UNIFEOB

ESCOLA DE NEG

2024

PROJETO INTEGRADO

UNIFEOB

CENTRO UNIVERSITÁRIO DA FUNDAÇÃO DE ENSINO OCTÁVIO BASTOS

ESCOLA DE NEGÓCIOS

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CIÊNCIA DA COMPUTAÇÃO

PROJETO INTEGRADO

SISTEMA DE GESTÃO E INTELIGÊNCIA DE NEGÓCIOS PARA ORGANIZAÇÕES SOCIAIS

Associação São Francisco de Assis

SÃO JOÃO DA BOA VISTA, SP

NOVEMBRO 2024

UNIFEOB

CENTRO UNIVERSITÁRIO DA FUNDAÇÃO DE ENSINO OCTÁVIO BASTOS

ESCOLA DE NEGÓCIOS

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CIÊNCIA DA COMPUTAÇÃO

PROJETO INTEGRADO

SISTEMA DE GESTÃO E INTELIGÊNCIA DE NEGÓCIOS PARA ORGANIZAÇÕES SOCIAIS

Associação São Francisco de Assis

MÓDULO COMPUTAÇÃO EM NUVEM

Estrutura de Dados – Prof. Marcelo Ciacco Almeida

Linguagem e Técnicas de Programação – Prof. Nivaldo de Andrade

Tópicos Avançados de Banco de Dados – Prof. Max Streicher Vallim

Computação em Nuvem – Prof. Rodrigo Marudi de Oliveira

Projeto de Computação em Nuvem – Profª. Mariângela Martimbianco Santos

Estudantes:

Enzo Dorigon Leandrini, RA 23000663
Felipe Fonseca Gimenes, RA 23000242
Marcos Vinícius Carvalho da Silva, RA 23000327
Rodrigo Espinosa Teixeira, RA 23000243

SÃO JOÃO DA BOA VISTA, SP
NOVEMBRO 2024

SUMÁRIO

1. INTRODUÇÃO

O projeto foi desenvolvido para prototipar um sistema de gestão personalizado para a ONG São Francisco de Assis, de São João da Boa Vista, com o objetivo de facilitar o controle e acompanhamento de doações de forma segura e acessível. Esta iniciativa, integrada ao contexto de um 'Sistema de Gestão e Inteligência de Negócios para Organizações Sociais', visa alavancar práticas de Business Intelligence (BI) para apoiar a tomada de decisões estratégicas da organização.

Para atingir essa meta, o sistema utiliza uma API conectada ao banco de dados SQL, possibilitando a integração eficiente e segura das informações de doadores, doações e necessidades da ONG. Com uma arquitetura baseada em computação em nuvem, o sistema busca oferecer uma plataforma escalável e disponível, alinhada com os princípios de sustentabilidade. A interface web desenvolvida facilita o acesso dos administradores e doadores aos dados em tempo real, promovendo a transparência e o engajamento contínuo.

Essa iniciativa permite não apenas a aplicação prática de conceitos de Computação em Nuvem, mas também reforça o compromisso da equipe com o desenvolvimento de soluções que impactem positivamente a sociedade.

2. DESCRIÇÃO DA EMPRESA

A Associação São Francisco de Assis em São João da Boa Vista é uma entidade beneficente dedicada ao atendimento gratuito de pessoas com deficiência. Fundada em 29 de abril de 1992, a associação foi criada por pais e familiares que identificaram a falta de centros de reabilitação especializados na cidade. Com uma abordagem multidisciplinar, a instituição oferece serviços nas áreas de fisioterapia, psicologia, terapia ocupacional, e fonoaudiologia, além de apoio e orientação aos familiares dos atendidos. A associação já prestou assistência a cerca de 15% da população de São João da Boa Vista, sendo uma referência na região para famílias de baixa renda.

Suas operações são sustentadas por doações da comunidade e um apoio financeiro da prefeitura, No entanto, as despesas mensais da entidade somam aproximadamente mais do que o valor pago pela prefeitura, sendo o restante arrecadado por meio de campanhas de telemarketing e doações via PIX (CNPJ: 67.161.810/0001-09). O endereço da associação é Rua Augusto Caetano, 27

3. PROJETO INTEGRADO

Para construção da plataforma, uma variedade de conhecimentos da área de tecnologia e elaboração de software foram utilizados, os quais serão descritos nesta sessão.

3.1 TÓPICOS AVANÇADOS DE BANCO DE DADOS

A fim de escolhermos qual banco de dados utilizar em nosso projeto, foi pesquisado a diferença entre banco de dados relacionais “SQL” e não relacionais “NoSQL”, destacando suas características, vantagens e desvantagens.

Como a proposta é desenvolver um software onde o objetivo é receber doações de usuários e criar novos registros, a melhor escolha foi utilizar banco de dados relacionais, utilizando a linguagem SQL, através do software MySQL

Para a escolha do banco de dados utilizado neste projeto, foi investigada a diferença entre bancos de dados relacionais (SQL) e não relacionais (NoSQL). Cada um desses tipos possui características específicas que influenciam a forma como os dados são armazenados, manipulados e escalados. Bancos de dados SQL, como o MySQL, são amplamente utilizados devido à sua capacidade de manter uma estrutura rígida, segurança e integridade transacional, sendo adequados para aplicações que exigem consistência. Em contrapartida, bancos de dados NoSQL são preferidos em sistemas que demandam alta escalabilidade e flexibilidade no esquema de dados, como em grandes volumes de dados não estruturados.

Devido à necessidade do projeto de receber doações e registrar dados de forma organizada e segura, optou-se por utilizar um banco de dados relacional, com a linguagem SQL, através do software MySQL, que proporciona maior controle e facilidade de uso para operações transacionais básicas e consistentes.

"Bancos de dados relacionais garantem integridade e consistência por meio de estruturas rígidas, enquanto bancos NoSQL priorizam a flexibilidade e a escalabilidade para dados não estruturados" (ROCKETSEAT, 2024).”

3.1.1 MODELO LÓGICO

Modelo Lógico

O modelo lógico define a estrutura conceitual das tabelas users, donations, payments e expenses, estabelecendo como os dados estão organizados e relacionados. Cada tabela é identificada por uma chave primária (id), e o campo email em users é único, garantindo que não haja duplicatas.

1. **Tabela users:**
   - A tabela users armazena informações básicas dos usuários, incluindo um identificador único (id), name, email, password, e phone.
   - Restrições conceituais:
     - O campo email é único, garantindo que cada usuário seja identificado de maneira distinta pelo e-mail.
     - O id é a chave primária, usada para identificar exclusivamente cada usuário.
   - Conceito: Esta tabela é essencial para a identificação de usuários no sistema e é necessária para gerenciar acesso e autenticação.
2. **Tabela expenses:**
   - A tabela expenses armazena informações sobre despesas, incluindo uma descrição (description), valor monetário (amount), data (date), e categoria opcional (category).
   - Restrições conceituais:
     - O id serve como chave primária, garantindo que cada registro de despesa seja único.
   - Conceito: Esta tabela é destinada a armazenar e categorizar despesas, permitindo que se mantenha o controle financeiro detalhado.
3. **Tabela payments:**

- Descrição Conceitual: A tabela payments armazena dados relacionados a pagamentos feitos pelos usuários. Cada pagamento tem um user_id que o associa ao usuário específico na tabela users.
- Campos e Restrições Conceituais:
  - id: Identificador único do pagamento (chave primária).

- user_id: Refere-se ao usuário que fez o pagamento, criando uma relação com a tabela users.
- payment_id: Identificador do pagamento (pode ser usado para relacionar a um sistema externo).
- amount: Valor do pagamento.
- payment_method: Método de pagamento utilizado (ex.: cartão, boleto).
- status: Status do pagamento (ex.: concluído, pendente).
- created_at e updated_at: Campos de auditoria para controle de data de criação e última atualização.

4. Tabela donations:

- Descrição Conceitual: A tabela donations armazena doações feitas, incluindo informações do doador e status do pagamento.
- Campos e Restrições Conceituais:
  - id: Identificador único da doação (chave primária).
  - donor_name: Nome do doador.
  - amount: Valor da doação.
  - date: Data da doação.
  - payment_method: Método de pagamento utilizado.
  - payment_status: Status do pagamento (ex.: pendente, concluído).
  - payment_id: Identificador do pagamento relacionado, permitindo o cruzamento com sistemas de pagamento externos.
  - created_at e updated_at: Campos de auditoria para registrar a data de criação e última atualização do registro.

3.1.2 MODELO FÍSICO

Modelo Físico

No modelo físico, cada tabela é detalhada com tipos de dados específicos e restrições. Na tabela users por exemplo, o campo id é uma chave primária com auto incremento (INT auto_increment), e o campo email é marcado como UNIQUE. Na tabela expenses, os campos description, amount, date, e category foram definidos para atender a requisitos de armazenamento e precisão de dados. O campo amount é do tipo DECIMAL(10,2), garantindo duas casas decimais para valores monetários.

1. **Tabela users:**
   - **Colunas e tipos de dados:**
     - id: INT, NOT NULL, PRIMARY KEY, auto_increment
     - name: VARCHAR(255), NOT NULL
     - email: VARCHAR(255), NOT NULL, UNIQUE
     - password: VARCHAR(255), NOT NULL
     - phone: VARCHAR(20), pode ser NULL
   - **Restrições:**
     - Chave primária: id
     - Restrições de unicidade: email é único.
   - **Índices:**
     - O campo id possui um índice automático devido à chave primária, o que acelera buscas por usuários.
2. **Tabela expenses:**
   - **Colunas e tipos de dados:**
     - id: INT, NOT NULL, PRIMARY KEY, auto_increment
     - description: VARCHAR(255), NOT NULL
     - amount: DECIMAL(10,2), NOT NULL
     - date: DATE, NOT NULL
     - category: VARCHAR(50), pode ser NULL
   - **Restrições:**
     - Chave primária: id
   - **Índices e Considerações de Armazenamento:**
     - Cada campo é projetado para otimizar o armazenamento e permitir fácil consulta de registros financeiros.

3. Tabela payments

- **Estrutura**:
  - id: INT, NOT NULL, PRIMARY KEY, auto_increment
  - user_id: INT, NOT NULL, chave estrangeira (relacionamento com users)
  - payment_id: VARCHAR(255), NOT NULL
  - amount: DECIMAL(10,2), NOT NULL
  - payment_method: VARCHAR(50), NOT NULL
  - status: VARCHAR(50), NOT NULL

- created_at: TIMESTAMP, permite NULL, valor padrão CURRENT_TIMESTAMP
- updated_at: TIMESTAMP, permite NULL, atualiza automaticamente com CURRENT_TIMESTAMP ao fazer alterações

4. Tabela donations

- **Estrutura**:
  - id: INT, NOT NULL, PRIMARY KEY, auto_increment
  - donor_name: VARCHAR(255), NOT NULL
  - amount: DECIMAL(10,2), NOT NULL
  - date: DATE, NOT NULL
  - payment_method: VARCHAR(50), permite NULL
  - payment_status: VARCHAR(50), valor padrão pending
  - payment_id: VARCHAR(255), permite NULL
  - created_at: TIMESTAMP, permite NULL, valor padrão CURRENT_TIMESTAMP
  - updated_at: TIMESTAMP, permite NULL, atualizado automaticamente com CURRENT_TIMESTAMP em cada alteração

3.2 LINGUAGEM E TÉCNICAS DE PROGRAMAÇÃO

Através da inclusão da prototipagem, desde a criação do front-end até o back-end, o desenvolvimento apresentou diversos desafios. A integração entre o front-end e o back-end desempenha um papel crucial, permitindo a comunicação efetiva entre as interfaces visíveis aos usuários e a lógica que processa e armazena os dados no servidor. Esse processo é facilitado por APIs, que atuam como intermediárias, promovendo uma troca de dados eficiente e escalável entre as duas camadas. Assim, a colaboração entre as equipes de desenvolvimento front-end e back-end é essencial para garantir consistência e uma experiência de usuário intuitiva.

Segundo Awari (2023), essa integração é vital, pois permite o desenvolvimento de sistemas coesos, onde a parte visual e interativa (front-end) se comunica adequadamente com os dados e processos internos (back-end), promovendo eficiência e segurança. A adoção de

APIs, frameworks e boas práticas de colaboração entre as equipes torna o processo mais ágil e estruturado, possibilitando um desenvolvimento mais eficiente e seguro (AWARI, 2023)

3.2.1 APPLICATION PROGRAMMING INTERFACE (API) - BACK-END.

1. Arquitetura RESTful: A API segue a arquitetura RESTful, que permite que diferentes partes do sistema se comuniquem usando verbos HTTP (GET, POST, PUT, DELETE) e recursos representados por endpoints. Essa arquitetura facilita a escalabilidade e manutenção, já que cada recurso é independente e pode ser acessado de forma isolada. A separação dos controladores no diretório `controllers` (por exemplo, `DonationsController.js`, `ExpensesController.js`, `PaymentController.js`, etc.) é uma aplicação prática, onde cada controlador lida com um aspecto específico da API.

2. Autenticação e Autorização: A segurança dos dados sensíveis é fundamental, especialmente em organizações sociais que lidam com informações confidenciais. A pasta `middleware` contém módulos de autenticação (`Auth.js`, `AdminAuth.js`, etc.), que gerenciam o acesso aos dados, assegurando que apenas usuários autorizados possam acessar ou modificar certos recursos. Esses arquivos verificam se o usuário está autenticado e define permissões específicas para ações administrativas ou de usuário regular, contribuindo para a integridade e privacidade dos dados.

3. Integração com Business Intelligence: Uma API voltada para a gestão de organizações sociais integra endpoints de Business Intelligence para fornecer dados analíticos em tempo real. E apoiar a tomada de decisões estratégicas, endpoints em `controllers` como `DonationsController.js` e `ExpensesController.js` fornecem dados agregados, como o total de doações, servindo de base para insights de BI.

4. Boas Práticas de Desenvolvimento: Testes Automatizados: Arquivos como `testLogin.js` e `resetTestUser.js` mostram resultados de testes feitos para validar funcionalidades críticas da API. Testes são essenciais para assegurar que novos desenvolvimentos não quebrem funcionalidades existentes e para verificar se o sistema responde adequadamente a diferentes casos de uso.

Imagem em anexos ao final do PI

3.2.2 FRONT-END

Nesse projeto, optamos por utilizar apenas HTML, CSS e algumas funções de JavaScript, complementados com elementos do Bootstrap para agilizar o desenvolvimento de um front-end responsivo e visualmente agradável. Embora frameworks como Angular tenham sido considerados, não conseguimos integrá-lo à nossa solução final. Optando, assim, por tecnologias mais diretas e acessíveis para o desenvolvimento.

Essa escolha permitiu construir uma interface prática e funcional, mantendo o foco na experiência do usuário e na responsividade, através dos recursos oferecidos pelo Bootstrap. Dessa forma, pudemos entregar um sistema eficiente e acessível, sem comprometer a qualidade visual e a facilidade de navegação.

3.3 COMPUTAÇÃO EM NUVEM

A computação em nuvem, tecnologia em constante crescimento por permitir uma maior flexibilidade no armazenamento de dados, é uma forma de não depender somente de um servidor ou de um computador. Dessa forma, você pode usar vários servidores para armazenar seus dados e realizar tarefas específicas, tudo através da internet. Sendo assim, o projeto foi desenvolvido com a intenção de criar uma plataforma de gerenciamento de dados em nuvem, no qual, a computação em nuvem, permitiu que os recursos fossem dimensionados conforme a demanda, sem precisar de um investimento em hardware adicional.

> “A computação em nuvem permite que as organizações acessem e armazenem informações sem gerenciar os próprios dispositivos físicos ou infraestrutura de TI.”(Google Cloud (s.d.))

3.3.1 OBJETIVOS DO PROJETO DE CLOUD COMPUTING

Por meio da computação em nuvem, o projeto busca otimizar vários aspectos operacionais da instituição. Um dos principais objetivos é reduzir a complexidade dos processos internos, como o controle de pessoas dando aos administradores uma capacidade de agir de forma mais eficiente e ágil na gestão de dados.

O baixo custo é um objetivo crucial, visto que a instituição não possui renda própria e depende de verbas governamentais e doações, dito isto a computação em nuvem oferece uma solução econômica ao tirar a necessidade de se investir em uma infraestrutura física. Outro grande benefício deste serviço é sua escalabilidade, a capacidade de ajustar os recursos às necessidades, com isto a instituição poderá lidar com possíveis mudanças nas necessidades operacionais.

3.3.2 APLICABILIDADE E BENEFÍCIOS DA CLOUD COMPUTING NO PROJETO

A computação em nuvem foi aplicada no projeto de forma concreta, centralizando o armazenamento de dados financeiros e pessoais dos usuários em um ambiente seguro e escalável. Isso permite à instituição automatizar o registro e a categorização de despesas, além de gerar relatórios financeiros ágeis e detalhados. A escalabilidade da nuvem oferece flexibilidade para que o sistema se ajuste à medida que o número de usuários cresce, eliminando a necessidade de investimentos prévios em infraestrutura física e permitindo que a empresa pague apenas pelos recursos que realmente utiliza.

Entre os principais benefícios, a computação em nuvem reduz custos operacionais, pois dispensa a manutenção física de equipamentos, e oferece segurança robusta com criptografia e monitoramento contínuo, facilitando o cumprimento da LGPD. A nuvem também agiliza o desenvolvimento de novas funcionalidades e permite a integração com tecnologias avançadas, como inteligência artificial, promovendo inovação constante. Com isso, a instituição consegue um serviço mais eficiente e seguro, mas também atende às necessidades dos usuários de forma prática e personalizada.

3.3.3 VANTAGENS DA CLOUD COMPUTING

A adoção da computação para a instituição oferece inúmeras vantagens que podem atender as necessidades da mesma

Uma das principais vantagens da computação em nuvem é a escalabilidade dos recursos, fazendo com que se tenha uma melhor administração dos recursos tanto para cima quanto para baixo, oferecendo uma economia no valor final.

Outra vantagem é o acesso remoto e a colaboração, com isso temos o acesso aos recursos e dados de qualquer lugar com conexão à internet, facilitando a colaboração entre equipes geograficamente dispersas

Por fim, a questão da segurança não foi um problema visto que a maioria dos provedores de nuvem investem fortemente, oferecendo recursos como criptografia e controle de acesso, além de conformidade com regulamentações.

3.3.4 DESENVOLVIMENTO EM CLOUD COMPUTING

O modelo de Software como Serviço (SaaS) oferece uma maneira de acessar software e aplicativos hospedados remotamente, eliminando assim a necessidade de instalações locais, isso não só agiliza a administração, mas também garante atualizações automáticas. O modelo de infraestrutura como serviço (IaaS) serve como base para a infraestrutura, facilitando a virtualização de recursos como armazenamento, ele oferece uma visão mais detalhada do controle sobre a configuração.

Enquanto isso, o balanceamento de carga desempenha um papel fundamental no aprimoramento da otimização. Componentes como servidores virtuais, bancos de dados em nuvem e serviços de armazenamento serão essenciais na arquitetura de cloud computing, esses componentes juntos trabalham para criar uma infraestrutura flexível e escalável, o'que permitiu se ajustar a picos de demanda

3.3.5 ESCOLHA DO PROVEDOR DE NUVEM (GOOGLE CLOUD OU AWS)

A escolha entre Google Cloud ou AWS é uma decisão, que deve se levar em consideração vários fatores, como as características do projeto, as preferências do grupo, os custos e o suporte que cada uma oferece.

Numa análise com os integrantes do grupo, escolhemos a AWS como provedor de cloud computing, visto que é um serviço com grande reconhecimento no mercado, e que possuímos algum conhecimento graças ao curso oferecido pela universidade.

Além disso, a AWS oferece um suporte técnico robusto, proporcionando um ambiente de fácil resolução para lidar com desafios específicos do projeto e futuros aprimoramentos.

A escolha deste provedor também foi influenciada pela escalabilidade da plataforma e pela diversidade de serviços oferecidos. Essa flexibilidade permitiu que o grupo selecionasse e utilizasse soluções específicas para atender as necessidades da instituição, otimizando a segurança e o desempenho.

3.3.6 DESENVOLVIMENTO EM CLOUD COMPUTING

No decorrer do desenvolvimento do projeto em cloud computing, adotou-se um modelo híbrido de aplicação, combinando elementos de arquitetura como o Software como Serviço (SaaS) e Infraestrutura como Serviço (IaaS).

A escolha de SaaS fez com que pudéssemos nos concentrar no desenvolvimento da aplicação, já o IaaS nos ofereceu a possibilidade de gerenciar e escalar a infraestrutura conforme o necessário. Essa abordagem híbrida possibilitou otimizar os custos e garantir uma melhor eficiência.

O balanceamento de carga desempenha um papel fundamental na otimização dos servidores, visto que ele distribui igualmente o trabalho entre os servidores, ao direcionar o trabalho de forma equivalente evitamos a sobrecarga dos servidores, garantindo uma experiência consistente e fluida mesmo em altas demandas.

Os bancos de dados em nuvem garantem acesso seguro e distribuído às informações necessárias, os servidores virtuais garantem escalabilidade dinâmica em resposta a picos de demanda e os serviços de armazenamento em nuvem oferecem soluções eficazes para gerenciamento de dados. A virtualização permite a criação de ambientes virtuais, o que aumenta a flexibilidade e a eficiência. A automação torna os processos de implantação e gerenciamento mais fáceis, enquanto os serviços baseados em API facilitam a integração de componentes da infraestrutura em nuvem. Esses componentes juntos agem para a criação de uma infraestrutura ágil que pode ser adaptada às necessidades específicas do projeto

3.3.7 GOOGLE CLOUD ou AWS

Etapa 1: Configuração do Ambiente na AWS

1. **Acesso à AWS Academy**:
    - Acesse o painel da AWS Academy com a conta educacional.
    - Revise os tutoriais e materiais de apoio na aba de **Módulos** do curso para entender melhor como configurar os recursos.
2. **Criar uma Estimativa na AWS Pricing Calculator**:
    - Use a AWS Pricing Calculator para estimar o custo de cada recurso.
    - Adicione os serviços planejados (EC2, RDS, S3 e SNS, conforme a necessidade).
    - Estabeleça limites de uso de acordo com o crédito de **US$100**, monitorando o consumo estimado para não ultrapassar o orçamento.

Etapa 2: Configuração da Instância EC2 (para Hospedar a API e o Sistema Angular)

1. **Configurar a Instância EC2**:
    - Na **AWS Console**, selecione **Serviços > EC2 > Instâncias** e clique em "**Launch Instance**" para criar uma nova instância.
    - **Região**: Escolha a região mais próxima do público-alvo para reduzir a latência.
    - **Tipo de Instância**: Selecione "t2.micro" ou "t3.micro", elegíveis para o nível gratuito (ótimos para desenvolvimento e pequenos testes).
    - **Sistema Operacional**: Escolha Linux (como Ubuntu), que é mais econômico em termos de uso de recursos.
    - **Configuração de Segurança**:
        - Configure as **regras de segurança** para permitir acesso via HTTP e HTTPS (porta 80 e 443) para o front-end Angular e a API.
    - **Volume de Armazenamento**: Use um volume EBS de 8 GB.
    - **Configuração de Uso**: Defina até 750 horas mensais para não ultrapassar o limite gratuito.
2. **Instalar e Configurar o Servidor Angular e API**:
    - Após iniciar a instância, conecte-se a ela via SSH.
    - Instale o **Node.js** e configure o servidor Angular e a API para rodarem no servidor EC2.

Etapa 3: Configuração do Banco de Dados (RDS com MySQL)

1. **Criar uma Instância RDS**:
    - No console da AWS, selecione **Serviços > RDS** e clique em "**Create Database**".
    - **Escolher o Banco de Dados**: Selecione MySQL.
    - **Tipo de Instância**: Use o "db.t2.micro" para manter o uso dentro do nível gratuito.
    - **Armazenamento**: Defina 20 GB de armazenamento EBS (SSD).
    - **Segurança e Backup**: Configure a segurança para que a instância de banco de dados seja acessível somente pela instância EC2, garantindo maior segurança.
2. **Configurar as Regras de Acesso**:
    - Permita o acesso da instância EC2 à instância RDS configurando o **Security Group** para a rede privada da VPC.

Etapa 4: Configuração do Armazenamento de Arquivos (S3)

1. **Criar um Bucket S3**:
    - No console da AWS, selecione **Serviços > S3** e crie um novo bucket para armazenar os arquivos da aplicação (ex: imagens, arquivos de usuário).

- **Configuração de Armazenamento**: Defina a quantidade inicial em 5 GB para um projeto básico.
- **Política de Acesso**: Configure as permissões do bucket para permitir acesso público apenas aos arquivos que precisam ser acessados externamente.

Etapa 6: Monitoramento e Ajustes de Custos

1. **Configuração do CloudWatch**:
   - Use o **CloudWatch** para monitorar o uso e o desempenho dos recursos.
   - Configure alertas básicos para acompanhar o uso de CPU e memória, especialmente da instância EC2 e do RDS.

3.4 ESTRUTURA DE DADOS

A escolha e implementação de estruturas de dados são fundamentais para o desenvolvimento de sistemas robustos e escaláveis, garantindo o desempenho eficiente e a integridade dos dados. Neste projeto, as estruturas de dados foram organizadas para atender aos requisitos do sistema, considerando a modularidade, a segurança e a integração com serviços de nuvem. Nós conduzimos um levantamento de requisitos detalhado, que orienta a construção e a otimização do backend, garantindo que cada módulo - dos controladores de doações e despesas aos serviços de autenticação e envio de notificações - seja implementado de forma eficaz e integrada. Além disso, são analisadas boas práticas de desempenho, escalabilidade e manutenção, visando um sistema adaptável para demandas crescentes e requisitos de nuvem no futuro.

Segundo Awari (2023), a escolha e implementação de estruturas de dados são fundamentais para o desenvolvimento de sistemas robustos e escaláveis, garantindo desempenho eficiente e integridade dos dados

"As estruturas de dados são essenciais no desenvolvimento de software, pois otimizam o acesso e o processamento das informações, resultando em aplicações mais rápidas e eficientes" (dti digital, 2024).

3.4.1 LEVANTAMENTO DE REQUISITOS

1. Descrição dos Módulos do Sistema:

- O sistema backend é composto por diferentes módulos organizados para atender funcionalidades específicas do projeto. Cada módulo manipula dados de forma independente, garantindo a coesão e a modularidade do código.

- **Controllers**: Os controladores são responsáveis por gerenciar as operações e a lógica de cada funcionalidade do sistema:

- DonationsController: Gerencia as operações relacionadas a doações, como criação, atualização, listagem e exclusão de registros de doações.

- ExpensesController: Controla as despesas, oferecendo funcionalidades para manipulação e visualização dos dados de gastos.

- PaymentController: Lida com as operações de pagamento, integrando transações e controlando a entrada de recursos financeiros.

- RecoverPassController: Responsável pela recuperação de senhas, fornecendo uma interface para gerenciar as solicitações de redefinição de senha.

- UsersController: Gerencia as operações de usuários, como criação, autenticação e atualização de informações de perfil.

- **Models**: Os modelos representam a estrutura dos dados e mapeiam cada entidade no sistema:

- Donations: Define a estrutura dos dados de doação, como valor, data e origem da doação.

- Expenses: Representa as despesas com atributos como descrição, valor e data de cada gasto.

- Payment: Estrutura os dados relacionados aos pagamentos, incluindo informações como nome, data e valor.

- Users: Modela as informações de usuário, como nome, email, e permissões.

- **Middleware**: O middleware adiciona camadas de segurança e validações adicionais:

- AdminAuth e Auth: Implementam autenticação para garantir que apenas usuários autorizados acessem recursos específicos.

- AuthAlterPass: Middleware específico para redefinir senhas, protegendo o sistema de acessos não autorizados durante a alteração de credenciais.

- **Service**: Contém funcionalidades de suporte, como envio de SMS para notificações ou confirmação de ações:

- authCode e sendSms: São serviços responsáveis pela geração e envio de códigos de autenticação e de mensagens SMS, auxiliando na segurança e comunicação com o usuário.

2. **Escolha de Estruturas Específicas para Otimização de Dados**:

○ A estruturação dos dados no sistema é baseada na utilização de objetos JavaScript (JSON) e tabelas de banco de dados relacionais, que permitem organizar e acessar os dados de maneira eficiente.

○ Embora o projeto não utilize explicitamente filas, árvores ou grafos, foram adotadas boas práticas de manipulação de dados que otimizam o desempenho e a escalabilidade do sistema. Exemplos incluem:

■ **Objetos e Arrays**: Em operações como listagem de doações e despesas, utilizam-se arrays para manipulação de coleções de dados de forma sequencial, facilitando a interação e a apresentação ordenada.

■ **Mapeamento Relacional**: O uso de um banco de dados relacional (MySQL) permite uma organização eficiente dos dados, utilizando chaves primárias e estrangeiras para vincular diferentes entidades, como usuários, pagamentos e despesas, garantindo integridade e facilitando buscas complexas.

○ **Ajuste de Operações de CRUD (Create, Read, Update, Delete)**:

■ As operações de inserção e leitura são otimizadas com índices em colunas-chave no banco de dados, melhorando a velocidade de busca e consulta.

○ **Escalabilidade e Manutenção da Consistência**:

■ A arquitetura do sistema foi projetada para escalabilidade horizontal. Assim, o sistema pode distribuir a carga de requisições entre várias instâncias, caso necessário, sem comprometer a consistência dos dados.

■ A estrutura de dados foi pensada para permitir fácil adição de novos módulos ou modificações nos existentes, garantindo a flexibilidade e manutenção a longo prazo do projeto.

3. **Considerações de Desempenho e Complexidade Algorítmica**:

○ O sistema foi projetado para minimizar a complexidade algorítmica nas operações mais frequentes, mantendo tempos de resposta baixos e uso de recursos eficientes:

■ **Listagens e Filtragens**: A ordenação e filtragem de grandes volumes de dados (por exemplo, listas de doações ou despesas) são realizadas diretamente nas consultas SQL, o que permite que o banco de dados execute essas operações de forma otimizada, antes mesmo de os dados chegarem à API.

- **Envio de Notificações**: Embora não seja utilizada uma fila explícita, o envio de notificações via SMS (integrado com o Amazon SNS) foi configurado para envio assíncrono, permitindo que o sistema gerencie várias requisições de envio em paralelo, evitando bloqueios.
  - **Alocação de Memória e Tempo de Resposta**:
- O uso de middleware para autenticação e validação de usuários reduz a carga nas consultas, pois bloqueia acessos não autorizados antes que o sistema execute operações mais pesadas.
- A estrutura modular, dividida em controllers, models e services, facilita o isolamento de responsabilidades, o que permite que o sistema aloque memória de forma eficiente e responda rapidamente às requisições.

3.4.2 VALIDAÇÃO DOS REQUISITOS

1. Integração das Estruturas de Dados com o Sistema em Nuvem:

- A integração com a nuvem foi realizada através do Amazon SNS, que desempenha um papel essencial na recuperação de senha. Esse serviço permite o envio de notificações SMS, assegurando que o usuário receba um código ou link de recuperação diretamente no dispositivo móvel, aumentando a segurança e a acessibilidade.

- Embora o banco de dados MySQL esteja sendo executado localmente, o sistema foi estruturado para facilitar a migração futura para o Amazon RDS. Essa migração proporciona elasticidade, permitindo que o banco de dados escalar horizontalmente para suportar um número maior de transações e operações, especialmente útil em ambientes de produção com altos volumes de acessos.

2. Otimização de Armazenamento e Desempenho:

- A configuração local do MySQL proporciona tempos de resposta rápidos, uma vez que os dados estão próximos à aplicação. No entanto, à medida que o sistema cresce, a necessidade de um banco de dados em nuvem, como o Amazon RDS, se torna evidente para garantir a escalabilidade e o suporte a múltiplas conexões simultâneas.

- No sistema atual, a diferenciação entre admins e usuários comuns é tratada no backend com verificações de "role" (papel) para determinar qual dashboard e quais

permissões cada usuário possui. Isso reduz a necessidade de consultas adicionais, uma vez que a verificação ocorre de forma eficiente na camada de aplicação.

3. Escalabilidade e Capacidade de Lidar com Grandes Volumes de Dados:

- Apesar de o MySQL estar sendo executado localmente, o sistema foi projetado considerando uma futura necessidade de escalabilidade. Com a migração para o Amazon RDS, o banco de dados seria configurado para escalar automaticamente, ajustando-se a picos de acesso e garantindo a disponibilidade em momentos de alta demanda.

- A elasticidade proporcionada pelo RDS garantiria que, caso a quantidade de dados cresça (por exemplo, com o aumento de registros de usuários, doações e despesas), o sistema permaneça responsivo e com bom desempenho, alocando mais recursos automaticamente conforme necessário.

4. Garantia de Integridade, Disponibilidade e Eficiência:

- A solução atual foi projetada para garantir a integridade dos dados através de constraints (restrições) no MySQL, como chaves primárias e estrangeiras, que previnem inconsistências e mantêm a relação entre tabelas.

- Com a adoção de um sistema de banco de dados em nuvem, o sistema pode aproveitar os mecanismos de backup e recuperação automáticos do Amazon RDS, garantindo a disponibilidade contínua e a rápida recuperação em caso de falhas.

- Além do mais, a integração com o Amazon SNS para notificações SMS garante que operações críticas, como a recuperação de senha, ocorram de maneira rápida e eficiente, mantendo a experiência do usuário fluida.

3.5 CONTEÚDO DA FORMAÇÃO PARA A VIDA: ENFRENTANDO ESTEREÓTIPOS

A Formação para a Vida é um dos eixos do Projeto Pedagógico de Formação por Competências da UNIFEOB.

Esta parte do projeto está diretamente relacionada com a extensão universitária, ou seja, o objetivo é que seja aplicável e que tenha real utilidade para a sociedade, de um modo geral.

3.5.1 ENFRENTANDO ESTEREÓTIPOS

- **Tópico 1:** Estereótipo e convívio social :

  O tópico aborda a questão dos estereótipos no convívio social, especialmente no ambiente universitário. Ele explora como os estereótipos surgem, seus impactos e como enfrentá-los. Um exemplo prático é a forma como, no cotidiano universitário, a aparência ou comportamento de estudantes frequentemente é associada a determinados cursos ou áreas de atuação. Alunos de jaleco são facilmente identificados como pertencentes às áreas da saúde, enquanto aqueles vestidos mais formalmente são muitas vezes atribuídos aos cursos de Direito ou Administração.

  Além disso, o texto menciona como, na prática, estereótipos podem influenciar nossas decisões profissionais e sociais, como a ideia de que certos cursos ou carreiras são mais apropriados para um gênero do que para outro. Exemplo disso é a separação comum de cursos "para homens" ou "para mulheres", que influencia a escolha de carreira de muitos, sem que percebam que estão sendo afetados por esses preconceitos.

  Essa discussão tem relevância prática porque nos convida a refletir sobre nossas próprias percepções e a agir de maneira mais consciente em nossas interações diárias, buscando sempre valorizar a diversidade e evitar generalizações que possam gerar preconceitos ou exclusões.

- **Tópico 2:** Estereótipo e representação:

  Explora como os estereótipos são construídos a partir de generalizações superficiais e disseminados por meio de ações, pensamentos e imagens.

  O estereótipo é destacado como um julgamento raso que afeta não apenas indivíduos, mas também grupos inteiros, como os brasileiros, que muitas vezes são vistos de maneira simplista e estigmatizada no exterior.

  Esses preconceitos são perpetuados e muitas vezes internalizados, afetando a maneira como julgamos os outros e nós mesmos. O documento também toca na busca pelo corpo ideal e como a pressão por padrões de beleza irrealistas gera transtornos psicológicos, como baixa autoestima, bulimia e anorexia.

  A discussão final sugere que é necessário questionar esses padrões e estereótipos, promovendo uma aceitação mais saudável e realista de quem somos, em vez de seguir ideais impostos pela sociedade.

- **Tópico 3:** Troco likes: a idealização da vida na internet:

A construção de estereótipos e sua influência no convívio social, especialmente em ambientes como a universidade e as redes sociais. Um estereótipo é uma generalização que pode levar à exclusão, preconceito e pressão para seguir padrões. Isso é observado tanto no mundo físico quanto no virtual, onde a idealização da vida, especialmente nas redes sociais, promove comparações e gera insatisfação pessoal.

Um exemplo prático, é a influência das redes sociais na busca pela aceitação e aprovação através de curtidas, comentários e seguidores. Isso cria uma pressão para aparentar uma vida perfeita, como quando alguém sente a necessidade de registrar todos os momentos do dia para postar nas redes, gerando um ciclo de validação social. Esse comportamento pode ser verificado diariamente, ao observar pessoas nas redes sociais tentando moldar suas realidades para atender a expectativas irreais.

Este exemplo evidencia a importância de desconstruir estereótipos e viver de forma mais autêntica, reconhecendo que nossa identidade não deve ser pautada por padrões externos impostos pela sociedade ou pelo ambiente digital.

- **Tópico 4:** Convivendo com a diferença:

  Aborda a importância de compreender e superar os estereótipos para promover uma convivência saudável entre diferentes grupos sociais.

  Desde a infância, somos expostos a uma variedade de pessoas e realidades, e essa convivência ensina que as generalizações são prejudiciais. Estereótipos, como associar muçulmanos a terrorismo ou tratar os orientais como gênios da tecnologia, são exemplos de como crenças distorcidas criam preconceitos. O tópico também trata da diversidade regional do Brasil, destacando como certas características são atribuídas a diferentes partes do país, muitas vezes de forma equivocada e pejorativa.

  O combate ao bullying e a convivência entre diferentes gerações também são tratados, enfatizando a necessidade de empatia, diálogo e aceitação das diferenças. Ao respeitar as diferentes trajetórias de vida e competências, as gerações podem se complementar e colaborar de forma mais produtiva.

  Em resumo, conviver com a diferença exige a superação de preconceitos por meio do diálogo e da empatia, reconhecendo o valor único de cada indivíduo e sua história.

3.5.2 ESTUDANTES NA PRÁTICA

ENFRENTANDO ESTEREÓTIPOS

ESTEREÓTIPO E CONVÍVIO SOCIAL

Para construir relações saudáveis e inclusivas, é fundamental desafiar esses rótulos e valorizar as individualidades. O respeito e a empatia são chaves para um ambiente onde todos se sintam vistos e aceitos.

Estereótipos afetam nossas interações diárias, criando preconceitos e barreiras no convívio social

ESTEREÓTIPO E REPRESENTAÇÃO

Desafiar os estereótipos e buscar representações justas é essencial para criar uma sociedade mais igualitária e empática.

Os estereótipos são generalizações simplistas sobre grupos de pessoas, muitas vezes reforçando preconceitos.

TROCO LIKES: IDEALIZAÇÃO DA VIDA NA INTERNET

A busca incessante por curtidas cria uma falsa sensação de felicidade, onde cada post é cuidadosamente editado para mostrar uma versão idealizada de si. A pressão para manter essa imagem pode distorcer a realidade e gerar comparações nocivas

É essencial lembrar que a vida real vai além dos filtros e das aparências digitais, valorizando momentos genuínos e autênticos.

CONVIVENDO COM A DIFERENÇA

A verdadeira harmonia nasce da aceitação mútua, onde cada indivíduo é reconhecido em sua singularidade

Conviver com a diversidade é essencial para desconstruir estereótipos e construir uma sociedade mais inclusiva

4. CONCLUSÃO

Por fim, ao longo do projeto passamos por diversos desafios que tivemos que superar, manter o funcionamento da relação entre back-end e front-end foi o principal. Essa etapa exigiu uma boa organização e harmonia entre cada integrante da equipe. Entretanto, superados os obstáculos iniciais, conseguimos não só consolidar os conhecimentos adquiridos, mas também fortalecer nossa capacidade de trabalhar sob pressão e em equipe. A experiência de integrar diferentes tecnologias e plataformas, como os serviços da AWS e APIs em Node.js, nos proporcionou uma visão mais ampla do desenvolvimento de sistemas, além de aprimorar nossas habilidades técnicas e de comunicação. Embora a adaptação a novas ferramentas tenha sido desafiadora, foi fundamental para o crescimento individual e coletivo de cada membro da equipe.

Este projeto também nos ensinou a importância de ter uma comunicação clara e contínua com todas as partes envolvidas, especialmente com a ONG, para garantir que as expectativas estivessem alinhadas e que os resultados atendessem às necessidades reais da organização. O aprendizado adquirido ao longo dessa jornada certamente será um diferencial importante em nossas futuras iniciativas profissionais, reforçando a ideia de que a prática é essencial para a evolução no campo da tecnologia.

REFERÊNCIAS

AWARI. Estruturas de Dados: Importância e Implementação. Disponível em: https://www.awari.com.br/estruturas-de-dados. Acesso em: 14 nov. 2024.

AWARI. *Integração Front-End e Back-End: A chave para o sucesso no desenvolvimento web*. Disponível em: https://awari.com.br. Acesso em: 07 nov. 2024.

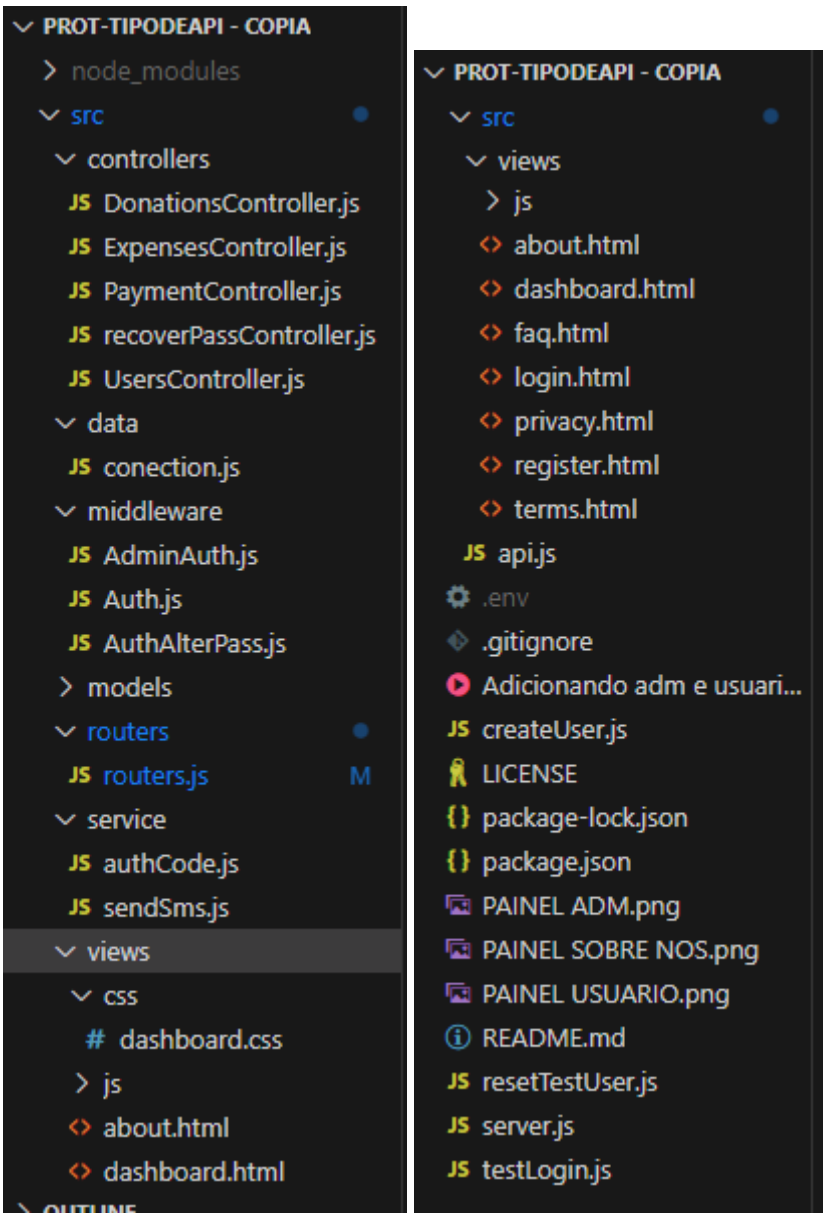
GOOGLE CLOUD. Vantagens e desvantagens da computação em nuvem. cloud.google.com, c2023. Disponível em: https://cloud.google.com/learn/advantages-of-cloud-computing?hl=pt-br. Acesso em: 18 out. 2024.

DIO. *O papel crucial das APIs: Back-end e Front-end unificados para maior eficiência*. Disponível em: Dio.me Acesso em: 07 nov. 2024.

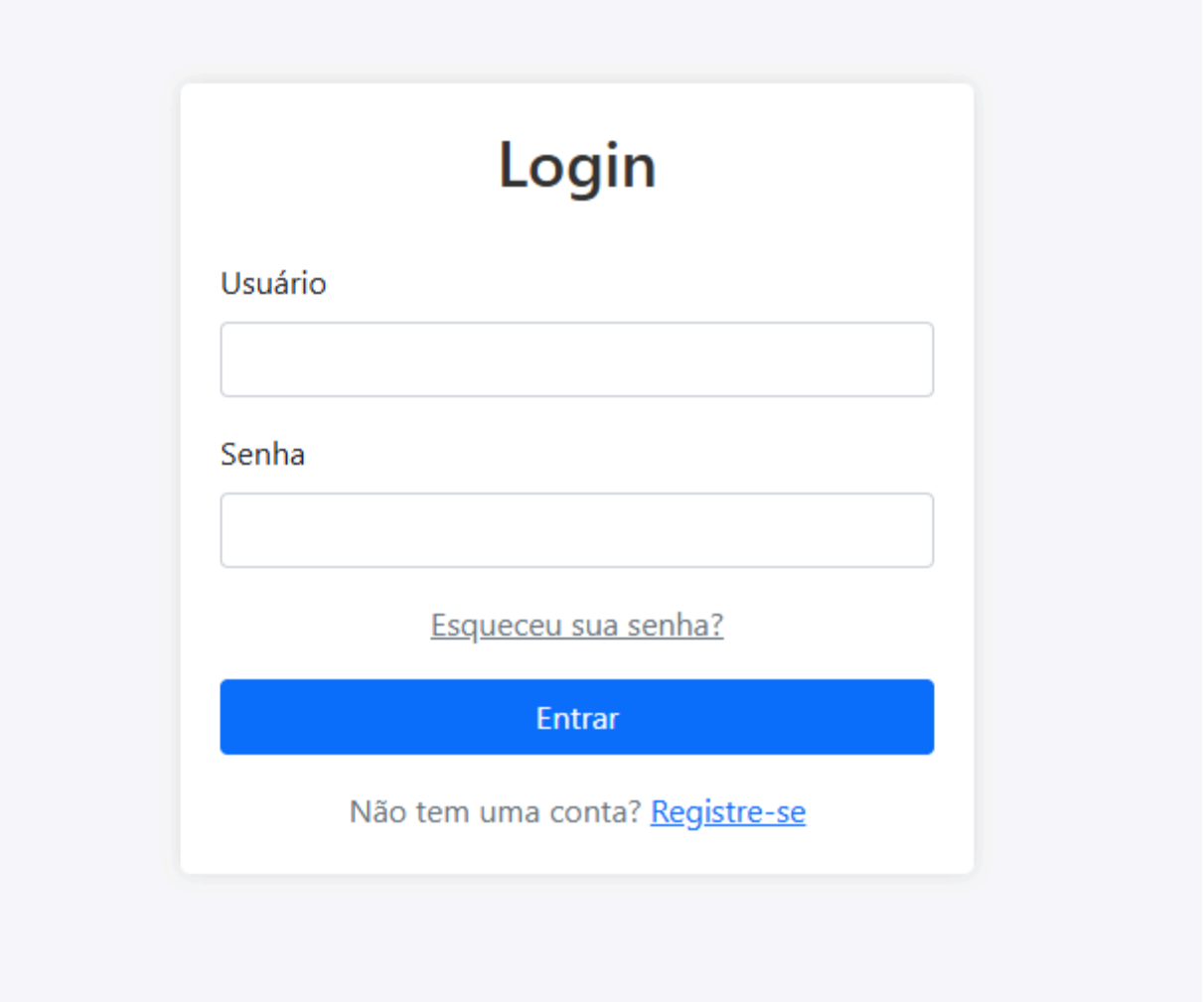
ROCKETSEAT. *Banco de Dados Relacional vs Não Relacional*. 15 fev. 2024. Disponível em: https://blog.rocketseat.com.br/banco-de-dados-relacional-nosql/. Acesso em: 7 nov. 2024.

dti digital. Estrutura de dados: o que é e qual a sua importância?. Disponível em: https://www.dtidigital.com.br/blog/estrutura-de-dados-importancia. Acesso em: 14 nov. 2024

ANEXOS

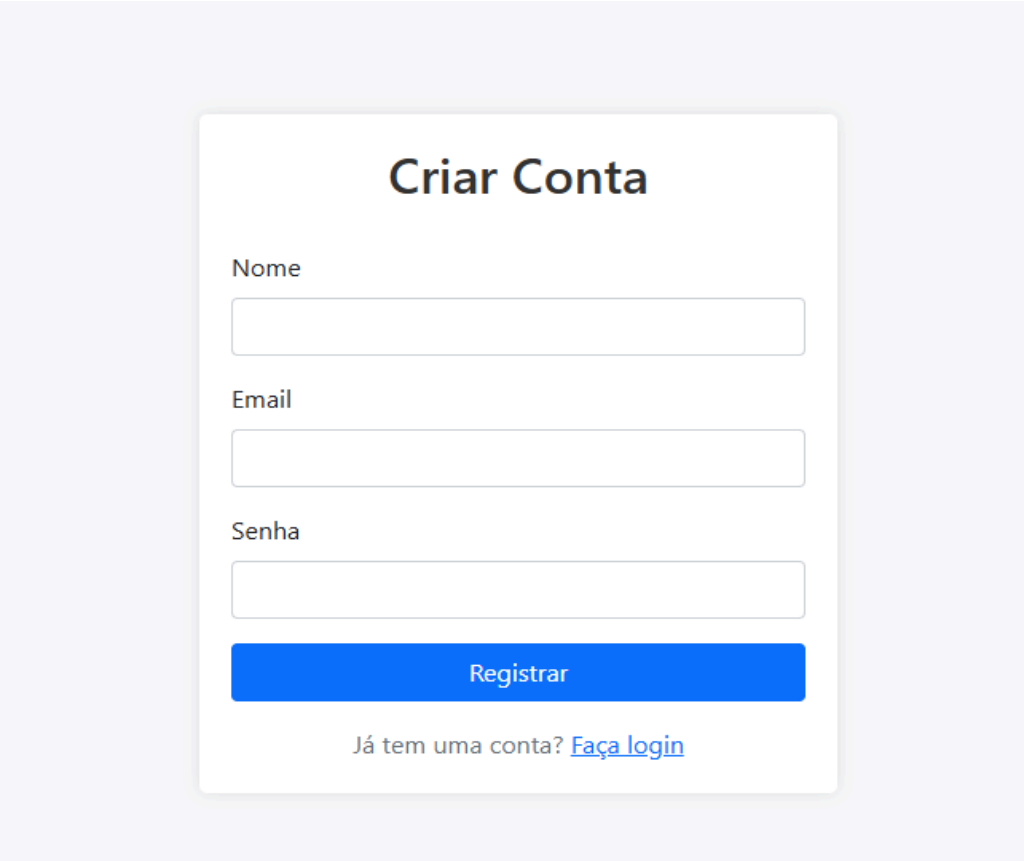

PROT-TIPODEAPI - COPIA
node_modules
src
controllers
DonationsController.js
ExpensesController.js
PaymentController.js
recoverPassController.js
UsersController.js
data
conection.js
middleware
AdminAuth.js
Auth.js
AuthAlterPass.js
models
routers
routers.js M
service
authCode.js
sendSms.js
views
css
dashboard.css
js
about.html
dashboard.html
PROT-TIPODEAPI - COPIA
src
views
js
about.html
dashboard.html
faq.html
login.html
privacy.html
register.html
terms.html
api.js
.env
.gitignore
Adicionando adm e usuari...
createUser.js
LICENSE
package-lock.json
package.json
PAINEL ADM.png
PAINEL SOBRE NOS.png
PAINEL USUARIO.png
README.md
resetTestUser.js
server.js
testLogin.js

Tela de Login



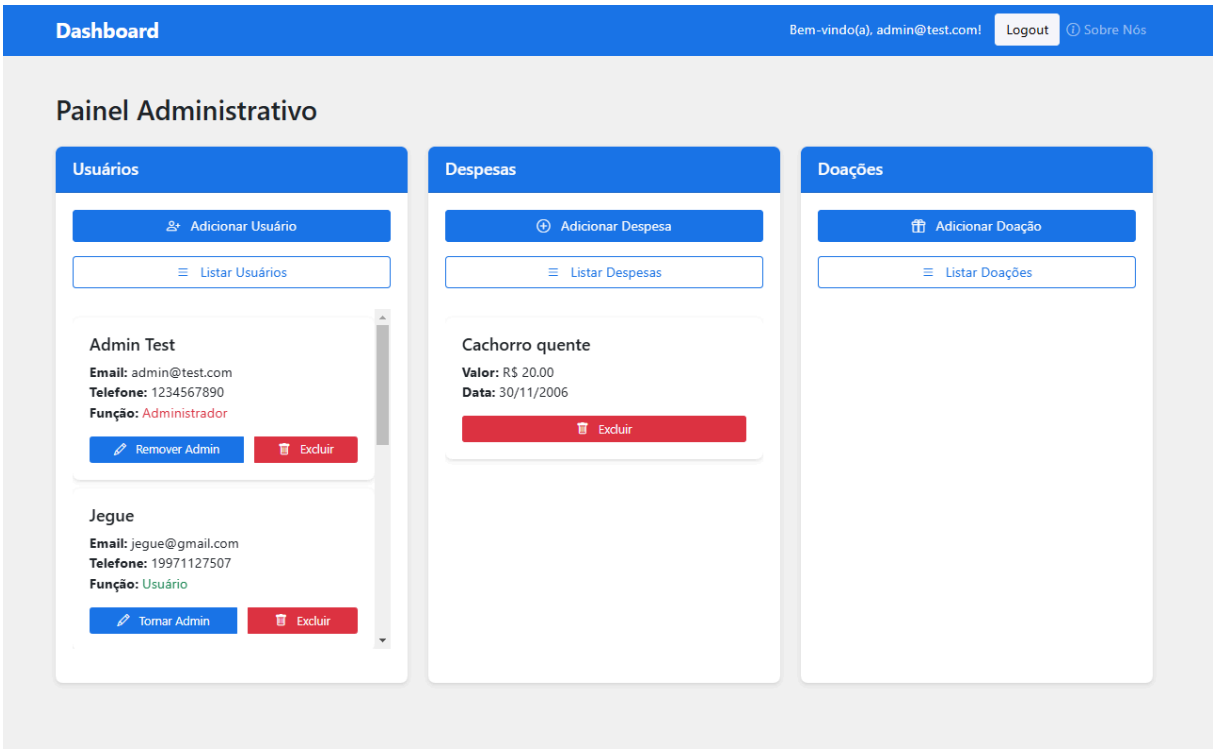
Fonte: Autoria própria

Tela de Registro



Fonte: Autoria própria

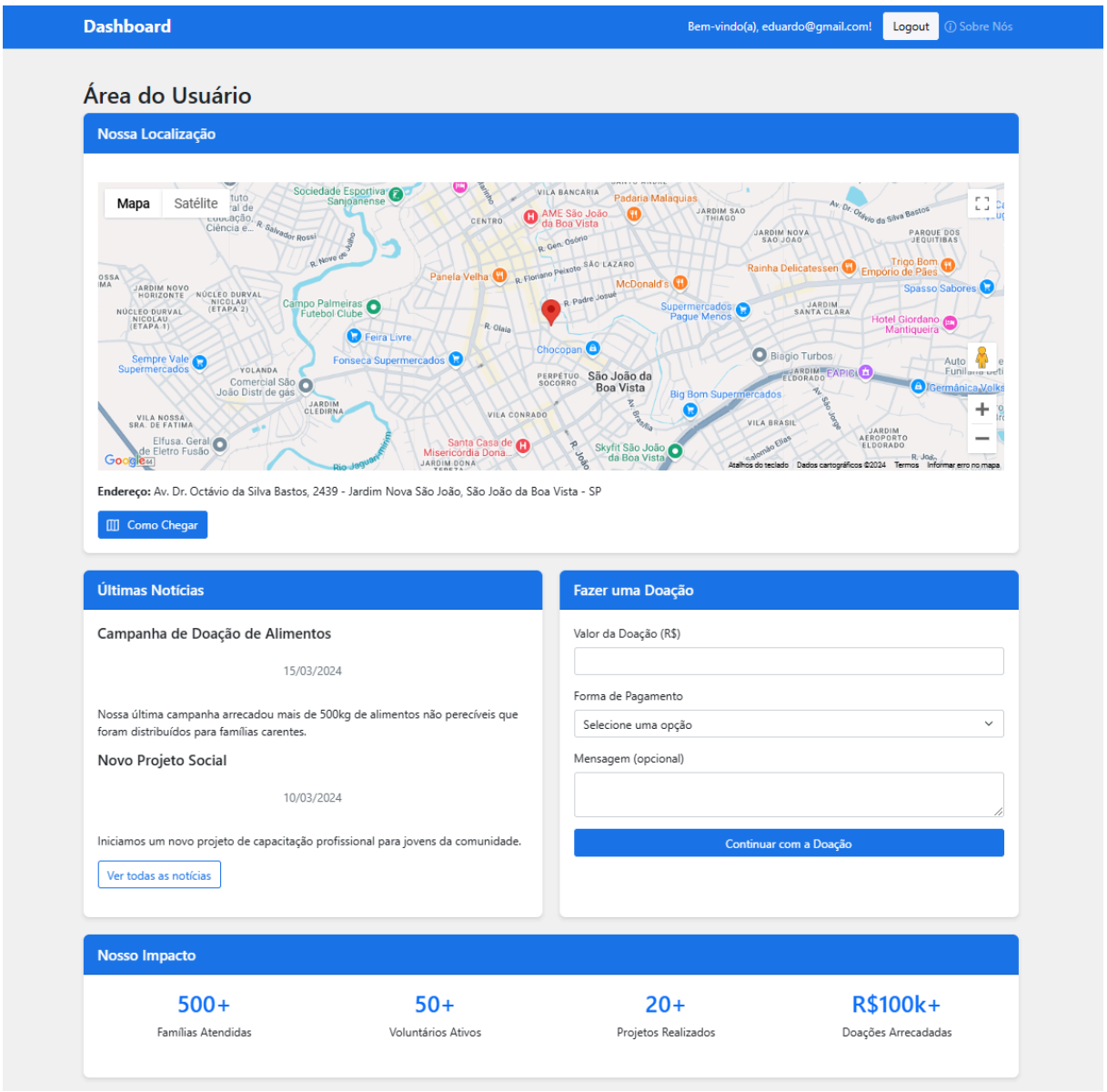
Painel Administrativo com integração ao banco de dados MySQL



Fonte: Autoria própria

Vídeo sobre as interfaces

Área do Usuário



Fonte: Autoria própria

Funcionamento da AWS SNS em relação ao sistema de Login do projeto


UNIFEOB