RECAM
Revista Camalotes

SANGUE BOM: Desenvolvimento de um banco de dados

GOOD BLOOD: Development of a database

SANGUE BUENO: Desarrollo de uma base de datos

Gabriela Espinoza de Souza[1]

Luiz Felipe Paes de Souza[2]

Marcel Yassumoto[3]

Sérgio Adriano Sousa Santos[4]

Weverton Gomes[5]

Prof. Me. Ana Claudia De Oliveira Pedro Andrêo [6]

Prof. Me. Edilene Aparecida Veneruchi de Campos [7]

Prof. Esp. João Carlos Domingos [8]

RESUMO: O presente artigo apresenta uma pesquisa referente ao projeto "Sangue Bom", uma plataforma web com fins assistenciais para doação de sangue e medula óssea, facilitando e agilizando a comunicação entre Hemocentros, postos de coleta e os usuários. Sangue Bom é focado em ajudar pacientes que necessitam de doação de sangue e medula

[1] Gabriela Espinoza de Souza é acadêmico(a)a do curso de Graduação em Análise e Desenvolvimento de Sistemas. ORCID iD: https://orcid.org/0000-0003-2184-7828. E-mail: ge7384005@gmail.com.
[2] Luiz Felipe Paes de Souza é acadêmico(a)a do curso de Graduação em Análise e Desenvolvimento de Sistemas. ORCID iD: https://orcid.org/0000-0001-9475-7968. E-mail: felipepaes7653@gmail.com.
[3] Marcel Yassumoto é acadêmico(a)a do curso de Graduação em Análise e Desenvolvimento de Sistemas. ORCID iD: https://orcid.org/0000-0003-2184-7828. E-mail: marcel10042002@gmail.com.
[4] Sérgio Adriano Sousa Santos é acadêmico(a)a do curso de Graduação em Análise e Desenvolvimento de Sistemas. ORCID iD: https://orcid.org/0000-0002-9999-8936. E-mail: sergio.adriano.sousa94@gmail.com.
[5] Weverton Gomes é acadêmico(a)a do curso de Graduação em Análise e Desenvolvimento de Sistemas. ORCID iD: https://orcid.org/0000-0002-1067-6392. E-mail: weverton.g.n@gmail.com.
[6] Ana Claudia De Oliveira Pedro Andrêo é professora da Faculdade Insted. ORCID iD: https://orcid.org/0009-0000-6593-4259. E-mail: anaclaudia.andreo@insted.edu.br.
[7] Edilene Aparecida Veneruchi de Campos é professora da Faculdade Insted. ORCID iD: https://orcid.org/0000-0002-4427-608X. E-mail: edilene.veneruchi@insted.edu.br.
[8] João Carlos Domingos é professor da Faculdade Insted. ORCID iD: https://orcid.org/ 0009-0009-7364-8127. E-mail: joao.domingos@insted.edu.br.

óssea, notificando de forma automatizada os usuários doadores em situações que eles se enquadrem para realizar doações. Este artigo foi elaborado em Campo Grande, Mato Grosso do Sul, na graduação de Análise e Desenvolvimento de Sistemas. O presente artigo teve como objetivo entender o desenvolvimento de um banco de dados, utilizando ferramentas e metodologias já consolidadas e reconhecidas para esse fim. O banco de dados quando implementado para um sistema, é uma tecnologia que permite o armazenamento e a gestão organizada de dados estruturados ou não estruturados, sendo indiferente a plataforma ou o tipo de sistema disponibilizado. Além de armazenar informações, também facilita o gerenciamento e compreensão dos dados, tornando-se mais gerenciáveis e acessíveis de forma adequada. O MySQL foi escolhido para o trabalho devido à sua ampla integração com diversas linguagens de programação, licenciamento Open Source, facilidade de uso e gerenciamento, alta escalabilidade e configuração flexível.
PALAVRAS-CHAVE: Banco de dados. Modelagem de dados. MySQL.

ABSTRACT: The present article presents a research related to the "Sangue Bom" project, a web platform with charitable purposes for blood and bone marrow donation, facilitating and expediting communication between Blood Centers, collection points, and users. Sangue Bom is focused on assisting patients in need of blood and bone marrow donation by automatically notifying donor users in situations where they qualify to make donations. This article was developed in Campo Grande, Mato Grosso do Sul, during the undergraduate program in Analysis and Systems Development. The objective of this article was to understand the development of a database using established and recognized tools and methodologies for this purpose. When implemented for a system, a database is a technology that allows the organized storage and management of structured or unstructured data, regardless of the platform or type of system provided. In addition to storing information, it also facilitates data management and comprehension, making them more manageable and appropriately accessible. MySQL was chosen for the work due to its extensive integration with various programming languages, open-source licensing, ease of use and management, high scalability, and flexible configuration.
KEYWORDS: Database. Data Modeling. MySQL.

RESUMEN: El presente artículo presenta una investigación relacionada con el proyecto "Sangue Bom", una plataforma web con fines benéficos para la donación de sangre y médula ósea, que facilita y agiliza la comunicación entre Centros de Sangre, puntos de recolección y usuarios. Sangue Bom se enfoca en ayudar a pacientes que necesitan donaciones de sangre y médula ósea, notificando de manera automatizada a los usuarios donantes en situaciones en las que califican para realizar donaciones. Este artículo fue desarrollado en Campo Grande, Mato Grosso do Sul, durante el programa de licenciatura en Análisis y Desarrollo de Sistemas. El objetivo de este artículo fue comprender el desarrollo de una base de datos utilizando herramientas y metodologías ya consolidadas y reconocidas para este propósito. Cuando se implementa en un sistema, una base de datos es una tecnología que permite el almacenamiento y la gestión organizada de datos estructurados o no estructurados, independientemente de la plataforma o tipo de sistema proporcionado. Además de almacenar información, también facilita la gestión y comprensión de los datos, lo que los hace más manejables y accesibles de manera adecuada. Se eligió MySQL para el trabajo debido a su amplia integración con varios lenguajes de programación, licencia de código abierto, facilidad de uso y administración, alta escalabilidad y configuración flexible.

PALABRAS CLAVE: Banco de Datos. Modelato de Datos. MySQL.

INTRODUÇÃO

A necessidade de armazenar dados é crescente em nossa atualidade, parte de nossas atividades dependem deles para serem desenvolvidas. As organizações têm cada vez mais interesse em armazenar e gerenciar dados de seus usuários, para isso investem em tecnologias para obter, gerenciar e manter em segurança as informações geradas.

Neste sentido, o artigo buscar entender o que é um banco de dados, como trabalhar com modelagem de dados, como é realizada a sua construção, as ferramentas e metodologias necessárias para seu desenvolvimento, manutenção e segurança. Também entender as diferenças de banco de dados relacionais e o banco de dados não-relacionais, as diferentes modelagens de dados, sendo elas o modelo de dados conceitual, modelo de dados lógico e modelo de dados físicos, que, apesar de diferentes, se complementam. Uma boa modelagem dos dados pode evitar uma má implementação do banco de dados e o consumo desnecessário de recursos, facilitando assim sua manutenção.

Através da pesquisa exploratória buscou-se apoio teórico para o desenvolvimento, na prática, do banco de dados do projeto "Sangue Bom". A escolha para utilização do *MySQL* dá-se por ser o mais popular banco de dados *open source* do mundo, oferecendo consistência, alta performance, confiabilidade e fácil utilização. Optou-se também pela utilização da biblioteca chamada "*Sequelize*" pelo seu método de conectar os bancos ao código do sistema.

Com método "*define*" criaram-se as tabelas de dados, com apresentação de exemplo prático desenvolvido para o presente artigo, tendo-se apresentação final da página de cadastro dos usuários do site do projeto "Sangue Bom", através do formulário com acesso através do clique no botão "Cadastrar", capturando as informações nome, CPF, e-mail, cidade, e demais informações de interesse para organização.

DESENVOLVIMENTO

Atualmente, os bancos de dados são o bem mais precioso de uma organização, sendo o coração de seus sistemas computacionais. Seu crescimento se deu com a revolução da internet nos anos 1990, onde os sistemas de

atendimento telefônicos automatizados foram convertidos em sistemas de bancos de dados em aplicações *web*.

A integração com *web* é tendência em *software* de banco de dados, oferecendo uma variedade de técnicas para disponibilização das informações que foram armazenadas no banco de dados por meio de uma conexão de intranet ou internet. Esta integração permite que diferentes usuários tenham acesso a informações a partir de um site na web. (MELLO; VICTÓRIA; NOWACZYK; MIGUEL, 1990). Um bom sistema de banco de dados deve evitar redundâncias, garantir sua segurança, fornecer independência e facilidade nos procedimentos de manutenção dos dados.

No banco de dados temos também a modelagem de dados dividida em três modelos seguidos pela seguinte ordem (FRANÇA; JÚNIOR, 2015):

- **Modelo de dados conceitual** – este se baseia no mais alto nível, sendo usada para envolver o cliente, seu foco não é a tecnologia, mas os aspectos de interesse do negócio do cliente. São mais fáceis de compreender por não envolver aplicações tecnológicas específicas ou limitações;
- **Modelo de dados lógico** – leva em consideração as limitações na implementação dos recursos adequados, utilizando-se de padrões, nomenclaturas, normalização, a integridade referencial, definindo assim as chaves primarias e estrangeiras. Esta deve ser criado com base nos modelos de dados criados no Modelo Conceitual;
- **Modelo físico de dados** – neste modelo é quando fazemos a modelagem física de um bando de dados. Neste momento é importante levar em consideração as limitações impostas pela SGBD, deve ter como base as modelagens de dados definidas no Modelo Lógico.
-

A modelagem de dados tem em consideração aspectos diversos, oferecendo possibilidades de compreensão de um problema e posteriormente propor o banco de dados mais adequado e consistente, assim atendendo as necessidades a longo prazo. Podemos afirmar que uma boa modelagem dos dados pode evitar uma má implementação do banco de dados e o consumo desnecessário de recursos, facilitando assim sua manutenção.

O banco de dados, quando implementado para um sistema, é uma tecnologia que permite o armazenamento e a gestão organizada de dados

estruturados ou não estruturados, sendo indiferente a plataforma ou o tipo de sistema disponibilizado.

A partir do momento em que se pretende iniciar a criação de um banco de dados, é indispensável estudar a fundo qual a sua definição e quais os tipos de bancos de dados existentes, para analisar qual o banco de dados mais apropriado para o projeto.

Iniciando pelo conceito, segundo (FRANÇA; JÚNIOR, 2015, p. 7) um banco de dados pode ser visto como uma coleção de dados relacionados, que devem estar organizados para facilitar a busca e atualização desses dados. Os dados devem ter significado implícito e se referirem a fatos.

O armazenamento de dados é imprescindível em nossa atualidade, tendo em conta que a maioria de nossas atividades envolvem diretamente a utilização de uma base de dados, principalmente em organizações e empresas. Constatada a sua importância, a seguir serão apresentados os conceitos fundamentais do banco de dados, como armazenar e fazer a sua gestão.

Os bancos de dados são conjuntos de dados interligados e organizados entre si, sendo utilizados para o fornecimento de informações. (DATE, 2004), afirma que dados são formas primárias e brutas, por vezes não fazendo sentido isoladamente. Já as informações, são agrupamentos de dados organizados fazendo sentido e gerando conhecimento.

Com a evolução das novas tecnologias, os bancos de dados transformaram o armazenamento em *software*, facilitando o acesso, a busca e a pesquisa de informações dentro das organizações, mudando a forma de tomada de decisões. Um banco de dados é formado por quatro componentes, sendo estes: *hardware*, *software*, dados e usuários.

Pode-se afirmar que existem duas categorias de banco de dados comentados por (NOLETO, 2020):

- Banco de dados relacionais – **SQL** (Structured Query Language);
- Banco de dados não-relacionais – **NoSQL**.

Os bancos de dados relacionais referem-se aos dados que são armazenados em colunas, nos quais a descrição se mantém em linhas e atributos, tendo como importância os pilares ACID (atomicidade, consistência, isolamento e durabilidade). Já o banco de dados não-relacional é uma alternativa ao modelo relacional, utilizado quando precisamos armazenar e trabalhar com dados que não são facilmente organizados em tabelas, como imagens, vídeos, gráficos, entre outros. Ele é projetado para lidar com grandes quantidades de dados não-estruturados e semi-estruturados, com flexibilidade e escalabilidade, geralmente sem seguir os princípios ACID.

O banco de dados relacional é amplamente utilizado para o armazenamento de dados em bancos de dados, trabalhando os dados em tabelas interligadas entre si. (SILBERSCHATZ; KORTH; SUDARSHAN, 1999). As informações são retidas em tabelas com objetivo de facilitar a sua manipulação nos bancos de dados. As colunas da tabela retêm os dados e em determinados campos, são armazenados os valores atribuídos.

O *MySQL* foi escolhido para o trabalho devido à sua ampla integração com diversas linguagens de programação, facilidade de uso e gerenciamento, recursos avançados de segurança e configuração flexível.

O gerenciador de banco de dados *MySQL* foi desenvolvido na Suécia por Allan Larson, David Axmark e o Filandes Michael Widenius. O Projeto teve início em 1980. O *MySQL* é um sistema de gerenciamento de base de dados relacionais, suporta linguagem estruturada de banco de dados *SQL*, sendo um dos SGDs mais utilizado no meio profissional, com mais de 5 milhões de contas ativas e conhecido mundialmente. (NEVE, RUAS, 2005). Destaca-se pela sua estabilidade e escalabilidade, com alto desempenho na recuperação e inserção de dados, facilitando o escalar e a manutenção das bases de dados, tornando mais fácil o gerenciamento quando combinado com diversos *softwares* que tornam mais fácil a sua utilização.

Seguindo com a pesquisa exploratória, optou-se pela utilização da biblioteca chamada "*Sequelize*" pelo seu método de conectar os bancos ao código do sistema, iniciando uma instância do banco de dados, passando o nome, usuário e senha.

> O Sequelize é um framework Node.js do tipo ORM (Object Relational Mapper). Suas funções são adotam o padrão Promises, que é uma implementação semântica para tratamento de funções assíncronas presente noECMAS40. Atualmente, ele está na versão 3.8.0 e possui funcionalidades para tratamento de transações, modelagem de tabelas, relacionamento de tabelas, replicação de bancos para modo de leitura e muito mais (PEREIRA1, 2016).

Este possui um poderoso mecanismo de migração, transformando um esquema já existente de banco de dados em uma versão nova, tendo também a capacidade de sincronização de banco de dados a estruturas do modelo utilizado. O "*Sequelize*" é um ORM (*Object/Relational Mapper*), suportando os dialetos *MariaDB*, *PostgreSQL*, *SQLite*, *MSSQL* e *MySQL*, que será utilizado no decorrer do artigo.

Seguimos com a criação de tabelas no banco, utilizando o "*Sequelize*" demonstrado na figura 1, por trabalhar com modelos ou "models", sendo que um modelo é uma abstração que representa uma tabela em seu banco de dados. O modelo informa ao "*Sequelize*" diversas informações sobre a entidade que ele representa, sendo alguma delas o nome da tabela no banco de dados e quais colunas ela possui (e seus tipos de dados).

Figura 1: Instância do banco de dados.

```
import { Sequelize } from "sequelize";

const db = new Sequelize("sanguebom",'Usuário' ,"Senha",{
    host: "localhost",
    dialect: "mysql"
})

export default db
```

Fonte: Elaborado pelos alunos.

Diante das constatações citadas anteriormente, seguimos com a elaboração do artigo, evidenciando as informações chaves no desenvolvendo de um o banco de dados para o projeto "Sangue Bom". Dando continuidade ao projeto, iniciamos com instância do banco de dados. Esta pode conter vários bancos de dados que foram criados pelos usuários, sendo possível o seu acesso através das mesmas ferramentas ou aplicativo do cliente.

Na sequência, foi utilizado o método "*define*" como na figura 2, para criação de uma nova tabela no banco de dados. Esse método recebe o nome da tabela e todos os atributos dela, bem como todos os campos que essa tabela vai possuir. Exemplo: nome, CPF, e-mail, cidade, entre outras informações que podem ser de interesse das instituições.

Figura 2: Método "*define*".

```
const UsuarioModel = db.define('Usuario', {
    usuarioID:{
        type: DataTypes.INTEGER,
        autoIncrement: true,
        allowNull: false,
        primaryKey: true,
    },
    nome:{
        type: DataTypes.STRING,
        allowNull: false,
    },
    telefone:{
        type: DataTypes.STRING,
        allowNull: false,
    },
    dataNascimento:{
        type: DataTypes.DATE,
        allowNull: false,
        validate:{
            isDate:true,
        }
    },
    cpf:{
        type: DataTypes.STRING,
        allowNull: false,
    },
    email:{
        type: DataTypes.STRING,
        allowNull: false,
        validate:{
            isEmail:true
        },
    },
    senha:{
        type: DataTypes.STRING,
        allowNull: false,
    },
    uf:{
        type: DataTypes.STRING,
        allowNull: false,
    },
```

Fonte: Elaborado pelos alunos.

Após criado a "*model*" foi utilizado o método "*sync*" do *Sequelize*, para poder seguir realmente com a criação da tabela, segundo (SEQUELIZE, 2022).

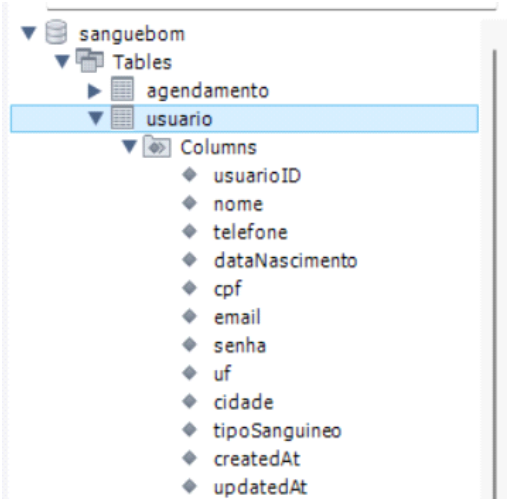
- **Model.sync()** - cria a tabela se ela não existir (e não faz nada se já existir);
- **Model.sync({ force: true })** - cria a tabela, descartando-a primeiro se ela já existir;
- **Model.sync({ alter: true })** - verifica qual é o estado atual da tabela no banco de dados (quais colunas ela possui, quais são seus tipos de dados, entre outros), e então realiza as alterações necessárias na tabela para que ela corresponda ao modelo.

Para a administração dos dados, foi utilizado o aplicativo "*MySQL Workbench*" ferramenta de gerenciamento de banco de dados *MySQL*. O uso do *MySQL* traz grandes benefícios para as empresas, sendo um deles a redução de custos com *softwares*, *hardwares* e mão de obra, característica fundamental apontada para escolha do banco de dados para suas aplicações. O *MySQL* é o sistema de gerenciamento de banco de dados mais utilizado pelas organizações. Destacado como um dos melhores gerenciadores de banco de dados, tem a versão com licença paga e livre, atendendo a necessidade de todos os portes de empresas, tendo grandes vantagens em relação aos seus concorrentes. (MILLANI, 2006). As figuras 3 e 4 mostram o resultado da organização das pastas, evidenciando a tabela criada no banco de dados:

Figura 3: Tabelas do banco de dados da plataforma.

Fonte: Elaborado pelos alunos.

Figura 4: Tabela usuário com todos os atributos colocados no código.



Fonte: Elaborado pelos alunos.

Outra vantagem bastante significativa do *MySQL*, é sua utilização em simultâneo por diversos usuários, com capacidade da manipulação de tabelas com mais de 50 mil registros, além de alta velocidade na execução dos comandos, com controle fácil e eficiente de privilégios dos usuários. O *MySQL* é poderoso, podendo executar bilhões de consultas diariamente, tem a capacidade de processar milhares de transações por minuto, ponto fundamental para as organizações que necessitam constantemente inserir ou consultar informações. (MILLANI, 2006).

Depois da "*model*" criada, tabela e banco conectado, chegou o momento de planejar as rotas que serão consumidas pelo *Front-end*, para acesso e manipulação das tabelas no banco. Para isso foi utilizado o conceito de *controller*: responsável por intermediar as requisições enviadas pelo *View* com as respostas fornecidas pelo *model*, processando os dados que os usuários informaram, e repassando para outras camadas.

Na *controller* do usuário estarão todos os métodos e rotas. Esta é responsável por exemplo: cadastrar um usuário (criar um usuário no banco), editar um usuário, deletar um usuário, consultar os usuários, entre outras funcionalidades.

> "A nossa agenda de contatos precisa ter como requisito mínimo um meio de permitir o usuário criar, listar, atualizar e excluir contatos. Esse é o conjunto clássico de funcionalidades, mais conhecido como CRUD (Create, Receive, Update e Delete). As rotas que utilizaremos para implementar o CRUD da agenda de contatos será aplicando o padrão de rotas REST. Esse

padrão consiste em criar rotas utilizando os principais métodos do HTTP" (GET, POST, PUT e DELETE) (PEREIRA2, 2016).

Esta camada é responsável pela intermediação das requisições fornecidas pelo *View* com respostas fornecidas pelo model. Um exemplo prático é mostrado na figura 5:

Figura 5: Rotas.

```
const UsuarioController = {

    async get(req, res) {
        try {
            const usuarios = await UsuarioModel.findAll();
            return usuarios.length > 0 ? res.status(200).json(usuarios) :
                res.status(204).send();

        } catch (error) {
            return res.json({ message: error.message })
        }
    },

    async getByUsuarioId(req, res) {
        const { usuarioID } = req.params
        const usuario = await UsuarioModel.findOne({
            where: {
                usuarioID: usuarioID,
            }
        })
        return usuario ? res.status(200).json(usuario) : res.status(204).send()
    },

    async post(req, res) {
        var usuario = req.body;
        usuario.senha =  await bcript.hash(usuario.senha, 8);
        console.log(req.body)
        try {
            UsuarioModel.create(usuario)
            return res.status(201).json(usuario);
        } catch (error) {
            console.log(error);
            return res.json({ message: error.message })
        }
    },

    async put(req, res){
        const { usuarioID } = req.params
        try {
            await UsuarioModel.update(req.body, { where: { usuarioID: usuarioID } });
```

Fonte: Elaborado pelos alunos.

Utiliza-se do método *post*, responsável por criar um registro de usuário no banco de dados, que são acessados através de uma rota especifica, exemplo: "/usuário". Para isso temos um arquivo chamado "*Routes*" que armazena as rotas que cada *controller* acessa, e cada método que será executado. É exemplificado, na figura 6, o trecho do código referente ao *route* usuário. O método *post* é utilizado para submeter um recurso específico, causando mudanças frequentes no estado do recurso. Já o *get* solicita as representações de recursos específicos, devendo retornar apenas dados.

Resumidamente, a requisição *post* é mais utilizada para enviar informações a serem processadas, já as requisições tipo *get*, são recomendadas para obtenção de dados de um determinado recurso.

Figura 6: Route usuário.

```
const UsuarioRoute  = express.Router();

UsuarioRoute.post("/usuario",UsuarioController.post);
UsuarioRoute.get("/usuario", UsuarioController.get);
UsuarioRoute.get("/usuario/:usuarioID", UsuarioController.getByUsuarioId);
UsuarioRoute.put("/usuario/:usuarioID", UsuarioController.put);
UsuarioRoute.delete("/usuario/:usuarioID", UsuarioController.delete);
UsuarioRoute.post("/login", UsuarioController.login)

export default UsuarioRoute;
```

Fonte: Elaborado pelos alunos.

Ou seja, para conseguir cadastrar um novo usuário, o nosso front-end terá que enviar os dados na rota "/usuário". Na figura 7 está a demonstração do servidor rodando na porta 8000 no arquivo index.js:

Figura 7: Rodando o Servidor.

```
const app = express()

app.use(cors())

app.use(express.json())

app.use(UsuarioRoute)


try{
    await db.authenticate()
    await db.sync()
    console.log("Conexão feita com sucesso!");
}catch(error){
    console.log(error);
}

app.listen(8000, ()=> console.log("Rodando na porta 8000"))
```

Fonte: Elaborado pelos alunos.

Seguindo para página de cadastro de usuário, o mesmo poderá preencher os dados do formulário e clica em "Cadastrar", ao clicar nesse botão, a função "*onFinish*", constando na figura 8, capturará todos os dados dos campos que ele digitou. A seguir envia estes dados para a rota *http://localhost:8000/usuario/*, com

isso localiza a rota "/usuário" e consequentemente cria um registro no banco de dados.

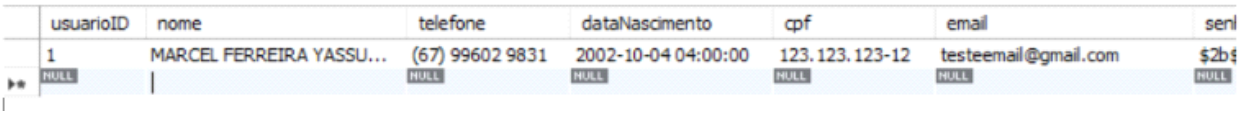
Figura 8: Exemplo de envio de dados para cadastro de usuário.

```
const URL = 'http://localhost:8000/usuario/'
const [loading, setLoading] = useState(false)


const onFinish = (values) => {
    setLoading(true)
    axios.post(URL, values)
    .then(()=>{
        setTimeout(() => {
            info()
            navigate("/")
        }, 1000)
    })

}
```

Fonte: Elaborado pelos alunos.

A função "*onFinish*" tem como objetivo retorna a chamada após o envio das informações do formulário, bem como a verificação dos dados. Na figura 9 segue o registro no banco de dados como exemplo:

Figura 9: Registro no banco de dados.



usuarioID	nome	telefone	dataNascimento	cpf	email	senl
1	MARCEL FERREIRA YASSU...	(67) 99602 9831	2002-10-04 04:00:00	123.123.123-12	testeemail@gmail.com	$2b$
NULL		NULL	NULL	NULL	NULL	NULL

Fonte: Elaborado pelos alunos.

CONCLUSÃO

O sistema de banco de dados é fundamental para armazenar e gerenciar informações de usuários de forma eficiente e segura, permitindo fácil acesso e atualização dos dados quando necessário. Segundo KENNET E JANE (2010), é importante que o banco de dados disponibilize interfaces para que programas e usuários externos possam acessá-lo, além de garantir a segurança das informações, *backup* para recuperação em caso de falhas e transações confiáveis.

Nos dias atuais, o armazenamento de dados é imprescindível, já que grande parte das atividades de empresas e indivíduos depende diretamente do acesso a uma base de dados organizada. O banco de dados conecta usuários e organizações para facilitar a busca e pesquisa de informações, contribuindo para o desenvolvimento de conhecimento para a organização.

No contexto do projeto "Sangue Bom", a criação de uma base de dados organizada e confiável é essencial para facilitar o acesso e a pesquisa de informações dos usuários, garantindo a agilidade e eficiência das atividades. Com a evolução da tecnologia, ferramentas e metodologias, é possível gerenciar os dados de forma segura e eficiente, garantindo a integridade e confiabilidade das informações. Em resumo, o banco de dados é um recurso valioso para o acesso à informação e para o sucesso das atividades de organizações e indivíduos. Neste projeto seu uso foi indispensável, para trazer mais agilidade nos agendamentos e no armazenamento de informações dos usuários, pois, com esse mecanismo de gerenciamento, pode-se tratar e transformar esses dados em informações seguras para a melhor forma de utilização dos usuários finais.

REFERÊNCIAS

DATE, *Introdução a Sistemas de Bancos de Dados*. 8ª Edição Americana. Rio de Janeiro: Elsevier Editora Ltda, 2004.

FRANÇA; JÚNIOR, Cicero Tadeu Pereira Lima França e Joaquim Celestino Júnior. *Computação:* Banco de dados 2ª ed. Fortaleza. 2015.

KENNET; JANE, Kennet Laudon e Jane Laudon. 2010. *Sistemas de Informações Gerenciais* 9ª ed: Pearson. p. 163,164.

MELLO, A.; VICTÓRIA JR., C.; NOWACZYK, D.; MIGUEL, W. *Computadores no seu futuro* 3º ed: banco de dados. São Paulo: Artmed, 1999. p. 192–200.

MILANI, André Milani, *Guia do programador*, São Paulo, Novatec editora, 2006.

NEVES; RUAS, Pedro Neves e Rui Ruas. 2005 *O GUIA PRÁTICO DO MySQL*, 1ª edição. Disponível em: http://www.centroatl.pt/titulos/tecnologias/imagens/excerto-e-book-caoguiapraticodomysql.pdf. Acesso em: 25 set. 2022.

NOLETO, Cairo Noleto. *Banco de dados:* tipos, o que é e suas diferenças!. 2022. Disponível em: https://blog.betrybe.com/tecnologia/bancos-de-dados/. Acesso em: 26 set. 2022.

PEREIRA1, Caio Ribeiro Pereira*. Construindo APIs REST com Node.js*. São Paulo. Casa do Código. 2016.

PEREIRA2, Caio Ribeiro Pereira. *Node.js: Aplicações web real-time com Node.js.* São Paulo. Casa do Código. 2016.

SEQUELIZE, *Sequelize:* Noções básicas do modelo. 2022. Disponível em: https://sequelize.org/docs/v6/core-concepts/model-basics/. Acesso em: 28 set. 2022.

SILBERSCHATZ; KORTH; SUDARSHAN, Abranham Silberschatz, Henry F Korth e S. Sudarshan. *Sistema de Banco de Dados.* 3™ ed. São Paulo: MAKRON Books, 1999.