UNIFEOB
ESCOLA DE NEGÓCIOS

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

CAMID - Casa de Apoio ao Menor Irmã Dulce

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

<CAMID - Casa de Apoio ao Menor Irmã Dulce>

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:

Bruno Dotta Aleixo, RA 23000353
Carlos Gabriel dos Santos B, RA 23000679
Felipe Augusto Paulino de Morais , RA 23000426
Fernando Candido da Silva , RA 23000690
Kayky Rodrigo Graciano de Freitas, RA 23000421
João Gabriel O. Marcondes de Sozo , RA 23000103

SÃO JOÃO DA BOA VISTA, SP
NOVEMBRO 2024

SUMÁRIO

1. INTRODUÇÃO

Nos dias atuais, as ONGs enfrentam uma demanda crescente, com um número cada vez maior de pessoas precisando de apoio e assistência. Além dos desafios diários como a escassez de recursos, essas instituições se deparam com uma nova realidade influenciada por três fatores principais: a redefinição do papel do Estado, os efeitos da globalização e as mudanças no mercado.

O Estado, ao buscar novas formas de colaboração com o setor privado e as organizações não governamentais, reforça a importância da sociedade civil na oferta de serviços essenciais. A globalização, por sua vez, intensifica a conexão entre diferentes contextos, apresentando novos desafios sociais que requerem respostas rápidas e eficazes. Paralelamente, às transformações no mercado incentivam um maior compromisso com a responsabilidade social, tanto por parte das empresas quanto dos indivíduos, que se engajam cada vez mais em causas relevantes.

Como bem coloca Kotler (2012, p. 46), "a satisfação do cliente é o resultado da comparação entre suas expectativas e a percepção do desempenho do produto ou serviço". Reconhecendo essa premissa, a DevSete oferece soluções inovadoras, como um sistema de gerenciamento que ajuda as ONGs a organizarem melhor suas necessidades, facilitando para doadores identificarem de que maneira podem contribuir. Com essas ferramentas, as ONGs conseguem otimizar o uso de seus recursos e focar em apoiar aqueles que mais precisam. Onde agora entregaremos à empresa um sistema onde conseguiram cadastrar e ter um controle do estoque e do valor de doações em dinheiro recebidas. Com a implementação dele dentro da instituição pretendemos alcançar um número maior de doadores que realmente cheguem ao ato da doação em si.

2. DESCRIÇÃO DA EMPRESA

A Casa de Apoio ao Menor Irmã Dulce (Camid) é uma Organização Não Governamental (ONG) dedicada a acolher e abrigar crianças e adolescentes que se encontram em situações de risco. Desde sua fundação em 2001, a Camid tem se comprometido com o atendimento personalizado e individualizado, visando o desenvolvimento integral dos jovens sob seus cuidados. Este desenvolvimento abrange várias áreas essenciais, como:

- Pessoal: Promovendo a autoconfiança e habilidades pessoais;
- Social: Facilitando a integração e a convivência harmoniosa na comunidade;
- Afetiva: Proporcionando um ambiente acolhedor e emocionalmente seguro;
- Física: Garantindo condições adequadas de saúde e bem-estar;
- Cognitiva: Estimulando o aprendizado e o desenvolvimento intelectual.

Tem seu endereço na Rua Santa Teresinha, 350, bairro Jardim Dona Tereza em São João da Boa Vista, SP, CEP 13.871-140.

E-mail para contato: contato@camid.org.br.

Telefone para contato: (19) 3631-7183.

CNPJ: 04.810.265/0001-06.

A Camid se destaca pelo compromisso com o bem-estar e o desenvolvimento integral das crianças e adolescentes, atuando como um suporte crucial para aqueles que necessitam de cuidados e proteção.

3. PROJETO INTEGRADO

Com nosso projeto em mente, fomos adaptando as matérias que aprendemos neste módulo.

Primeiramente, foi implementado a atualização do nosso banco de dados, com aprendizado em aula utilizamos funções e vínculos para que se obtenha uma melhor visualização dos tópicos e tabelas utilizados.

Seguindo com base nos códigos, fomos introduzidos a novas implementações, onde foi realizado a utilização do Angular e de APIs, junto com professor Marcelo e Nivaldo. Nesta etapa foram feitas atualizações e modificações de nossos códigos e atualização do vínculo com o banco de dados.

Para finalizar, junto ao professor Marudi, fomos introduzidos a transferência de dados para a nuvem, verificamos os valores e métodos para realizar a utilização e testes da integração em nuvem.

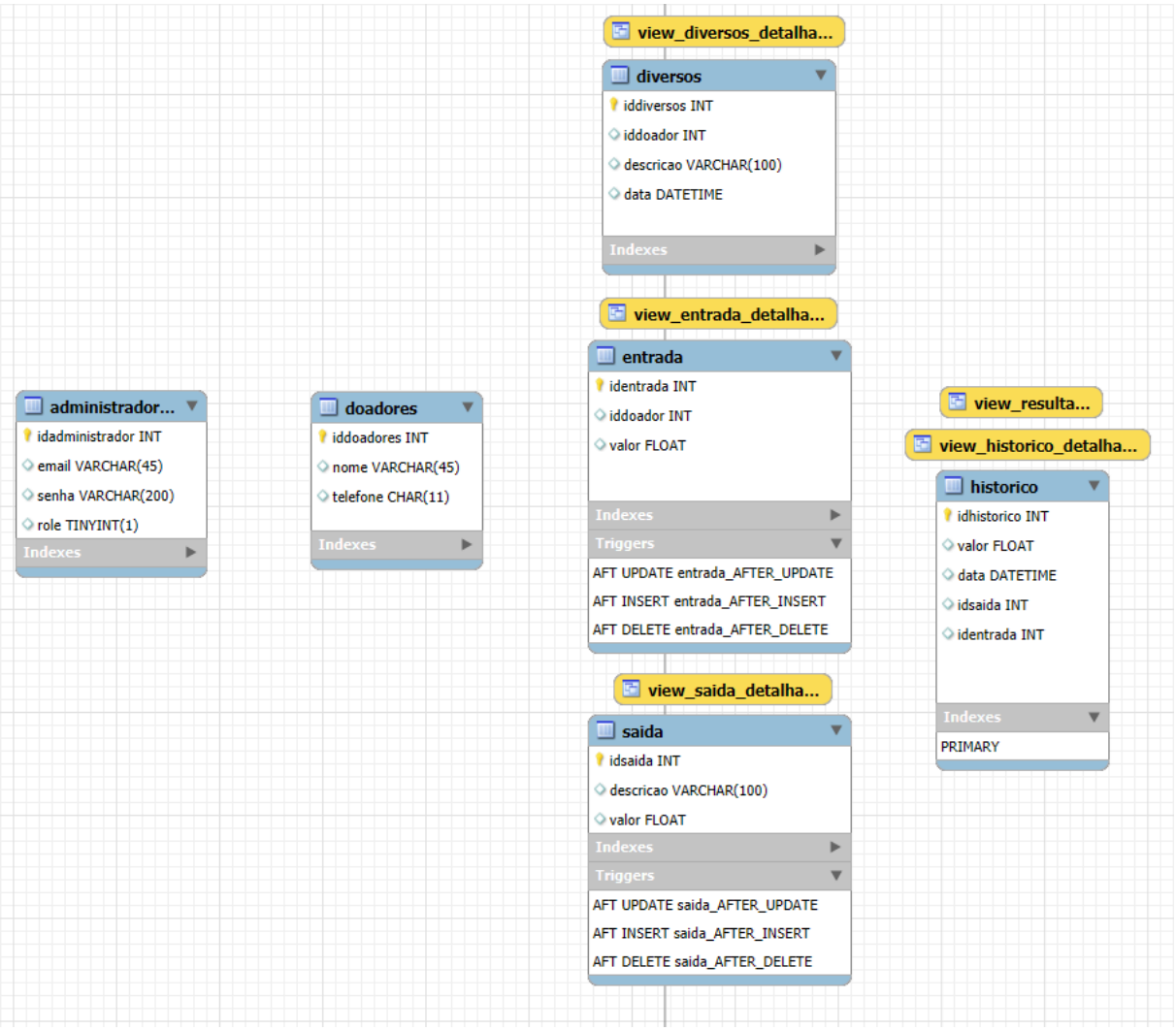
3.1 TÓPICOS AVANÇADOS DE BANCO DE DADOS

O banco de dados foi desenvolvido usando o MYSQL que é um banco relacional, pois a estruturação dele permite de forma mais fácil o relacionamento entre as tabelas, faz com que unir dados de diversas tabelas por meio dos "*JOINs*" seja uma tarefa simples, e também é levado em conta que MYSQL é gratuito, isso faz com que diminua os custos para a implementação total deste projeto para a Camid.

3.1.1 MODELO LÓGICO

O banco de dados desenvolvido, tem seu modelo lógico arquitetado da seguinte forma:

Modelo lógico do banco de dados da Camid:



Fonte: Autores(2024).

O Modelo possui seis tabelas, cada uma com uma finalidade específica:

Administradores: Onde serão cadastrados os operadores do sistema (Apenas usuários cadastrados nessa tabela tem permissão para fazer operações no banco de dados).

Doadores: Onde serão cadastrados os doadores da Camid.

Diversos: Onde serão cadastrados doações diferentes de moeda corrente (Por exemplo, doação de comida não perecível, ou doação de roupas).

Entrada: Onde serão cadastrados todas as doações em moeda corrente (seja PIX, depósito bancário, dinheiro físico).

Saída: Onde serão cadastrados todos os usos do dinheiro doado à Camid (seja por compras, investimentos, reparações).

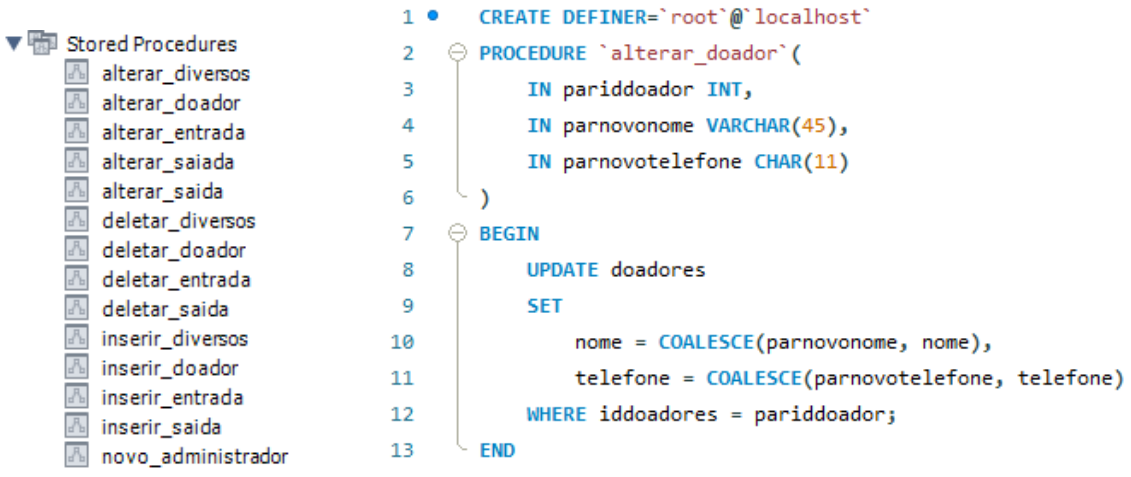
Histórico: Onde ficarão cadastrados todas as doações recebidas (diferentes de doações diversas) juntamente com todas as saídas.

3.1.2 MODELO FÍSICO

Para um bom funcionamento do banco de dados, foi necessário implementar *procedures*, *triggers* e *views*, essas ferramentas auxiliarão na utilização do banco.

Procedures, ou procedimentos são métodos desenvolvidos dentro do banco de dados para a execução de tarefas específicas, como realizar uma inserção, alteração ou exclusão de dados. Os procedimentos utilizados foram para inserção, alteração e exclusão das tabelas: administradores, doadores, diversos, entrada e saída.

A seguir a lista de procedimentos desenvolvida no banco de dados e à direita um exemplo de procedimento para realizar alteração de doador:



Fonte: Autores(2024). Fonte: Autores(2024).

Triggers ou gatilhos são alguns procedimentos também criados dentro do banco de dados que são ativados quando determinada ação acontece no banco, essa ação pode ser uma inserção, uma alteração ou uma exclusão em alguma tabela. Os gatilhos usados no banco de dados estão apenas nas tabelas "Entrada" e "Saída", funcionam da seguinte forma: sempre que ocorre uma inserção, atualização ou exclusão em alguma dessas duas tabelas, também é inserido, atualizado ou excluído esse mesmo dado na tabela histórico.

Gatilho acionado quando é inserido dados na tabela "Entrada" para inserir os mesmos dados na tabela "Histórico".

```
1  CREATE DEFINER=`root`@`localhost`
2  TRIGGER `entrada_AFTER_INSERT`
3  AFTER INSERT ON `entrada`
4  FOR EACH ROW
5  BEGIN
6      INSERT INTO historico(valor, data, identrada)
7      VALUES (NEW.valor, NOW(), NEW.identrada);
8  END
9
```

Fonte: Autores(2024).

Views ou visualizações são consultas em uma ou várias tabelas que exibe dados conforme definido na consulta. As visualizações são seguras, pois é possível evitar mostrar dados sensíveis do banco de dados, elas são práticas também, pois é possível elaborar uma consulta complexa com vários *joins* e guardar dentro da visualização, dessa forma, sempre que a executar, é como se estivesse escrevendo e executando uma grande linha de código complexo para retornar um *select* específico.

Foi elaborada uma visualização para cada tabela, elas tiveram como objetivo detalhar dados das tabelas, unir dados de tabelas relacionadas e também ocultar alguns dados sensíveis e, por fim, retornar esses dados ao usuário final. Ao fim deste documento, na seção Anexos, é possível visualizar as views implementadas e o exemplo de uma view bem estruturada (o exemplo a seguir) nos anexos 1 e 2.

Exemplo da visualização "Histórico detalhado" sendo executado:

idhistorico	identrada	idsaida	nome	descricao	valor	data
1	5	NULL	Carlos	NULL	199.9	2024-09-01 09:14:11
2	6	NULL	Fernando	NULL	92	2024-09-01 09:15:31
3	NULL	1	NULL	5kg de carne	148.7	2024-09-01 09:23:46
4	NULL	2	NULL	12L leite	50	2024-09-01 10:08:06
5	7	NULL	NULL	NULL	200	2024-09-20 21:29:47
6	NULL	3	NULL	40kg de arroz branco	170	2024-09-20 21:39:54
7	NULL	4	NULL	10kg de feijão	86	2024-10-14 21:15:10

Fonte: Autores(2024).

Visualizações elaboradas no banco de dados:

Fonte: Autores(2024).

Exemplo da visualização "Histórico detalhado" onde une doações recebidas, compras realizadas, com nomes de doadores e data e hora realizada.

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `view_historico_detalhado` AS
    SELECT
        `h`.`idhistorico` AS `idhistorico`,
        `e`.`identrada` AS `identrada`,
        `s`.`idsaida` AS `idsaida`,
        `d`.`nome` AS `nome`,
        `s`.`descricao` AS `descricao`,
        `h`.`valor` AS `valor`,
        `h`.`data` AS `data`
    FROM
        `historico` `h`
    LEFT JOIN `entrada` `e` ON `h`.`identrada` = `e`.`identrada`
    LEFT JOIN `saida` `s` ON `h`.`idsaida` = `s`.`idsaida`
    LEFT JOIN `doadores` `d` ON `e`.`iddoador` = `d`.`iddoadores`;

```

Fonte: Autores(2024).

3.2 LINGUAGEM E TÉCNICAS DE PROGRAMAÇÃO

A API foi estruturada de uma forma que impede que haja retrabalho em casos de mudança de banco de dados. O próximo tópico explicará com mais detalhes sobre a API, mas em resumo, os métodos da pasta *Controller* é chamado em solicitações do *front-end*, esse método cria um objeto utilizando um modelo da pasta *Models*, e o entrega para o *Repository*, só então o *Repository* se comunica com o banco de dados.

A API fica bem sólida quando é estruturada dessa forma, pois com eventuais mudanças no banco de dados, boa parte do código da API é preservada e o retrabalho fica somente por parte do *Repository*, essa estruturação é um pouco mais trabalhosa para ser desenvolvida, mas acaba que é mais vantajosa a longo prazo.

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

A API da Camid foi desenvolvida em JavaScript com ajuda da biblioteca NodeJS, foi projetada orientada a objetos e organizada em algumas pastas, sendo elas:

Controllers: Onde ficarão os métodos separados por telas do front-end (por exemplo: a tela de login, tem um arquivo loginController.js dentro da pasta controllers, que vai ter todos os métodos que a tela de login do front-end vai necessitar).

Data: Onde ficará a conexão com o banco de dados usando a biblioteca Knex.

Middlewares: Onde ficará um método que verifica se o usuário tem permissão para realizar determinada tarefa no front-end.

Models: Onde ficarão as classes que representarão as tabelas do banco de dados.

Repositories: Onde ficarão os métodos para fazer o CRUD (inserção, alteração, busca ou exclusão) de cada tabela do banco de dados (dentro da pasta repositories terá um arquivo por tabela do banco, cada arquivo terá todos os métodos do CRUD para modificar o banco de dados).

Routers: Onde ficarão as rotas URL para a execução dos métodos, salvos nos arquivos da pasta controllers.

A API funciona da seguinte forma: quando clicar em algum botão no front-end, ele estará configurado para fazer uma requisição de URL na API (back-end), quando for feito essa requisição, o arquivo "*router.js*" irá verificar se existe alguma rota de URL cadastrada, se

sim, acionará os métodos de autenticação (se necessário) e de controladores. Quando um controlador é acionado, ele pode necessitar de um arquivo JSON para dar continuidade no código, por exemplo, para cadastrar um novo doador, é necessário que passe um arquivo parecido com esse para o controlador:

```
{
        “nome”:”José Macedo”,
        “telefone”:”19944446666”
}
```

O controlador faz um objeto da classe doador usando um arquivo “doadoresModel” da pasta models, e passa esse objeto para o repositório “doadoresRepo.js” específico de doadores, e chama o método para inserção no banco de dados. Quando o repositório tenta inserir no banco de dados o objeto passado para ele, retorna uma resposta do banco (êxito, ou falha), que por sua vez retorna ao controlador e posteriormente, para o *front-end* onde está o usuário final.

Para assegurar que usuários sem permissões façam alterações no banco de dados, utilizamos a biblioteca *jsonwebtoken*, que permite criar e verificar se *tokens* são válidos, também é possível visualizar seus valores. Funciona da seguinte forma, se no momento de fazer *login* no *front-end* o usuário passou email e senha de um usuário cadastrado na tabela administradores, será gerado um *token* para o usuário, com ele, será possível utilizar todos os métodos de inserção, alteração e exclusão (busca está disponível sem precisar do *token*). O método “*authToken*” que se encontra dentro da pasta “*middlewares*” verifica se o *token* passado pelo usuário à API é válido, se sim, ele deixa prosseguir com a rota, caso seja um *token* inválido, ele impede a continuidade.

Para a segurança da conexão com o banco de dados, foi utilizado a biblioteca dotenv, que permite criar um arquivo .env e configurar dados sensíveis de conexão do banco de dados, como o endereço, porta, usuário, senha, nome do banco e até a chave principal geradora de *tokens*. A utilização do .env é considerada uma boa prática de programação, pois é possível compartilhar seu código da API sem se comprometer com dados sensíveis vazados na internet sobre a conexão com seu banco de dados.

3.2.2 FRONT-END

No desenvolvimento de um front-end para o “Sistema de Gestão e Inteligência de Negócios para Organizações Sociais”, vamos usar o Angular, que é um framework front-end de desenvolvimento web com a importação de standalone, utilizando uma abordagem mais simplificada para as criações de componentes.

O desenvolvimento da Camid foi projetada com um foco especial em designer responsivo, garantindo que a plataforma se adapte a diferentes dispositivos, como desktops, tablets e smartphones, permitindo que os usuários acessem o sistema com facilidade, independentemente do tamanho da tela.

Outro aspecto importante foi o cuidado com a experiência do usuário (UX) e também a interface do usuário (UI). Priorizamos a simplicidade no desenvolvimento, sendo fácil de compreender a interação com o site, proporcionando uma interação mais intuitiva e agradável. Além de ser visualmente atrativo.

Além disso, o projeto faz uso de frameworks modernos de desenvolvimento web, como o Angular 17 no modo standalone. Com isso, conseguimos criar uma aplicação dinâmica e de alto desempenho, aproveitando as facilidades de modularização e componentes reutilizáveis dentro do projeto.

Essa escolha se baseia nas vantagens do angular como um framework robusto e eficiente, com suporte nativo ao TypeScript, o que otimiza a organização do código e melhora a segurança da aplicação (Conteige.cloud, 2024).

3.3 COMPUTAÇÃO EM NUVEM

A computação em nuvem é uma tecnologia atual que permite o armazenamento, processamento e gerenciamento de dados e aplicações através de servidores remotos, ou seja, que não estejam armazenados em computadores pessoais e/ou empresariais. Em suma, os servidores são providos e disponibilizados sob demanda; empresas como Amazon Web Service, Google Cloud Platform e Microsoft Azure oferecem esses serviços de infraestrutura em nuvem, facilitando o processo e permitindo com que empresas ou pessoas físicas possam acessar todo este poder computacional de maneira flexível e, na medida do possível, econômica.

3.3.1 OBJETIVOS DO PROJETO DE CLOUD COMPUTING

Em âmbito geral, a computação em nuvem pode ser fundamental para otimizar processos, reduzir custos e melhorar a eficácia operacional, permitindo que mais recursos sejam direcionados para áreas importantes do negócio. Para o nosso projeto, junto com a instituição de caridade CAMID, esperamos poder traçar objetivos palpáveis e alcançá-los com a tecnologia de armazenamento em nuvem.

Podemos, por exemplo, utilizar uma plataforma para registrar e monitorar todas as entradas e saídas de recursos, garantindo a transparência e o controle sobre tal, também permitindo que tais dados sejam acessados de qualquer lugar, podendo inclusive gerar relatórios concisos para compartilhar entre parceiros, doadores e colaboradores da instituição.

No tópico a seguir explicarei mais sobre as aplicações práticas;

3.3.2 APLICABILIDADE E BENEFÍCIOS DA CLOUD COMPUTING NO PROJETO

A computação em nuvem pode e deve ser utilizada de maneira a centralizar e otimizar os gerenciamentos de recursos. No nosso caso, isso diz respeito às doações em dinheiro, alimentos, roupas e outros itens. Uma plataforma especializada criará um controle preciso e atualizado, independente do local ou dispositivo de acesso.

A nuvem também permitirá que a CAMID aumente ou reduza seus recursos tecnológicos conforme a necessidade, possibilitando adicionar mais usuários permitidos para o acesso das informações, aumentar o armazenamento e/ou ampliar os serviços rapidamente. E a vantagem seria a não-necessidade de investir em infraestrutura física, como servidores

locais e máquinas com maior capacidade de armazenamento. Tudo isso estará disponível nos serviços contratados, o que seria ideal para casos de campanhas sazonais ou emergenciais, onde o número de doadores e beneficiários pode aumentar de forma repentina e temporariamente.

Deste modo, os gastos minimizados com manutenções e equipamentos específicos de TI seriam redirecionados para as atividades financeiras da caridade em si.

3.3.3 VANTAGENS DA CLOUD COMPUTING

Uma das vantagens de ferramentas colaborativas como a nuvem é criar a possibilidade de trabalhos em conjunto de forma híbrida entre o online e o presencial. Organizar campanhas e coordenar atividades com seu time de qualquer lugar é algo essencial para instituições que dependem de funcionários, voluntários e parcerias externas para caminhar. Com a comunicação facilitada através da plataforma, o trabalho remoto se torna cada vez mais eficiente.

A automação de atividades também se enquadra como um recurso indispensável para o trabalho remoto, permitindo gerenciar rotinas de trabalho, atualizar informações de inventário, gerar relatórios atualizados a qualquer momento, enviar e-mails de forma automática e otimizar o tempo ao reduzir erros operacionais bobos.

Outro ponto crucial é a segurança e proteção de dados sensíveis. Os provedores de nuvem investem muitos recursos em segurança, oferecendo backups periódicos e automáticos, proteção contra ataques cibernéticos e um total controle de acesso para o contratante. Isso protege as informações confidenciais de doadores e beneficiários, e fortalece a confiança de todos para com a casa de caridade e suas atividades.

Dessa forma, a computação em nuvem se torna um recurso estratégico para a CAMID maximizar o impacto de sua missão social, através da transparência e confiabilidade de sua causa, juntamente de recursos e processos otimizados.

3.3.4 DESENVOLVIMENTO EM CLOUD COMPUTING

Para o nosso projeto com a CAMID, o modelo de Infraestrutura como Serviço (IaaS) permite que utilizaremos recursos de computação virtualizados (como servidores, armazenamento e redes) sem precisar investir em infraestrutura física. Assim, ganhamos flexibilidades e controle sobre o sistema, além de manter os custos baixos e escaláveis conforme a necessidade.

Os principais recursos e benefícios seriam:

Escalabilidade e Flexibilidade: O modelo IaaS nos permite ajustar o uso de recursos com facilidade, especialmente em períodos de alta demanda. Isso, junto da característica de apenas pagar o que fora utilizado, gera custos mais flexíveis e reduzidos aos se comparar com um cenário onde possuímos servidores próprios.

Acessibilidade: todos colaboradores e membros da equipe gestora podem acessar o sistema e os dados de qualquer lugar, o que facilita o trabalho remoto e a comunicação efetiva.

Segurança e Backup: Os dados ficam protegidos contra falhas e com backups automatizados, impedindo que se percam e gerem prejuízos indesejados.

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

Nossa equipe optou em desenvolver com a AWS. Por conseguirmos compreender as necessidades da empresa e os objetivos do projeto, a AWS se destacou devido a sua ampla gama de serviços flexíveis, permitindo-nos ter uma alta escalabilidade dos serviços em nuvem de acordo com o aumento da demanda. Além disso, a confiabilidade que ela nos fornece é de alto nível, garantindo a segurança completa e também o desempenho que o projeto exige.

Na AWS, escalamos nosso projeto com alguns objetivos:

Escalabilidade: AWS permite ter uma expansão dos serviços conforme o necessário, permitindo uma capacidade para acomodar variações de uso.

Preço: Estrutura de nuvem “pague conforme o uso” foi uma opção essencial para o nosso projeto, pois minimiza os custos desnecessários e tempo de inatividade do sistema.

Confiabilidade: Com um sistema global e redundante, a AWS oferece alta disponibilidade e recuperação de desastres, essenciais para minimizar prejuízos e tempo de inatividade.

Suporte: A AWS oferece uma gama de suporte personalizada conforme o cliente necessita.

3.3.6 DESENVOLVIMENTO EM CLOUD COMPUTING

O modelo de aplicação que usaremos, como descrito anteriormente, será o IaaS (Infraestrutura como Serviço). Ele fornecerá servidores, redes e armazenamento virtuais, em forma de nuvem. Com isso, poderemos hospedar nosso sistema de gerenciamento de doações e dados de beneficiários sem a necessidade de uma estrutura física aplicada a isso, sendo o modelo ideal para o desenvolvimento e armazenamento de documentos e arquivos de forma segura. O IaaS se mostrou mais eficaz quando o assunto é reduzir os custos com hardware, oferecendo uma alta escalabilidade e permitindo o crescimento conforme a demanda aumenta, conforme balanceamento de carga.

A importância e a essência do balanceamento de carga na cloud computing é garantir que o desempenho do sistema seja consistente e estável, independente da situação. Ele ajuda a distribuir as requisições e cargas de trabalho entre vários servidores, evitando que se sobrecarregue, e permitindo com que a velocidade de resposta aos usuários seja ágil e satisfatória. Também, como comentado antes, o balanceamento aumenta a disponibilidade do sistema conforme necessário, além de reduzir o risco de quedas do sistema.

Todas essas funcionalidades acontecem por meio de um software especializado, que monitora a demanda em tempo real e distribui o tráfego entre os servidores disponibilizados.

Entrando no aspecto de anatomia de cloud computing, podemos definir suas principais características que, conforme estudo, foram levadas em consideração na tomada de decisões do projeto.

Começando pela **Infraestrutura Física**, também conhecida como **Data Centers**. Nada mais são do que servidores físicos que hospedam todos os recursos computacionais da nuvem. Cada centro é equipado com protocolos rigorosos de segurança, disponibilizando backups e garantindo que os dados contidos estejam sempre disponíveis.

Em seguida, as **Camadas de Virtualização** transformam os recursos físicos dos Data Centers em virtuais. Esse processo de virtualização permite que múltiplos usuários partilhem

do mesmo hardware, sem interferências, através de uma divisão e alocação de cloud conforme a necessidade de cada cliente.

Com tudo pronto para o uso da nuvem por parte dos clientes, os **Serviços de Gerenciamento e Orquestração** monitoram e alocam os recursos conforme demandas, realizando os ajustes automáticos de carga e escalonamento. São estas, as camadas de orquestração, as responsáveis pela configuração e curadoria dos servidores e das redes da nuvem.

Por fim, as **Interfaces e APIs** acessam a nuvem e permitem a integração entre os aplicativos e automações, criando enfim a comunicação entre o usuário e fazendo com que o sistema possa ter funcionalidades completas e complexas.

Para finalizarmos as explicações, devemos citar que a cloud computing é sustentada por paradigmas tecnológicos, responsáveis por garantir a eficiência, segurança e escalabilidade dos serviços. São eles:

Virtualização: Permite a criação de Máquinas Virtuais (VMs) dentro de um único servidor físico, tornando possível e viável o uso de recursos físicos dos servidores de maneira eficiente e flexível para diversas aplicações simultâneas.

Computação Distribuída: A nuvem usa servidores distribuídos para processar os dados e executar várias tarefas simultaneamente, aumentando a velocidade e a capacidade do processamento em si, permitindo assim a divisão dessas tarefas complexas entre várias máquinas.

Escalabilidade Horizontal e Vertical: Horizontalmente, a escalabilidade funciona de modo a adicionar mais servidores para lidar com o aumento de tráfego. Verticalmente, apenas ajusta a capacidade de processamento de um servidor em específico. Ambas as formas são necessárias para o pleno funcionamento e ajuste dos recursos em tempo real.

Multitenancy: A Multilocação diz respeito a recursos físicos e/ou aplicativos compartilhados entre vários usuários, mantendo a separação de dados e garantindo que cada um tenha sua experiência segura, personalizada e individual.

Automação e Orquestração: Automatiza as tarefas e os recursos da nuvem, balanceando a carga entre os servidores. Permite o escalonamento e garante uma resposta mais rápida quanto a alterações das demandas.

3.3.7 GOOGLE CLOUD ou AWS

A Amazon AWS sobressaiu em relação ao Google Cloud, não considerando somente a questão do preço, mas sim em questão de praticidade, facilidade e simplicidade.

A Unifeob nos proporcionou uma ponte com os serviços da Amazon, garantindo US$100,00 (cem dólares) para nós alunos nos familiarizarmos com todos os serviços dela. Por isso, ao migrar para o projeto, foi bem simples, pois as atividades em sala durante o semestre letivo utilizou boa parte dos serviços da AWS, praticamente os mesmos do projeto da Camid.

3.4 ESTRUTURA DE DADOS

Os levantamentos de requisitos do projetos foram:

- **Cadastro de doadores**:
    - Nome, endereço e contato.
- **Registro de doações**:
    - Realizada por Pix,dinheiro, alimento e roupa.
- **Histórico de doações**:
    - Entrada de dados e Saída de dados Cadastrados.

3.4.1 LEVANTAMENTO DE REQUISITOS

O objetivo do projeto será fazer o levantamento sobre alguns requisitos que serão de extrema importância dentro dele,onde terá uma chuva de ideias entre os colaboradores da equipe e também da ONG serão requisitos fundamentais para o projeto os quais são eles:

- Interface amigável para permitir que a ONG faça upload de informações sobre as doações.
- Painel de controle para visualização de dados sobre as origens dos doadores.
- Necessidade de armazenamento seguro e acesso rápido aos dados.
- Uso de estruturas de dados eficientes para organizar e filtrar as informações, como filas para gerenciamento das doações e listas para armazenar os doadores.
- Possibilidade de integração com outras plataformas, como sistemas de pagamento ou redes sociais.

3.4.2 VALIDAÇÃO DOS REQUISITOS

O objetivo foi garantir que as funcionalidades propostas atendessem às reais necessidades da organização. Primeiramente, os requisitos foram discutidos e alinhados com a ONG por meio de uma reunião com o diretor Rodrigo, onde foram consideradas suas demandas específicas, como a gestão de doações e a necessidade de uma interface simples e acessível.

Para assegurar a viabilidade técnica, foi realizada uma análise detalhada das ferramentas e tecnologias disponíveis, como o uso de um banco de dados relacional MySQL e a API em Node.js. A validação também incluiu a criação de protótipos para testar a

usabilidade e a interação com o sistema, garantindo que a interface fosse intuitiva para os usuários.

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: Esteriótipo e convívio social** Caracterizamos o estereótipo por meios de etnia, idade e gênero, muitos preconceitos existem ainda na sociedade, ter interações diversas com outras pessoas, dificultando a harmonia e sendo difícil de interagir. No meio social, é de suma importância entender o lado das pessoas e por meio desta, conseguimos respeitar uns aos outros, fazendo com que cada estereótipo possa ser inclusivo, e contribuir valores por quem realmente são.
- **Tópico 2: Estereótipos e Representação** Estereótipo pode acarretar sérios problemas dentro da sociedade, pois ocorre preconceitos e ideias errôneas sobre a pessoa. A apresentação social, nada mais é que como as pessoas enxergam você dentro da esfera, como na mídia, no trabalho ou nas interações diárias. Isso pode acarretar problemas, sobre desigualdade social, onde as visões de grupo são sempre baseadas em estereótipos.
- **Tópico 3: Troco likes: A idealização da vida na internet** É notório afirmar que na internet, muitas vezes os padrões são movidos por likes, em uma versão idealizada de sua vida, promovendo mentiras para ganhar curtidas, tendo imagem superficial e estereótipos de si mesma, podendo acarretar sua vida fora da internet por meio de mentiras.
- **Tópico 4: Convivendo com a diferença** a importância de combater estereótipos e preconceitos que afetam o convívio social, deste o ambiente de trabalho até o dia a dia. Exemplos incluem a visão generalizada de muçulmanos como terroristas ou a falta de oportunidade para jovens e idosos no mercado de trabalho. O choque de gerações, por sua vez, pode ser uma oportunidade para aprendizado e colaboração, desde que haja respeito pelas diferenças.

3.5.2 ESTUDANTES NA PRÁTICA

Link do vídeo no youtube: https://youtu.be/O0CgOI-jy00

QR code:

4. CONCLUSÃO

Inicialmente, o projeto foi motivado pela necessidade de criar uma solução tecnológica para a ONG Casa de Apoio ao Menor Irmã Dulce (Camid), focando na gestão de doações e no acompanhamento de entradas e saídas de recursos. Para isso, a equipe optou por implementar um sistema de gerenciamento robusto, utilizando tecnologias modernas que garantem maior controle sobre os dados e promovem a eficiência nas operações da ONG. No banco de dados optamos pelo MySQL,um banco relacional gratuito que atende às necessidades do projeto, permitindo o relacionamento eficiente entre as tabelas, como doadores, administradores, entradas e saídas. Usamos triggers, procedures e views, otimizando a manipulação de dados e assegurando que as informações sejam tratadas de forma automática e consistente. A API foi desenvolvida em NodeJS, e utilizamos Controllers, Models, Repositories e Middlewares, o que facilita futuras atualizações e manutenção, também foi utilizado o uso de tokens JWT para autenticação garante maior segurança, impedindo o acesso não autorizado ao sistema.

No front-end, o uso do Angular 17 possibilitou a criação de uma interface responsiva e de fácil usabilidade, proporcionando uma boa experiência para o usuário final.

Priorizamos a simplicidade, sem abrir mão da funcionalidade, o que resultou em um design eficiente e atrativo. Outra parte de extrema importância foi a computação em nuvem, onde optamos por utilizar a AWS como provedor, foi possível escalar os recursos conforme a demanda da ONG, além de garantir segurança, backup e acessibilidade.

O uso do modelo IaaS (Infraestrutura como Serviço) permitiu o controle eficiente dos custos e a flexibilidade necessária para expandir ou reduzir os serviços conforme a necessidade da Camid. Durante o caminhar do projeto encontramos alguns desafios para a sua finalização dentre eles o que mais se destacaram foi o Angular 17 no modo standalone e a integração da API com o banco de dados, também tivemos dificuldades para a escolha do provedor e na parte da implementação das práticas de segurança .

Ao final do projeto percebemos quanta experiência ele nos trouxe, além de conseguir atender ao propósito que foi pedido pela ONG, e ele não apenas atende às demandas atuais dela, mas também oferece escalabilidade e flexibilidade para futuros upgrades.

REFERÊNCIAS

KOTLER, P.; KELLER, K. L. Administração de marketing. 14. ed. São Paulo: Pearson Prentice Hall, 2012.

CONTEINGE.CLOUD. **Vantagens e desvantagens do framework Angular.** Em Conteige.cloud. Disponível em https://conteige.cloud/angular-vantagens-e-desvantagens/

Pressman, R. S. (2019). Engenharia de Software: Uma Abordagem Profissional (9ª ed.). McGraw-Hill Education, Capítulo 7, pp. 212-213.

ANEXOS

Anexo 1: **Visualizações elaboradas no banco de dados:**

Fonte: Autores(2024).

Anexo 2: **Exemplo da visualização "Histórico detalhado" onde une doações recebidas, compras realizadas, com nomes de doadores e data e hora realizada.**

```
1  CREATE
2      ALGORITHM = UNDEFINED
3      DEFINER = `root`@`localhost`
4      SQL SECURITY DEFINER
5  VIEW `view_historico_detalhado` AS
6      SELECT
7          `h`.`idhistorico` AS `idhistorico`,
8          `e`.`identrada` AS `identrada`,
9          `s`.`idsaida` AS `idsaida`,
10         `d`.`nome` AS `nome`,
11         `s`.`descricao` AS `descricao`,
12         `h`.`valor` AS `valor`,
13         `h`.`data` AS `data`
14     FROM
15         `historico` `h`
16     LEFT JOIN `entrada` `e` ON `h`.`identrada` = `e`.`identrada`
17     LEFT JOIN `saida` `s` ON `h`.`idsaida` = `s`.`idsaida`
18     LEFT JOIN `doadores` `d` ON `e`.`iddoador` = `d`.`iddoadores`;
19
```

Fonte: Autores(2024).