Proceedings of the 1974 annual conference-Volume 1*. [S.l.: s.n.], 1974. p. 180–186. Citado na página 27.

SUUS. *Row Level Security for Ruby on Rails*. [S.l.], 2019. Disponível em <https://github.com/suus-io/rls_rails>, Acessado em: 30.12.2021. Citado na página 29.

VERIZON. *2019 Data Breach Investigations Report*. [S.l.], 2019. Disponível em <https://www.verizon.com/business/resources/reports/2019-data-breach-investigations-report.pdf>, Acessado em: 30.12.2021. Citado na página 14.

ZHANG, X. M. et al. *Designing a SQL query rewriter to enforce database Row Level Security*. Tese (Doutorado) — Massachusetts Institute of Technology, 2016. Citado na página 24.

ZHU, L. et al. Combined access control model embedding configurable policy for fine-grained data security. *Microprocessors and Microsystems*, Elsevier, v. 75, p. 103060, 2020. Citado na página 28.