Associação Propagadora Esdeva
Centro Universitário Academia – UniAcademia
Curso de Sistemas de Informação
Trabalho de Conclusão de Curso – Artigo

DATAW:
API WEB PARA ANONIMIAZAÇÃO DE DADOS

Mateus Souza de Melo[1]
Centro Universitário Academia, Juiz de Fora, MG

Evaldo de Oliveira da Silva[2]
Centro Universitário Academia, Juiz de Fora, MG

Linha de Pesquisa: Engenharia de Software

RESUMO.

Com o surgimento da Lei Geral de Proteção de Dados (LGPD), as organizações enfrentam muitos desafios para garantir a privacidade e a segurança de informações pessoais, exigindo tecnologias eficientes para anonimização de dados que garantam a privacidade e a conformidade legal. Entre os desafios encontrados, destaca-se a dificuldade de implementar soluções que preservem a identificação de indivíduos no uso de dados pessoais por partes das empresas. Esse trabalho tem como objetivo desenvolver uma API web, chamada *DataW*, projetada para anonimizar dados utilizando o método de generalização que substitui informações especificas por valores mais genéricos, em conformidade com a LGPD. Para atingir esse objetivo, foi projetada uma API REST utilizando MVC e *Strategy* em sua arquitetura, através de tecnologias como Node.js, PostgreSQL com a extensão *Anonymizer* e o framework Express.js. A API foi implementada de forma modular, promovendo baixo acoplamento e facilidade de manutenção, permitindo sua configuração e uso através de requisições HTTP através dos *endpoints*. Os resultados demonstram que a *DataW* é capaz de receber conjuntos de dados em formato JSON, processá-los através de requisições HTTP e retornar o resultado, garantindo sua aplicabilidade em dados pessoais, preservando a identidade de pessoas naturais. Embora funcional, a *DataW* apresenta limitações relacionadas ao suporte exclusivo ao método de generalização. A inclusão de outras técnicas de anonimização, bem como o desenvolvimento de uma interface gráfica para facilitar a interação com a API são possibilidades para um trabalho futuro. Apesar das limitações, esta solução representa uma contribuição significativa para a proteção de dados pessoais, alinhando a tecnologia em conformidade com a LGPD.

Palavras-chave: dados pessoais, proteção de dados, privacidade, anonimização, lgpd, generalização, api.

[1] Discente do Curso de Sistemas de Informação do Centro Universitário Academia – UniAcademia.
[2] Docente do Curso de Sistemas de Informação do Centro Universitário Academia - UniAcademia

1. INTRODUÇÃO

A Lei 13.709, Lei Geral de Proteção de Dados (LGPD) entrou em vigor no Brasil em 2020 (BRASIL 2018). Foi criada com o objetivo de regulamentar o tratamento de dados pessoais em território nacional, ao modo que, harmoniza, complementa e unifica um ecossistema de mais de 40 normas setoriais, permitindo padronizar de forma direta e indireta a proteção e a privacidade dos dados em questão. A LGPD teve como base discussões que resultaram na criação da lei europeia, nomeada como GDPR (*General Data Protection Regulation*), e permite que pessoas tenham um controle maior sobre seus dados, assim como também igualar interesses econômicos e sociais. (MONTEIRO, 2018).

Segundo Bisso et al. (2020) os dados pessoais dentro do contexto das leis LGPD e GDPR são informações que possibilitam de maneira direta ou indireta identificar uma pessoa natural, por exemplo, o código de pessoa física (CPF), registro geral (RG) e nome completo. Outros dados como IP, profissão e localização são considerados não pessoais e podem se tornar dados pessoais também caso sejam utilizados em conjunto para identificar uma pessoa natural. Informações que podem violar a intimidade, honra e imagem das pessoas naturais, convicções religiosas, políticas, origem racial e étnica, dados genéticos, biométricos ou algo em relação à saúde ou vida sexual de um indivíduo, são considerados dados pessoais sensíveis.

A implementação da LGPD exige que processos operacionais sejam estabelecidos pelas organizações, envolvendo diferentes desafios. Ao interpretar a lei, esses processos podem estar relacionados ao controle da inserção de novos dados, anonimização e alteração dos dados existentes e a concessão de perfis de acessos. Porém, a lei não determina como esses processos devem estar especificados. A ausência de clareza dos órgãos responsáveis pela aplicação da lei dificulta que profissionais e organizações estejam adequadamente preparados. Com isso, a LGPD demanda por novas tecnologias, treinamento adequado de profissionais, tratamento de dados, processos de governança de dados. Essas demandas aumentam o custo e o nível de conhecimento para aplicar a lei (CAIRES, 2023).

A LGPD é uma legislação que requer um profundo conhecimento dos seus termos e conceitos legais, de modo que sua correta interpretação e aplicação se torna um desafio. Outro fator a ser considerado é a necessidade de criar uma mudança cultural dentro das organizações, não se restringindo apenas a aspectos técnicos, mas promovendo uma conscientização e capacitação dos envolvidos a respeito da proteção de dados (SOUZA, 2024). Até os dias atuais, diversas instituições

foram impactadas com a aplicação da LGPD, desde pequenas até grande porte, onde cerca de 80% das empresas não se adequaram totalmente as exigências da lei, apesar do prazo desde sua aprovação. As empresas de pequeno porte geralmente são as que mais tem dificuldades para sua adequação, pois em sua maioria, não contam com uma assessoria especializada para essa tarefa (SOUSA, 2023).

A necessidade das organizações terem que se adequar à LGPD para fazerem o tratamento de dados pessoais por meio de tecnologias e processos, oportuniza a customização de soluções de software para facilitar sua implantação. A anonimização de dados é uma tarefa que necessita de soluções que envolvem a aplicação de técnicas em arquitetura de software e banco de dados para apoiar a privacidade de dados prevista pela LGPD. Desta forma, este trabalho tem como objetivo geral aplicar a anonimização de dados por meio de uma API web chamada *DataW* (*Data AnonymizaTion by API Web*). Como objetivos específicos, tem-se os seguintes :

- Pesquisar métodos de anonimização e banco de dados para suportarem a implementação;
- Especificar uma interface de processamento de aplicativos entre um servidor web e um navegador web (API Web).
- Implementar o protótipo de uma API Web para anonimizar dados.

O restante do trabalho encontra-se organizado em seções. A Seção 2 descreve o referencial teórico que fundamenta os conceitos de anonimização e as tecnologias aplicadas. Os conceitos abordados se baseiam na documentação oferecida pela extensão PostgreSQL *Anonymizer*, disponível para o banco de dados relacional PostgreSQL. A Seção 3 apresenta os métodos de anonimização. A Seção 4 apresenta a implementação e execução do *DataW*, enquanto a seção 5 demonstra os casos de uso. A Seção 6 faz as considerações finais e apresenta trabalhos futuros.

2. REFERENCIAL TEÓRICO

Esta seção descreve os métodos de anonimização de dados por meio da documentação oferecida pelo PostgreSQL *Anonymizer* (POSTGRESQL, 2024a). Ele é uma extensão para mascarar ou substituir informações de identificação pessoal (*Personally Identifiable Information*) ou dados comercialmente sensíveis. As ferramentas Anoppi e Web *Anonymizer* são descritas como trabalhos correlatos, criadas com o objetivo de fornecer anonimização de dados.

2.1. MÉTODOS DE ANONIMIZAÇÃO

Existem várias técnicas disponíveis que podem ser utilizadas para atingir a anonimização dos dados, impedindo-o de ser vinculado a uma pessoa natural. De acordo com a documentação do PostgreSQL *Anonymizer* (POSTGRESQL, 2024b) e Silva (2020), as técnicas para anonimizar dados pessoais são as seguintes:

- *Destruction* (Destruição): A forma mais rápida e direta de anonimizar os dados de algum indivíduo é destruí-los. Sendo assim, a proposta dessa técnica é ocultar o valor de um dado ao substituí-lo por algum valor estático. Por exemplo, em uma base de dados, onde uma determinada coluna representa dados sensíveis, todos os valores poderiam ser substituídos pela palavra “CONFIDENCIAL”.

- *Adding Noise* (Adição de Ruído): Ao utilizar essa técnica o objetivo principal seria causar uma variação em cima de valores numéricos e datas. Para aplicar essa variação deve ser fornecido o valor a ser anonimizado e um raio, por exemplo 0.27 (27%), utilizado para deslocar o valor original para menos ou para mais com base em um valor randômico gerado no intervalo do raio entre 1% e 27%. O resultado da anonimização será retornado com base nesses valores.

- *Randomization* (Randomização): Como o próprio nome sugere, é a geração de dados randômicos utilizados no lugar dos dados pessoais. Por padrão o PostgreSQL *Anonymizer* já fornece uma série de funções para gerar dados randômicos.

- *Faking* (Falsificação): Essa forma de anonimização tem como premissa a substituição dos dados sensíveis por valores randômicos, porém mais realistas, evitando a identificação de um indivíduo por meio dos registros de dados. O PostgreSQL *Anonymizer* conta com métodos que permitem a geração desses dados como forma de facilitar o uso dessa técnica.

  É citado também o *Advanced Faking* (Falsificação Avançada), que nada mais é que uma

versão mais avançada e complexa da falsificação de dados, mas sua utilização se restringe a casos mais básicos, sendo necessário recorrer a outros recursos.

- Pseudonimização: Compartilha semelhanças com a técnica *Faking*, também gerando valores realistas. A sua principal diferença está no fato de que seu valor é determinístico, pois suas funções sempre geram o mesmo valor com base em uma seed (semente) e um salt (salto), podendo a última ser opcional, já que seu objetivo é aumentar a complexidade para evitar ataques de força bruta e dicionário.

- *Generic hashing*: Não é exatamente uma técnica de anonimização, de acordo com a documentação do PostgreSQL *Anonymizer.* Em alguns casos pode ser necessário a geração de um *hash* determinístico com base no dado original. Caso um par de chaves primárias (ou estrangeiras) seja uma "chave natural", pode haver informações reais permitindo a associação de um indivíduo. Ao utilizar o *hash* em cima dessas colunas permite que a integridade referencial permaneça intacta. Porém é importante ter em mente que essa é na verdade uma forma de pseudoanonimização, ou seja, os dados podem ser "de-anonimizado" usando o valor do *hash* e a função utilizada. Com essas 2 partes é possível re-identificar algumas pessoas usando força bruta ou dicionários.

- *Partial Scrambling* (Codificação parcial): Seu objetivo é esconder parte dos dados, substituindo a parte oculta por um caractere. Caso um cartão de crédito seja anonimizado, o resultado seria algo como: 31XX XXXX XXXX XX87.

- *Conditional Masking*: Em alguns casos talvez seja necessário a aplicação de filtros de máscara apenas para certos valores. Por algum motivo, caso seja necessário preservar valores nulos ("NULL"), ocultando apenas as ocorrências onde contém um valor.

- Generalização: Essa técnica tem como principal objetivo diminuir a precisão dos dados, substituindo o valor original por um intervalo no qual o original esteja contido, reduzindo o risco de re-identificação.

2.2. TRABALHOS CORRELATOS

Esta seção tem por objetivo apresentar trabalhos correlatos, desenvolvido pelos autores em seus respectivos artigos acadêmicos para ajudar o processo de anonimização dos dados.

A ANOPPI é um projeto financiado pelo Ministério da Justiça da Finlândia, criado especificamente para o idioma finlandês com o objetivo de anonimizar documentos de decisões judiciais de maneira automática e semi automática. Por meio de uma interface desenvolvida com ferramentas utilizadas na web, como HTML, Javascript e React, é possível interagir com a aplicação para anonimizar dados. Além da interface web, é possível interagir também através de API REST por meio de requisições web, permitindo que o processo seja realizado programaticamente, automatizando a anonimização dos documentos (ARTTU, 2022).

Ao utilizar a interface web da aplicação para enviar os documentos, ocorrerá um processo diferente no *backend* (servidor web), diferentemente de utilizar somente por meio da API REST devido às particularidades da interface web. Através da interface web o servidor armazena os documentos enviando-os para um banco de dados PostgreSQL que gera um identificador para cada um deles. Os identificadores também são armazenados na memória local do navegador do usuário. Já por meio da API REST, além de não necessitar uma interface web, nenhum dado é armazenado no banco de dados. Com a API é possível anonimizar os documentos de maneira automática ou semi automática, enviando-os na requisição feita a API, resultando como saída outro documento, anotado ou anonimizado (ARTTU, 2022).

Web *Anonymizer* é uma aplicação web criada para anonimizar dados, removendo a identificação de conjuntos de dados que contenham informações pessoais, no qual seu nome reflete bem seu objetivo. A aplicação é baseada em uma solução desktop existente chamada ARX *Anonymizer* que disponibiliza uma API (uma interface que oferece diversos métodos) usada para integrar uma API REST desenvolvida para a aplicação web citada acima. O objetivo é fornecer uma maneira fácil de anonimizar dados através de um serviço que rode na web, implementando o máximo possível de *features* apresentada na aplicação desktop na qual é baseada. Para realizar o processo, é feito o upload do conjunto de dados desejado, permitindo ao usuário configurar na aplicação uma regra que atenda às suas necessidades. Após isso, a API disponibiliza 2 conjuntos de dados, a versão original dos dados e outra anonimizada (FERREIRA, 2017)

A API Rest do Web *Anonymizer* foi desenvolvida na linguagem Java, em conjunto com o *framework* Spring Boot, permitindo ser consumida por um cliente web (ou qualquer outro) através de requisições HTTP (FERREIRA, 2017).

Ao criar a DataW como um serviço de anonimização de dados por meio de uma API Rest, comparando com os trabalhos correlatos descritos acima, é possível perceber as vantagens que este trabalho tem em relação aos seus pares. A primeira vantagem a se destacar, em relação com a ANOPPI é pelo fato do uso da DataW não ser restrita ao idioma Finlandês (e qualquer outro) ou especifico para uso em documentos de decisões judiciais, tornando a ferramenta demasiadamente nichada, diferentemente da API apresentada nesse trabalho, pensada para uso geral.

Comparando com a ferramenta de nome Web Anonymizer, o uso das dependências e ferramentas utilizadas para implementar esse projeto pode impactar no desempenho em máquinas menos potentes devido a JVM utilizada para rodar o Java, linguagem de programação escolhida para criar o serviço web. Outro aspecto a ser destacado é a utilização da API de outro projeto no qual foi baseada, utilizando-a para integrar a API Rest, cujo objetivo é implementar o máximo de *features* semelhante da aplicação original, tornando-a extremamente dependente dos métodos fornecidos. A DataW apesar de depender do Postgres Anonymizer, o projeto foi pensado para ser o mais enxuto e leve possível, além do fator configuração, que se deve a facilidade de configurar e rodar projetos em Node.js.

3. DEFINIÇÃO DO MÉTODO DE ANONIMIZAÇÃO

O método de generalização foi escolhido para implementar a *DataW*. Essa forma de anonimização é particularmente interessante, pois seus valores permanecem verdadeiros após a anonimização, ao mesmo tempo que evita a identificação de um indivíduo, permitindo seu uso para estatísticas. Outro motivo pelo qual generalização foi a minha escolha se deve as características dos demais métodos, que em sua maioria remove, esconde ou substitui os dados por algum outro valor, não havendo muito o que poderia ser explorado devido grande parte do trabalho ser resolvido pelo próprio PostgreSQL Anonymizer, com exceção da Generalização e Adição de Ruído, no qual o primeiro foi escolhido por sua facilidade de implementação.

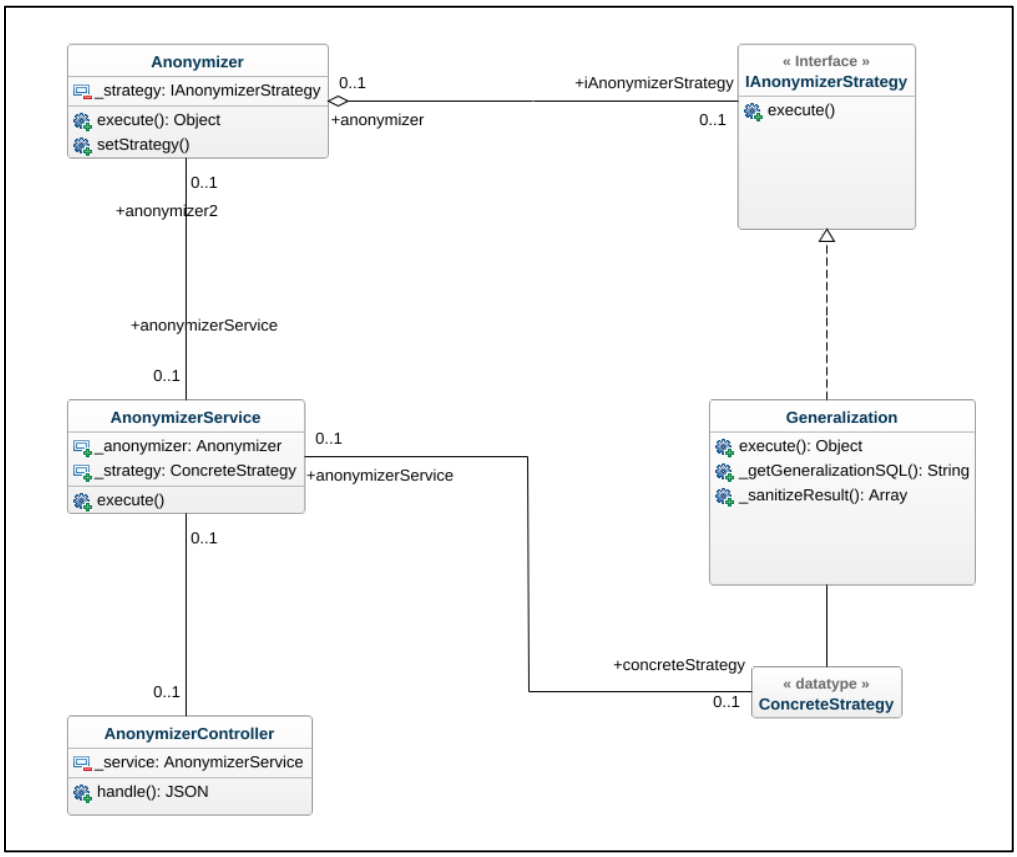
As tecnologias/ferramentas utilizadas consistem no uso do banco de dados relacional PostgreSQL em conjunto com uma extensão chamado PostgreSQL *Anonymizer*, que permite aplicar

regras de anonimização através de consultas SQL. Também é usada a linguagem de programação Javascript e um interpretador Javascript fora do navegador, conhecido como Node.js. Além das citadas acima, também há o uso de outras ferramentas, como o Express.js que é um *framework* para facilitar a criação de APIs e através disso disponibilizar um serviço onde os *endpoints* serão utilizados para enviar os dados a serem anonimizados, retornando o resultado na resposta da requisição.

4. IMPLEMENTAÇÃO DA DATAW PARA ANONIMIZAR DADOS COM BASE NA LGPD

Ao criar o serviço, foi decidido seu uso através de uma API seguindo o modelo MVC, separando-a em camadas que facilitem sua implementação e manutenção posterior. Também foi utilizado o *design pattern Strategy*, um padrão comportamental que define uma família de algoritmos, separando em classes cada um dos algoritmos (estratégias), permitindo um menor acoplamento. A Figura 1 apresenta o diagrama de classes da implementação da *DataW*.

Figura 1. Diagrama de classes da implementação do *DataW*.



Fonte: Elaborado pelo autor.

A primeira etapa do trabalho foi a implementação de um arquivo de configuração para variáveis de ambiente. Esse arquivo é usado para inserir valores sensíveis em diversas partes do código da API, pois não devem ser inseridos diretamente. Esse arquivo contém variáveis que determinam a porta de execução da API, Token JWT para restrição de rotas e até valores que permitem o acesso ao banco de dados. Na Figura 2 é mostrado o arquivo '.env' com as varáveis de ambiente.

Figura 2. Arquivo de variáveis de ambientes, para informações sensíveis no código.

```
DB_NAME=postgres
DB_USER=postgres
DB_PASSWORD=admin
DB_HOST=postgres-anonymizer
DB_PORT=5432
DB_DIALECT=postgres
NODE_ENV=development

PORT=3000

SECRET_JWT=secretExample
```

Fonte: Elaborado pelo autor.

A partir desse ponto começa a implementação do código Javascript (com Node.js), utilizando o *framework* Express.js para facilitar a construção do projeto. O primeiro passo é começar desenvolvendo o servidor HTTP que executará na porta definida previamente no arquivo de configuração, importado no começo do arquivo, seguido pelo módulo HTTP e o arquivo central da API, conforme mostrado na imagem abaixo, Figura 3. Caso a porta não seja passada na variável de ambiente ou por algum outro motivo a aplicação não consiga acessá-lo o será passado 3000 como valor padrão.

Figura 3. Código responsável por iniciar o servidor da API.

```
require('dotenv').config()
const http = require('http')
const app = require('./app')
const port = process.env.PORT || 3000

const server = http.createServer(app)

server.listen(port, console.log(`Server is running on port ${port}`))
```

Fonte: Elaborado pelo autor.

A seguir é criado o arquivo central da API, nomeado como ‘app.js’. Esse arquivo é responsável por centralizar configurações gerais da API. Essa configurações são:

- Importação do *framework* (Express.js), utilizado para criação de APIs Web utilizando a linguagem de programação Javascript, executado pelo Node.js. Disponível em https://expressjs.com/pt-br/;
- Módulo para tratamento para permitir requisições de diversas origens;
- *Middleware* de uso comuns das rotas da AP (endereço utilizado para realizar a requisição HTTP);
- Instância do Express.js a uma variável, contendo os métodos do *framework*, onde é possível definir as configurações gerais da API.

Na Figura 4 é possível ver o código dessa parte do projeto.

Figura 4. Arquivo onde é configurado o Express.js e suas dependências (rotas e *middlewares*)

```
1  const express = require('express')
2  const app = express()
3  const cors = require('cors')
4
5  const anonymizerRouter = require('./src/routes/anonymizer')
6
7  app.use(cors())
8  app.use(express.urlencoded({ extended: false }))
9  app.use(express.json())
10
11 app.use('/', anonymizerRouter)
12
13 module.exports = app
```

Fonte: Elaborado pelo autor.

A estrutura da API foi dividida em camadas, separando suas responsabilidades em diferentes módulos, como a camada de serviço (*service*), responsável pela regra de negócio. Nessa etapa é criada uma instancia da classe *Anonymizer* e a classe responsável pelo algoritmo de Generalização, método escolhido para anonimizar os dados e classe concreta do padrão *Strategy*.

As operações serão executadas e caso tudo ocorra conforme o esperado, os dados serão retornados dentro da classe do controlador (*controller*), porém caso tenha algum tipo de erro, será lançada uma exceção que deverá ser tratada em outro módulo. Na Figura 5 é possível ver o código da camada de serviço.

Figura 5. Camada de Serviço da API.

```
1  const Anonymizer = require('./../model/Anonymizer')
2  const Generalization = require('./../model/Generalization')
3
4  class AnonymizerService{
5      async execute(data){
6
7          const anonymizer = new Anonymizer()
8          anonymizer.strategy = new Generalization()
9
10         try {
11             //Executar operação de anonymização
12             const result = await anonymizer.execute(data)
13
14             return result
15         } catch (error) {
16             throw error
17         }
18     }
19 }
20
21 module.exports = AnonymizerService
```

Fonte: Elaborado pelo autor.

Já a camada de controle (*Controller*) contém os métodos que são utilizados quando alguma rota da API é chamada, ou seja, quando alguma requisição é realizada, o método da classe que estiver definido junto com uma determinada rota será executado quando sua rota for chamada, instanciando todas as dependências e executando-as, para em seguida retornar os dados em formato JSON (*JavaScript Object Notation*). Havendo algum erro, será retornado um JSON informando o *status code* e uma mensagem de erro. Segue abaixo, na Figura 6, o código da camada de controle.

Figura 6. Camada de controle da API.

```
const AnonymizerService = require('./../services/AnonymizerService')

class AnonymizerController{
    async handle(request, response){
        const { body } = request
        const service = new AnonymizerService()

        try {
            const result = await service.execute(body)
            return response.status(201).json(result)

        } catch (error) {
            response.status(500).json({
                status: 500,
                message: error.message
            })
        }
    }
}

module.exports = AnonymizerController
```

Fonte: Elaborado pelo autor.

Na imagem abaixo, Figura 7, é apresentado o módulo da API responsável por definir as rotas. É por meio dessas rotas que é possível consumir o serviço de anonimização através de requisições HTTP.

Cada conjunto de rotas são separadas em um único módulo, responsável por definir os *endpoints*, os caminhos de cada recurso, facilitando a divisão de responsabilidades e manutenção posterior, caso necessário.

Figura 7. Arquivo para configuração de rotas.

```
const express = require('express')
const router = express.Router()

const AnonymizerController = require('./../controllers/AnonymizerController')
const controller = new AnonymizerController()

router.post('/anonymizer/execute', controller.handle)

module.exports = router
```

Fonte: Elaborado pelo autor.

A seguir, como mostrado na Figura 8, foi utilizada uma classe abstrata como uma interface da Orientação a Objetos (uma vez que a linguagem de programação Javascript não tem suporte a interfaces), cujo o objetivo é definir um contrato com métodos que as classes descendentes (que herdam de IAnonymizerStrategy) devem implementar. A implementação da regra de negócio do método não é importante nessa etapa do projeto, pois os métodos herdados da “interface” pelas classes descendentes devem ser sobrescritos, implementando a própria lógica referente ao algoritmo de anonimização utilizado.

Figura 8. Classe usada como interface, definindo contratos através de métodos

```
1  const NotImplementedException = require('./../exception/NotImplemented')
2
3  class IAnonymizerStrategy{
4    async execute(){
5      throw new NotImplementedException()
6    }
7  }
8
9  module.exports = IAnonymizerStrategy
```

Fonte: Elaborado pelo autor.

O modelo de dados (*Model*) corresponde ao algoritmo/estratégia de anonimização do projeto, desempenhando também a classe concreta do padrão *Strategy* que herda da classe IAnonymizerStrategy tratada como uma interface, seguindo o contrato estabelecido por ela. A classe *Generalization* encapsula todas as operações (e lógica) referente a estratégia de generalização ao anonimizar os dados do usuário, fornecendo todos os métodos responsáveis por executar a regra de negócio, utilizando instruções SQL por meio de um ORM Javascript chamado Sequelize.js conforme apresentado na Figura 9, que foi utilizado para tornar o processo de acesso e manipulação do banco de dados mais simples.

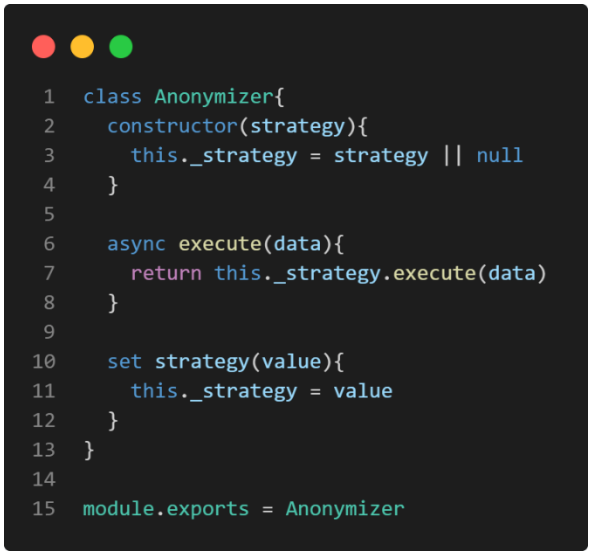
Figura 9. Classe com o algoritmo de generalização.

```
const IAnonymizerStrategy = require('./../interface/IAnonymizerStrategy')
const sequelize = require('./../database/sequelize')
const { QueryTypes } = require('sequelize')

class Generalization extends IAnonymizerStrategy{
  constructor(){
    super()
  }

  async execute(data){
    const result = {}

    const promiseArray = Object.entries(data).map(field => {
      return new Promise(async (resolve, reject) => {
        const [key, data] = field
        const { value, type, anon } = data

        try {
          const query = this._getGeneralizationSQL(type, value, anon)
          if (query == null) throw new Error('InvalidTypeError.')

          const [anonymizedData] = await sequelize.query(query, {
            type: QueryTypes.SELECT
          })

          const sanitizedResult = this._sanitizeResult(anonymizedData)

          result[key] = sanitizedResult
          resolve()
        } catch (error) {
          reject(error)
        }
      })
    })

    await Promise.all(promiseArray)

    return result
  }

  _getGeneralizationSQL(dataType, value, anon){

    const methods = {
      number: `SELECT anon.generalize_int4range(${value}, ${anon})`,
      daterange: `SELECT anon.generalize_daterange('${value}', '${anon}')`
    }

    const sql = methods[dataType] || null
    return sql
  }

  _sanitizeResult(anonymizedData){
    const values = Object.values(anonymizedData)[0].map(obj => obj.value);
    return values
  }
}

module.exports = Generalization
```

Fonte: Elaborado pelo autor.

A classe apresentada abaixo na Figura 10 apresenta a implementação da classe *Anonymizer*, que também desempenha um papel importante no padrão *Strategy*, responsável por definir o algoritmo usado na anonimização. Isso permite um baixo acoplamento no projeto, permitindo estender o comportamento com outras formas de anonimização de maneira simples, bastando atribuir a estratégia desejada ao atributo da classe.

Figura 10. Recebe o algoritmo de anonimização e o executa através dele.



```
class Anonymizer{
  constructor(strategy){
    this._strategy = strategy || null
  }

  async execute(data){
    return this._strategy.execute(data)
  }

  set strategy(value){
    this._strategy = value
  }
}

module.exports = Anonymizer
```

Fonte: Elaborado pelo autor.

Para acesso e operações no banco de dados foi utilizado o ORM Sequelize com o objetivo de facilitar a conexão com o banco de dados e a execução de instruções SQL. Na imagem abaixo, Figura 11, é possível ver algumas das variáveis de ambientes, mostradas anteriormente na Figura 2.

Figura 11. Configuração do ORM Sequelize.

```
const { Sequelize } = require('sequelize')

const sequelize = new Sequelize(process.env.DB_NAME, process.env.DB_USER, process.env.DB_PASSWORD, {
    host: process.env.DB_HOST,
    dialect: process.env.DB_DIALECT,
    port: process.env.DB_PORT,
    logging: false
})

module.exports = sequelize
```

Fonte: Elaborado pelo autor.

Realizar a anonimização por meio da API necessita de uma requisição HTTP ao seu *endpoint*, enviando um conjunto de dados em formato JSON, executando os processos descritos acima. Ao final do processo o mesmo conjunto de dados será retornado, porém com seus valores anonimizados.

Para realizar essa tarefa eu vou utilizar a ferramenta *Postman*, cujo é facilitar o teste, depuração e interação com APIs de forma fácil e eficiente, permitindo fazer chamadas HTTP para testar endpoints, visualizar respostas de forma fácil. O *Postman* oferece uma interface gráfica para facilitar o uso durante o desenvolvimento e o teste da API. Nas imagens abaixo, Figura 12 e 13, é mostrado os dados enviados e o resultado da anonimização, respectivamente.

Figura 12. Dados JSON usados como exemplo na anonimização.

```
{
  "zipcode": {
    "value": 76182413,
    "type": "number",
    "anon": 1000
  },
  "birthDate": {
    "value": "2010-12-29",
    "type": "daterange",
    "anon": "decade"
  }
}
```

Fonte: Elaborado pelo autor.

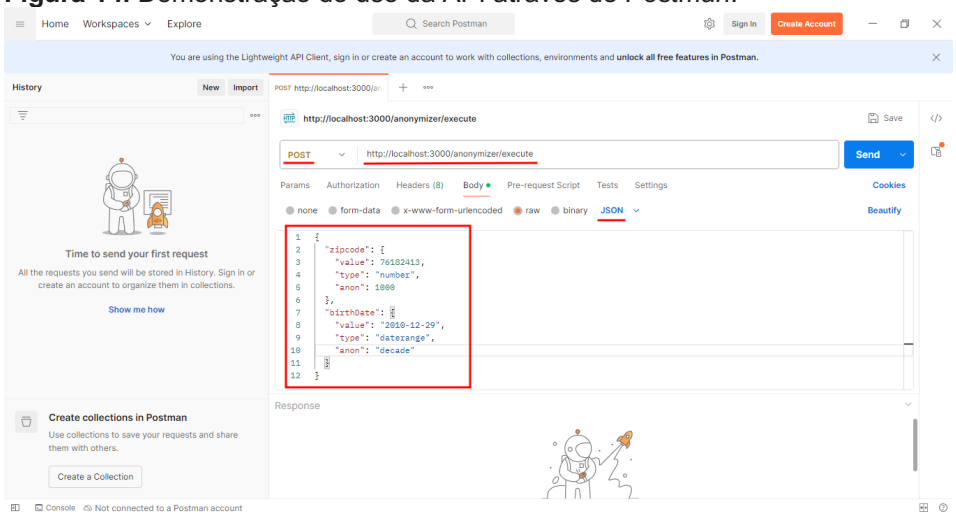
Figura 13. Resultado da anonimização dos dados fornecidos na imagem 12.

```
{
  "zipcode": [
    76182000,
    76183000
  ],
  "birthDate": [
    "2010-01-01",
    "2020-01-01"
  ]
}
```

Fonte: Elaborado pelo autor.

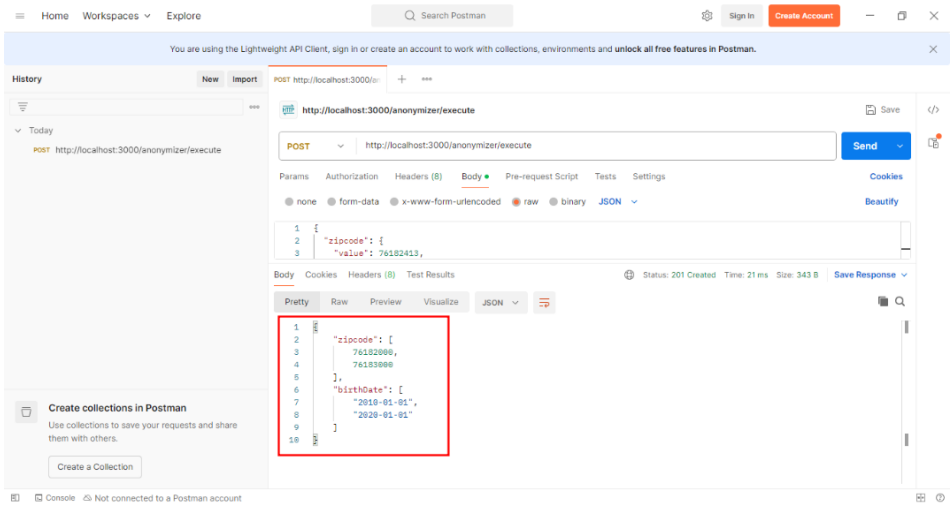
Para testar a API com o *Postman* é necessário configurá-lo com os valores corretos com que se deseja trabalhar. Logo, é necessário definir o endereço URL do local onde a API está sendo executada (incluindo sua porta e a rota utilizada para a anonimização), o método HTTP utilizado na requisição, o formato de dado a ser enviado, que nesse caso é JSON e os dados a serem anonimizados. Na imagem 14 é mostrado o *Postman* com as devidas configurações para testar a API e enviar os dados corretamente, enquanto na imagem 15 é mostrado o resultado da anonimização após a requisição realizada com a ferramenta.

Figura 14. Demonstração do uso da API através de *Postman*.



Fonte: Elaborado pelo autor.

Figura 15. Resultado da anonimização dos dados realizado no *Postman*.



Fonte: Elaborado pelo autor.

5. ESTUDOS DE CASOS

Estudo de Caso 1: Incidência hospitalar

Em um ambiente hospitalar ou um grupo de pesquisadores que necessitam investigar sintomas nos pacientes, no qual a faixa etária é um fator importante a se considerar, pois dependendo da idade dos indivíduos, os sintomas podem se manifestar em maior ou menor grau. Também há aqueles que apresentam uma melhor recuperação ou estão em maior vulnerabilidade. Nesse caso, a idade dos indivíduos seria um dado pessoal passível de aplicar generalização para diminuir sua precisão. Logo, ao aplicar generalização, o grupo que estivesse conduzindo a pesquisa poderia trabalhar com uma faixa de idade. Na imagem 16 e 17 é possível ver a entrada dos dados e o resultado, respectivamente.

Figura 16: Dados do estudo de caso 1.

Fonte: Elaborado pelo autor.

Figura 17: Resultado do estudo de caso 1.

Fonte: Elaborado pelo autor.

Estudo de Caso 2: Estudo Demográfico

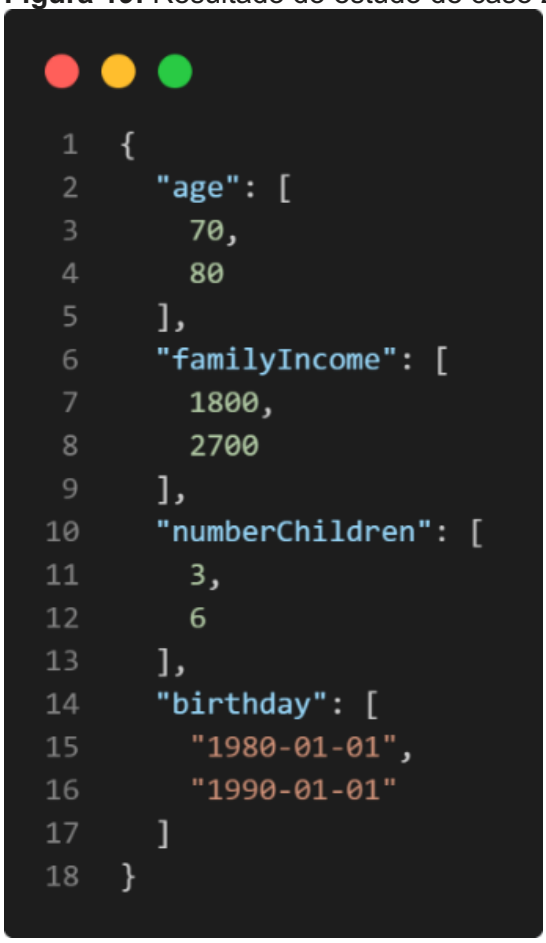
Caso fosse realizado um estudo demográfico, onde é necessário obter dados considerados pessoais que permitam identificar esses indivíduos de maneira pessoal, principalmente ao fazer o cruzamento dos mesmos. Dados como idade, renda familiar, quantidade de filhos e data de nascimento são dados que necessitam de anonimização, no qual a proposta da generalização atende muito bem, permitindo conduzir o estudo ao mesmo tempo que protege os dados dos participantes sem perder a capacidade de análise, como mostrado na figura de número 18 e 19.

Figura 18: Dados do estudo de caso 2.

```
{
  "age": {
    "value": 77,
    "type": "number",
    "anon": 10
  },
  "familyIncome": {
    "value": "1852",
    "type": "number",
    "anon": "900"
  },
  "numberChildren": {
    "value": 5,
    "type": "number",
    "anon": 3
  },
  "birthday": {
    "value": "1988-05-15",
    "type": "daterange",
    "anon": "decade"
  }
}
```

Fonte: Elaborado pelo autor.

Figura 19: Resultado do estudo de caso 2.



```
{
  "age": [
    70,
    80
  ],
  "familyIncome": [
    1800,
    2700
  ],
  "numberChildren": [
    3,
    6
  ],
  "birthday": [
    "1980-01-01",
    "1990-01-01"
  ]
}
```

Fonte: Elaborado pelo autor.

Estudo de Caso 3: Análise de Compras

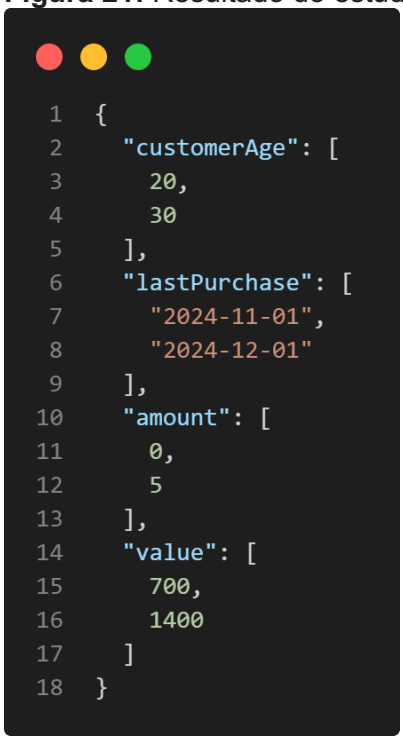
Um e-commerce pode ter a necessidade de realizar uma análise estatística para entender o perfil de compra de seus clientes, verificando quais produtos foram os mais comprados recentemente, entender a faixa etária dos indivíduos que fizeram determinada compra em um mês, associando os dados das vendas com a idade. Sendo assim, ao utilizar a generalização como forma anonimização seria possível proteger dados dos clientes a respeito de datas exatas das compras, quantidade de itens, valores das vendas, idade das pessoas ou mesmo impedir o cruzamento de dados que permita a identificação de um indivíduo de maneira pessoal. Ao fazer isso é possível analisar os padrões de compra ao mesmo tempo que mantém tudo em conformidade com a LGPD. Nas imagens de número 20 e 21 são mostrados os dados antes e depois da anonimizado, respectivamente.

Figura 20: Dados do estudo de caso 3.

```
{
  "customerAge": {
    "value": 28,
    "type": "number",
    "anon": 10
  },
  "lastPurchase": {
    "value": "2024-11-13",
    "type": "daterange",
    "anon": "month"
  },
  "amount": {
    "value": 3,
    "type": "number",
    "anon": 5
  },
  "value": {
    "value": "1200",
    "type": "number",
    "anon": "700"
  }
}
```

Fonte: Elaborado pelo autor.

Figura 21: Resultado do estudo de caso 3.



```
{
  "customerAge": [
    20,
    30
  ],
  "lastPurchase": [
    "2024-11-01",
    "2024-12-01"
  ],
  "amount": [
    0,
    5
  ],
  "value": [
    700,
    1400
  ]
}
```

Fonte: Elaborado pelo autor.

6. CONSIDERAÇÕES FINAIS

A implementação da LGPD cria oportunidades para que as organizações executem processos para controlar a inserção, alteração e anonimização de dados e a concessão de perfis de acessos. A generalização é um método usado para anonimizar dados permitindo que eles permaneçam verdadeiros ao mesmo tempo que evita a identificação de um indivíduo através daqueles dados, além da possibilidade de utilizá-los para estatísticas. Este trabalho apresentou a API web chamada *DataW*, que implementa o método de generalização. A *DataW* é um serviço que pode ser configurado e integrado através de um ponto de acesso, apenas enviando os dados, para em seguida receber o resultado sem se preocupar como está sendo processado.

Como a produção desse trabalho dedicou-se a utilizar apenas a generalização, o uso da ferramenta se torna limitado. Sendo assim, dependendo da natureza do dado e da regra de negócio, a *DataW* pode apresentar limitações, uma vez que poderia disponibilizar opções para anonimizar dados por meio de outras técnicas. Desta forma, uma ótima maneira de dar sequência seria implementar outras técnicas em trabalhos futuros para que a aplicação suporte a anonimização de dados para diferentes contextos ou talvez a criação de uma interface para o usuário interagir com a API de maneira visual, implementando uma aplicação completa.

7. REFERÊNCIAS

ARTTU, Oksanen Arttu. An Anonymization Tool for Open Data Publication of Legal Documents. 2022. University Of Helsinki, Helsinki – Finlândia, 2022.

BISSO, Rodrigo et al. Vazamentos de Dados: Histórico, Impacto Socioeconômico e as Novas Leis de Proteção de Dados. Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação, [S.l.], v. 3, n. 1, mar. 2020. ISSN 2446-7634. Disponível em: <https://revistas.setrem.com.br/index.php/reabtic/article/view/378>. Acesso em: 07 nov. 2024. doi: http://dx.doi.org/10.5281/zenodo.3833275.

BRASIL. Lei Geral de Proteção de Dados Pessoais (LGPD).Disponível em : https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm . Acesso em 05 de nov de 2024.

CAIRES, Kaio Alves Caires. A PROTEÇÃO DOS DADOS E A LGPD: DESAFIOS NA IMPLEMENTAÇÃO DA LGPD. 2023. Monografia (Bacharelado em Direito), Pontifícia Universidade Católica de Goiás, Goiânia – GO, 2023.

FERREIRA, Christophe da Silva Ferreira. A Web Anonymizer Platform for Datasets with Personal Information. 2017. Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos, Departamento de Ciência de Computadores, Porto – Portugal, 2017.

MONTEIRO, R. Existe um direito à explicação na Lei Geral de Proteção de Dados do Brasil?. INSTITUTO IGARAPÉ. 2018. Artigo Estratégico 39. Rio de Janeiro, Brasil.

POSTGRESQL. Various Masking Strategies. 2024a. Disponível em: <https://postgresql-anonymizer.readthedocs.io/en/latest/masking_functions/#write-your-own-masks>. Acesso em: 26 de out. de 2024.

POSTGRESQL. Anonymization & Data Masking for PostgreSQL. 2024b Disponível em https://postgresql-anonymizer.readthedocs.io/en/stable/ Acesso em 05 de nov de 2024.

SILVA, Lucas Henrique de Moura e. Escolha da Criptografia ideal e Anonimização de Dados Sensíveis Citados a Lei Geral de Proteção de Dados. 2020. Bacharelado em Engenharia de Computação, Centro Universitário de Anápolis - UniEvangélica, Anápolis – GO, 2020.

SOUSA, Bruno Henrique Assis de Sousa. DESMISTIFICANDO A LGPD: AVANÇOS E DESAFIOS PARA ADEQUAÇÃO DAS ORGANIZAÇÕES. 2023. Monografia (Bacharelado em Ciência da Computação), Universidade Federal de Campina Grande, Campina Grande – PB, 2023.

SOUZA, Kadmiel Duarte de Souza. LEI GERAL DE PROTEÇÃO DE DADOS (LGPD): IMPACTOS, DESAFIOS E PERSPECTIVAS NO CENÁRIO JURÍDICO E EMPRESARIAL DO BRASIL. 2024. Monografia (Bacharelado em Direito), Faculdade Fasipe Rondonópolis, Rondonópolis – MT, 2024.