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

<NavyTec>

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

<NavyTec>

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:

Giovana Muniz dos Santos, RA 23000136
Julia Karoline Ramos Andrade, RA 23000895
Livia Ribeiro Venezian, RA 23001001
Nathan Heller Torati Salmaso, RA 23000428
Thalison Antonio de Andrade Eufrosino, RA 23000906

SÃO JOÃO DA BOA VISTA, SP
NOVEMBRO 2024

SUMÁRIO

1. INTRODUÇÃO

Neste projeto, o principal objetivo foi dar continuidade ao Sistema de Gerenciamento de Pacientes iniciado no módulo anterior, tendo como beneficiário a Associação de Apoio à Pessoa Com Câncer Lucas Tapi, uma organização sem fins lucrativos.

Assim como na última etapa do projeto, a manipulação de dados utilizando Business Intelligence foi essencial, uma vez que o principal objetivo do sistema é a migração dos registros de pacientes e profissionais para um banco de dados, facilitando não só o acesso como também a visualização desses dados.

Como forma de aumentar a eficiência operacional, a AWS foi escolhida como provedora de cloud computing, uma vez que oferece diversos recursos que contribuem para o armazenamento e gerenciamento de dados, fornecendo também um alto nível de segurança.

No desenvolvimento do back-end do projeto, a equipe optou por utilizar JavaScript e Node.js para implementar uma API que fosse eficiente e segura. Buscando garantir a escalabilidade, os princípios da arquitetura RESTful foram utilizados para a construção da API, utilizando também um processo de autenticação com JWT para proteger a confidencialidade dos dados dos pacientes e profissionais.

Tendo decidido os pontos principais do funcionamento do sistema, foi possível prosseguir para a idealização da interface, e posteriormente para sua implementação. Para o desenvolvimento do front-end do projeto, a equipe decidiu utilizar a plataforma Angular.

2. DESCRIÇÃO DA EMPRESA

A organização não governamental visada para a elaboração do projeto do módulo Engenharia de Software tem como razão social Associação de Apoio à Pessoa Com Câncer Lucas Tapi, sendo mais conhecida como Associação dos Sonhos Lucas Tapi com CNPJ 33.267.699/0001-00 de Vargem Grande do Sul localizado na Rua Major Correa, 709, Centro.

Seu principal propósito é oferecer assistência às famílias que enfrentam a batalha contra o câncer, fornecendo apoio psicológico e auxílio financeiro por meio da arrecadação de doações da comunidade. Esta ONG atende diretamente pessoas e suas famílias afetadas pela doença.

3. PROJETO INTEGRADO

3.1 TÓPICOS AVANÇADOS DE BANCO DE DADOS

A criação de um banco de dados eficiente e bem estruturado foi a principal ferramenta para o desenvolvimento deste projeto, uma vez que que o sistema final busca armazenar os dados dos pacientes cadastrados na ONG, e demais informações relevantes sobre seu status e localização, assim como os dados dos profissionais responsáveis pelo atendimento desses pacientes. O armazenamento dos dados foi feito de forma local, e para isso utilizamos o MySQL, um sistema de gerenciamento de banco de dados de grande popularidade, e como dito por Manzano, (2011, p.23), é uma ferramenta prática e de fácil uso, assim como um excelente desempenho e estabilidade, além de rodar em diversas plataformas computacionais.

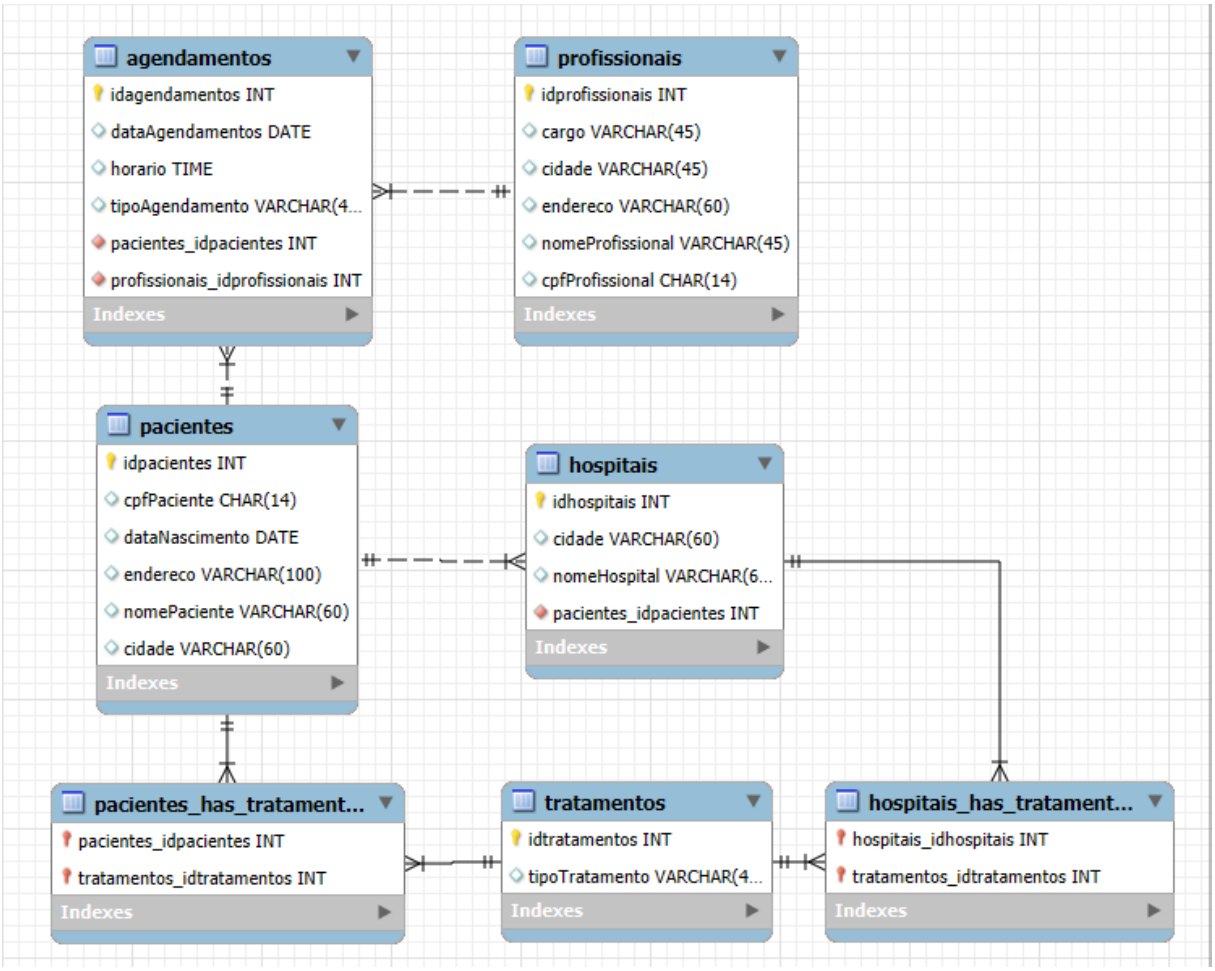
Assim, esta foi a primeira etapa a ser analisada e construída antes de avançar com o desenvolvimento do sistema.

3.1.1 MODELO LÓGICO

Uma vez que o modelo lógico serviria como base para o restante do projeto, o primeiro passo foi passar os planos do sistema para o MySQL por meio de tabelas, facilitando a visualização dos atributos e relacionamentos que cada setor teria.

O sistema conta uma tabela para armazenar os dados dos pacientes, uma para os profissionais afiliados e também para os hospitais localizados na região de atuação da ONG. Além disso, uma tabela de agendamentos será utilizada para registrar as consultas marcadas com os pacientes. Por fim, uma tabela denominada “tratamentos”, que servirá para associar quais pacientes precisam de determinada forma de tratamento com os hospitais que a oferecem. Nos Anexos A e B é possível observar um comando *Select Inner Join* das tabelas de pacientes, tratamentos e hospitais, e sua respectiva *View*, exemplificando a facilidade de acessar essas informações pelo sistema. Segue abaixo o modelo lógico do projeto, feito pelo MySQL Workbench.

Figura - Modelo Lógico



Fonte: Autores

3.1.2 MODELO FÍSICO

O modelo físico do projeto consta com stored procedures, (Anexos C - E), para as principais funções do sistema, sendo elas a inserção de novos elementos nas tabelas citadas anteriormente, assim como a alteração e a deleção desses dados. Como descrito por Milani e Gonçalves, (2021, p.233), a utilização de stored procedures traz vários benefícios, restringindo a interação dos desenvolvedores com o banco e evitando a execução de códigos mal intencionados ou com falhas, uma vez que o stored procedure será responsável por gerenciar todos os recursos necessários. Para garantir um maior controle das informações no banco de dados, uma tabela chamada *Logs* foi criada para armazenar dados enviados por meio de triggers, (Anexos F - H), cada vez que um dos stored procedures é utilizado. Dessa forma é possível saber exatamente quais dados foram alterados e quando, tornando mais fácil monitorar o que acontece no sistema. Por exemplo, quando um stored procedure é utilizado para excluir um paciente, um trigger é acionado e uma nova linha é adicionada na tabela de *Logs,* como pode ser verificado no anexo I.

3.2 LINGUAGEM E TÉCNICAS DE PROGRAMAÇÃO

Uma API é um conjunto de definições e protocolos que permitem a comunicação entre diferentes sistemas e softwares. Ela funciona como uma ponte que conecta aplicações, facilitando o compartilhamento de dados e funcionalidades de forma segura e controlada. Em vez de acessar diretamente um banco de dados ou lógica interna, as APIs expõem apenas certos serviços e dados de maneira estruturada, permitindo que outras aplicações os utilizem para construir novos recursos. No contexto da web, uma API geralmente se refere a uma API REST, que usa métodos HTTP que no caso desse projeto são os recursos GET, POST, PUT e DELETE para gerenciar e manipular dados remotamente.

No desenvolvimento do projeto para a ong lucas tapi, foi utilizada a linguagem JavaScript com Node.js, aplicando técnicas de programação assíncrona e modularidade. Para tornar o código mais compreensível e sustentável, foram aplicados os princípios de Clean Code e o padrão de projeto Model-View-Controller (MVC) como pode ser visto no anexo J. Segundo Martin (2020, p.46), “escrever código limpo é um desafio que exige disciplina e boas práticas que facilitam a manutenção e compreensão por outros desenvolvedores”.

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

Na construção da API do Sistema de Gestão e Inteligência de Negócios para Organizações Sociais, adotou-se a arquitetura RESTful para criar endpoints escaláveis e bem organizados. Foi implementada a autenticação com JWT para garantir que apenas usuários autorizados tenham acesso aos dados, assegurando a segurança das informações sensíveis.
Além disso, com o Twilio o administrador da api recebe uma notificação por mensagem sempre que um novo usuário é registrado, como se pode ver no código do anexo K.

3.2.2 FRONT-END

Após definir o objetivo do projeto, o foco foi desenvolver um front end que atendesse às necessidades dos usuários, que, neste caso, são os membros da organização. O principal objetivo foi criar uma interface intuitiva e fácil de usar. Segundo Barreto, Jr., e Barboza

(2018, p.21), o design de interação visa desenvolver sistemas que sejam não apenas simples e funcionais, mas também estéticamente agradáveis e atraentes para os usuários. Embora o sistema seja projetado para uso nos computadores da organização, o design é responsivo, garantindo uma boa experiência em diferentes dispositivos.

Para o front end, foi utilizado o framework Angular, que, com sua abordagem de componentes, permite criar interfaces modulares e reutilizáveis, facilitando a organização e manutenção do código. A utilização de componentes standalone também contribuiu para uma maior modularização e reutilização.

3.3 COMPUTAÇÃO EM NUVEM

A computação em nuvem revolucionou a forma como acessamos e utilizamos recursos computacionais. Em vez de possuir e gerenciar seus próprios servidores e infraestrutura, a computação em nuvem permite que você utilize recursos sob demanda, através da internet, pagando apenas pelo que usar. É como alugar um computador poderoso, mas sem os custos e a complexidade de manutenção. Como dito por McCarthy, (1969 apud ERL e Monroy, 2024, p.22): “Se computadores do tipo que eu preconizo se tornarem os computadores do futuro, então um dia a computação pode vir a ser organizada como uma utilidade pública, assim como o sistema telefônico é uma utilidade pública. [...] A utilidade computacional poderia se tornar a base de uma nova e importante indústria”.

Essa definição captura a essência da computação em nuvem: ela não é um local físico, mas uma forma de entregar serviços de TI de maneira mais eficiente. Ao invés de investir em hardware e software, as empresas podem se concentrar em seus negócios, utilizando os recursos da nuvem para escalar rapidamente, reduzir custos e aumentar a agilidade. A definição mais aceita foi publicada National Institute of Standards and Technology (NIST), (2011, apud ERL e Monroy, 2024, p.23): “A computação em nuvem é um modelo para possibilitar acesso ubíquo à rede, conveniente e sob demanda a um bolsão compartilhado de recursos computacionais configuráveis (como redes, servidores, armazenamentos, aplicativos e serviços) que podem ser rapidamente provisionados e liberados com mínimo esforço gerencial ou interação com o provedor de serviço. Esse modelo de nuvem é composto por cinco características essenciais, três modelos de serviço e quatro modelos de implantação.”-National Institute of Standards and Technology (NIST).

Benioff destaca o impacto transformador da computação em nuvem. Ela não apenas altera a forma como as empresas utilizam a tecnologia, mas também redefine seus modelos de negócios e a forma como competem no mercado.

3.3.1 OBJETIVOS DO PROJETO DE CLOUD COMPUTING

Para começar é preciso entender que a Computação em Nuvem é o fornecimento de serviços de computação (servidores, armazenamento, banco de dados, rede…etc), pela internet. Com o auxílio da computação em nuvem, diminui-se gastos desnecessários e aumenta a eficiência operacional, tornando-a mais dinâmica. O uso de plataformas cloud, garante por exemplo que os profissionais consigam acessar seus dados a partir de qualquer local.

Possui uma maior capacidade de dimensionamento elástico, ou seja um armazenamento maior e uma maior largura de banda.

Por fim, a execução de ferramentas na nuvem traz mais eficiência à rotina e apesar de todos os benefícios, é importante ressaltar que a segurança da informação é uma preocupação fundamental na computação em nuvem. Os principais provedores de nuvem investem em tecnologias de segurança de ponta para proteger os dados de seus clientes. Ao escolher um provedor confiável, você pode ter a certeza de que seus dados estarão seguros."

3.3.2 APLICABILIDADE E BENEFÍCIOS DA CLOUD COMPUTING NO PROJETO

Com a computação em nuvem a organização social fará o cadastro dos pacientes de uma forma rápida, e terá acesso a esses dados com facilidade.

A segurança dos dados dos pacientes será maior, eles não precisariam se preocupar com dados desatualizados.

3.3.3 VANTAGENS DA CLOUD COMPUTING

A empresa em questão tem o sistema de cadastro de pacientes todo manual.
A vantagem do cloud computing para eles seria custo, pois a computação em nuvem vai eliminar gastos de capital como compra de hardware e software, execução de data centers locais entre outros.

Aumentaria a velocidade de modo que a maior parte dos serviços de computação em nuvem são fornecidos para que grandes quantidades de recursos sejam provisionados em minutos.

O desempenho seria maior pois os serviços são atualizados regularmente de forma rápida e eficiente, tendo assim uma economia no escalonamento. Traz a vantagem da redução de custos de backup de dados.

E por fim a segurança que eles teriam, pois a nuvem fornece um amplo conjunto de políticas, tecnologias e controles que fortalecem a segurança. Protegendo dados, aplicativos e a infraestrutura contra possíveis ameaças.

3.3.4 DESENVOLVIMENTO EM CLOUD COMPUTING

Utilizaremos o PaaS (Plataforma como Serviço) para desenvolvimento de aplicações web ,já que no desenvolvimento é oferecido uma maior escalabilidade, O SaaS (Software como Serviço) já que possui uma fácil implementação e baixo custo operacional. E o IaaS (Infraestrutura como Serviço) Assim a empresa vai ter um maior controle e flexibilidade dos dados.

O balanceamento de carga é uma técnica fundamental para garantir o alto desempenho, a disponibilidade e a eficiência das aplicações em nuvem. Temos a Distribuição de carga, Melhora do desempenho, Escalabilidade. Vai permitir que a empresa tenha um serviço de alta qualidade e escalabilidade, garantindo uma boa experiência com o aplicativo.

Como componentes utilizamos Camada de Infraestrutura, Camada de Plataforma e Camada de Aplicação. E como elementos utilizamos Modelo de serviço, Escalabilidade, Segurança, Gerenciamento e Custos.

Com isso escolhemos a arquitetura ideal que melhor adaptou-se ao orçamento, otimizamos custos, facilitamos a manutenção do aplicativo.

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

A escolha da AWS como provedor de nuvem foi fundamentada em uma análise cuidadosa das necessidades e objetivos da empresa, considerando aspectos como preço, desempenho, segurança e flexibilidade. A AWS se mostrou a opção mais adequada para atender aos requisitos do projeto e garantir o sucesso da sua iniciativa na nuvem.

Analisamos os principais pontos considerando aspectos como preço, confiabilidade, flexibilidade, escalabilidade, suporte, entre outros, e através dessa análise escolhemos o provedor que melhor se encaixa na empresa.

A migração para a nuvem oferece inúmeros benefícios, mas também pode trazer novos desafios relacionados à segurança e à conformidade. Para garantir a proteção dos dados e a aderência às regulamentações da empresa, consideramos os seguintes aspectos: modelo de compartilhamento e responsabilidade que faz com que o provedor seja responsável pela segurança da infraestrutura enquanto o cliente é responsável pela segurança dos dados e aplicações executadas na nuvem, criptografia, backup e recuperação de desastres,entre outros.

3.3.6 DESENVOLVIMENTO EM CLOUD COMPUTING

Temos os seguintes Modelos de Aplicação em Cloud Computing:

SaaS (Software as a Service): O software é entregue como um serviço pela internet, como o Google Docs.

PaaS (Platform as a Service): Oferece uma plataforma para desenvolvedores criarem aplicações sem se preocupar com a infraestrutura.

IaaS (Infrastructure as a Service): Disponibiliza recursos de hardware virtualizados, como servidores e armazenamento.

O balanceamento de carga é uma técnica que distribui o tráfego de internet entre vários servidores. Se todo o tráfego fosse direcionado para um único servidor, ele rapidamente seria sobrecarregado, causando lentidão e até mesmo falhas no site. O balanceamento de carga evita esse problema, distribuindo as solicitações entre vários servidores.

Um balanceador de carga atua como um controlador de tráfego aéreo, direcionando as requisições para os servidores disponíveis, utilizando de diferentes algoritmos para isso, como por exemplo Round Robin,Weighted Round Robin, Least Connections.

Os aspectos da anatomia de cloud computing são: Front-end que é a interface do usuário, Controlador responsável por gerenciar os recursos, Storage vai armazenar os dados dos clientes no caso da empresa em questão, Network que vai conectar os componentes, e temos o Server que vai ser usado para executar as aplicações.

Como paradigmas tecnológicos subjacentes de cloud computing temos a Virtualização, Contêinerização, Microsserviços, APIs e a Automatização.

3.3.7 GOOGLE CLOUD ou AWS

Uma proposta eficaz para a implantação de um projeto em cloud computing deve contemplar os seguintes aspectos, Análise das Necessidades e Objetivos, Escolha do Modelo de Cloud Computing, Seleção do Provedor de Nuvem, Arquitetura da Solução, Planejamento da Migração, Treinamento e Suporte, Governança da Nuvem. A proposta é clara, completa e detalhada abordando todos os tópicos necessários para a realização da proposta.

As funcionalidades são vastas e abrangentes e permitem que a empresa seja mais ágil e flexível. Enquanto no gerenciamento envolve o controle e otimização dos recursos e serviços.

Cenário 1 - Arquitetura de microservices: Construção de uma aplicação modularizada utilizando ECS (Amazon Elastic Container Service) ou EKS (Amazon Elastic Kubernetes Service) para gerenciar contêineres.

Implementação de serviços autônomos e comunicação entre eles utilizando serviços como API Gateway e SNS.

Cenário 2 - Análise de Dados:Criação de um pipeline de dados utilizando serviços como S3, EMR (Elastic MapReduce) e Redshift (data warehouse) para processar e analisar grandes volumes de dados.

Visualização dos resultados utilizando serviços como QuickSight.

3.4 ESTRUTURA DE DADOS

Neste projeto, o objetivo foi garantir a usabilidade e eficiência do sistema, facilitando o gerenciamento de pacientes, hospitais, profissionais e agendamentos. As estruturas foram selecionadas considerando a linguagem JavaScript, tanto no backend com Node.js e Express quanto no frontend com Angular e TypeScript.

De acordo com Cury (2018, p.10), é essencial que os desenvolvedores criem sistemas não apenas eficientes e de fácil uso, mas também que possam ser atualizados com praticidade e rapidez para atender às crescentes demandas do mercado. Portanto, foram implementadas no projeto estruturas de dados que facilitam a modificação e utilização desses dados em diferentes tipos de operações. Uma das principais estruturas implementadas foi a de objetos, que permite desenvolver o código de maneira estruturada, facilitando o armazenamento e a recuperação dos dados. A seguir, apresenta-se o método para o cadastro de um novo paciente:

Figura - Método para criar um paciente

```
// Método para criar um paciente
async create(req, res) {
  const { nome, hospital_id, medico_id } = req.body;
  await knex('pacientes').insert({ nome, hospital_id, medico_id });
  return res.status(201).json({ message: 'Paciente cadastrado com sucesso!' });
},
```

Fonte: Autores

3.4.1 LEVANTAMENTO DE REQUISITOS

A partir dos requisitos levantados, foram definidas as estruturas de dados mais adequadas para cada módulo do sistema. Como discutido por Cury (2018, p.23), as estruturas de dados proporcionam o armazenamento de informações de maneira organizada e eficiente, permitindo maior complexidade e controle do que variáveis básicas em algoritmos simples. No desenvolvimento do front end, uma estrutura amplamente utilizada foi o “FormGroup” do Angular, que, embora não seja uma estrutura de dados tradicional, age como uma ao organizar e armazenar dados de entrada de forma lógica, agrupando controles de formulário relacionados em uma única unidade. Essa abordagem permite tratar o formulário como um conjunto coeso, facilitando a validação e a extração dos valores dos campos para envio ao backend. Nas imagens abaixo, evidencia-se o “FormGroup” chamado “loginForm”, que agrupa os campos email e password da página de login, possibilitando a manipulação e validação centralizada dos dados.

Figura - Formulário de Login

```
<form [formGroup]="loginForm">
  <app-primary-input
    formControlName="email"
      inputName="email"
      type="email"
      label="Email"
      placeholder="joao@example.com.br"
  >
```

Fonte: Autores

Figura - Código da classe LoginComponent com validações de login

```
export class LoginComponent {
  loginForm!: FormGroup;
  constructor(
    private router: Router
  ){
    this.loginForm = new FormGroup({
      email: new FormControl('', [Validators.required, Validators.email]),
      password: new FormControl('', [Validators.required, Validators.minLength(8)])
    })
  }

  submit(){
    console.log(this.loginForm.value)
  }

  navigate(){
    this.router.navigate(["/registrar"])
  }
}
```

Fonte: Autores

3.4.2 VALIDAÇÃO DOS REQUISITOS

O provedor de computação em nuvem utilizado foi a AWS. A escolha desse serviço se deu devido à ampla variedade de recursos que se encaixam com o banco de dados relacional MySQL utilizado no projeto. Os principais serviços da AWS utilizados foram o Amazon RDS (Relational Database Service), que pode ser escalado horizontalmente, permitindo que consultas de leitura sejam distribuídas entre várias instâncias, melhorando a performance sem comprometer a integridade dos dados.

Além do RDS, também foi utilizado o Amazon EC2, que permite a criação e gerenciamento de servidores virtuais chamados de instâncias. Essas instâncias são como máquinas físicas, mas estão hospedadas na nuvem, e podem ser configuradas e escaladas conforme as necessidades do usuário.

3.5 CONTEÚDO DA FORMAÇÃO PARA A VIDA: ENFRENTANDO ESTEREÓTIPOS

3.5.1 ENFRENTANDO ESTEREÓTIPOS

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

É importante destacar como os estereótipos influenciam o ambiente universitário e a vida social, levando a preconceitos e exclusão. Reconhecer a individualidade nas interações em grupo, promovendo a inclusão é de extrema importância no dia a dia, não de forma presencial como também em plataformas virtuais. Além disso, esses estereótipos são extremamente prejudiciais mesmo em ambientes profissionais, onde um empregador pode deixar de contratar uma pessoa baseando-se em idéias errôneas, acreditando que o gênero ou raça de um indivíduo afeta sua competência.

- **Tópico 2: Estereótipo e representação**

Apesar de grande parte dos estereótipos e preconceitos serem espalhados por indivíduos de forma orgânica, ou seja, por meio de piadas, histórias e demais comentários, as mídias digitais ainda são um dos principais instrumentos de influência na sociedade. A representação de certas culturas e minorias em filmes, séries e outros pode facilmente convencer alguém que não tenha acesso a informações reais sobre o que está sendo apresentado. Dessa forma, é fácil aprender e internalizar preconceitos sem perceber.

Assuntos como a pressão para alcançar a padrões de beleza irrealistas e atender a papéis de gênero desatualizados tornam o convívio social superficial e hostil, onde todos estão infelizes com sua própria identidade, e mesmo que de forma subconsciente, passam a julgar seus colegas e amigos da mesma forma. Desafiar esses estereótipos é crucial para promover uma sociedade mais inclusiva e justa, não tolerando comentários preconceituosos e educando outras pessoas sobre os prejuízos desse tipo de comportamento.

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

Diante disso, com as questões das mídias digitais vive-se um equilíbrio complexo entre o mundo material e o digital, onde as redes sociais influenciam profundamente a

percepção de identidade, sucesso e beleza. A busca por validação online pode distorcer a realidade, gerando comparações prejudiciais e padrões inalcançáveis. Mídias como séries de TV e plataformas como YouTube moldam comportamentos e expectativas, criando uma nova economia de influenciadores que impõem tendências. Essas influências também podem promover padrões de beleza mutáveis, levando a práticas prejudiciais à saúde e à autoimagem, enquanto reforçam estereótipos e preconceitos na sociedade. No cenário atual, a busca por validação online leva os indivíduos a problemas. O problema não é a tecnologia, mas a forma como as pessoas se relacionam com ela, como a utilizam.

O autor Wolzor diz ‘toda a gente quer falar mas ninguém quer ouvir’,

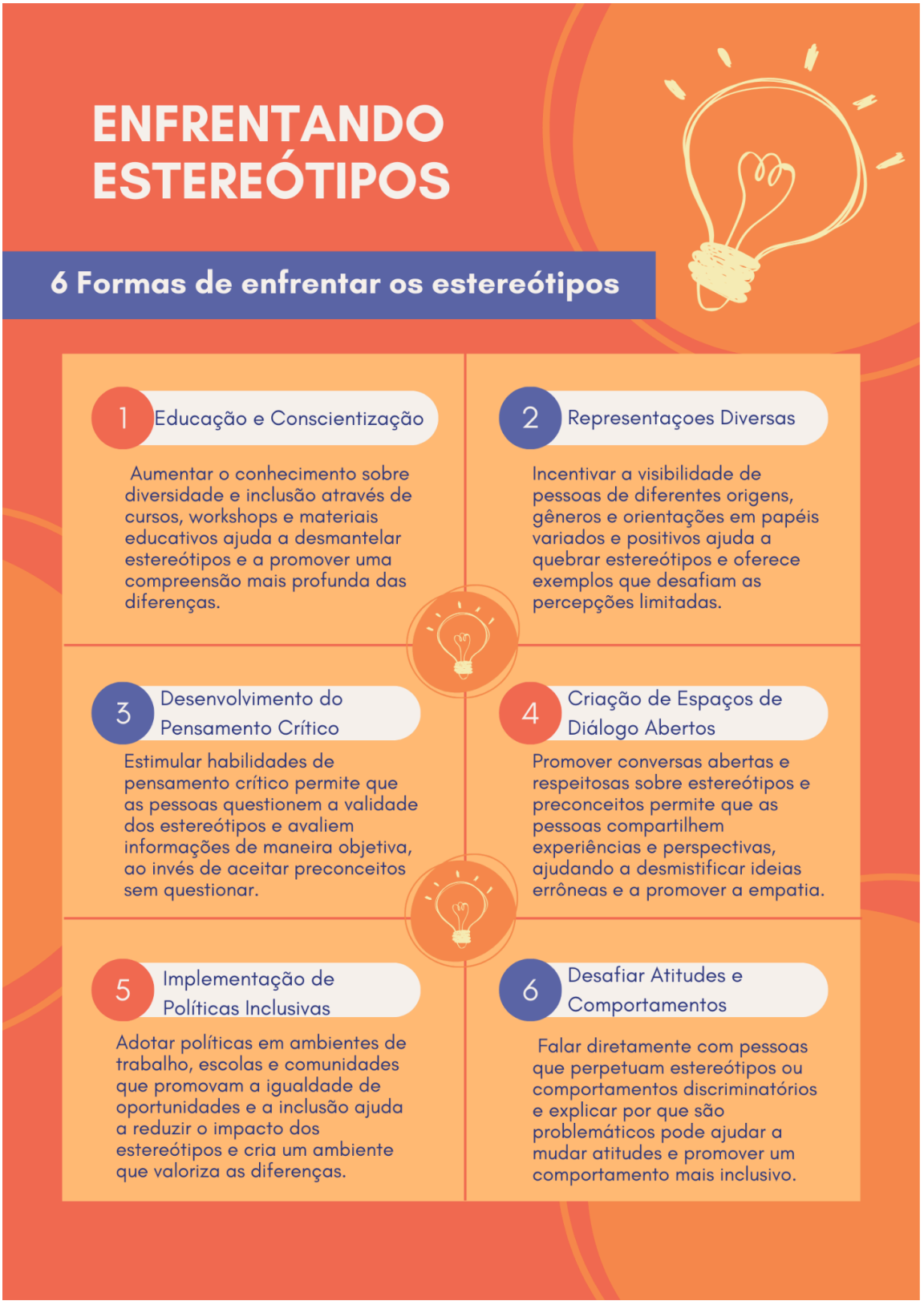
- **Tópico 4: Convivendo com a diferença**

Com o conteúdo apresentado, conclui-se que estereótipos são generalizações imprecisas sobre grupos sociais, que distorcem a percepção e afetam negativamente o tratamento dessas pessoas. Para combatê-los, é essencial promover a convivência entre diferentes culturas e refletir sobre nossos próprios preconceitos. A educação e a valorização da diversidade são fundamentais para construir uma sociedade mais inclusiva e justa.

3.5.2 ESTUDANTES NA PRÁTICA

O objetivo desse material é contribuir para as pessoas refletirem as questões culturais do Brasil, relacionadas a cada uma de suas regiões. Dessa forma, a equipe criou um Banner que reflete esse pensamento com a intenção de conscientizar e educar mais pessoas. Segue o link da publicação no facebook: https://www.facebook.com/share/p/fxUSv1v4WMfxfJwv/

Banner “Enfrentando Estereótipos”



Fonte: Autores

4. CONCLUSÃO

O projeto documentado acima teve como objetivo otimizar e tornar mais eficientes os processos de gerenciamento de cadastro e acompanhamento de pacientes da Associação de Apoio à Pessoas com Câncer Lucas Tapi. A utilização dos recursos AWS para a computação em nuvem, em conjunto com a arquitetura RESTful e a autenticação JWT, garantiram um sistema seguro e prático, que atingiu os requisitos pré-estabelecidos pela equipe.

A interface foi projetada para ser eficiente e intuitiva para os usuários, usando a ferramenta Angular para o desenvolvimento do front-end para tornar o fluxo de trabalho mais dinâmico e facilitar o acesso às informações. O sistema traz não só a otimização do atendimento aos pacientes e demais atividades da Ong, como mantém as informações seguras e confidenciais.

Por fim, o projeto foi concluído de forma satisfatória, atingindo as metas escolhidas pela equipe e os requisitos necessários para a Ong. Com o auxílio dos professores foi possível otimizar o processo de desenvolvimento, ganhando conhecimentos assim como experiência prática, sendo assim tanto a Ong como a equipe foram beneficiadas pelo projeto.

REFERÊNCIAS

CURY, Thiago E.; BARRETO, Jeanine dos S.; SARAIVA, Maurício de O.; et al. Estrutura de Dados. Porto Alegre: SAGAH, 2018. E-book. p.2. ISBN 9788595024328. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788595024328/. Acesso em: 07 nov. 2024.

BARRETO, Jeanine dos S.; JR., Paulo A P.; BARBOZA, Fabrício F M.; et al. Interface humano-computador. Porto Alegre: SAGAH, 2018. E-book. p.22. ISBN 9788595027374. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788595027374/. Acesso em: 08 nov. 2024.

ERL, Thomas; MONROY, Eric B. Computação em Nuvem: Conceitos, Tecnologia, Segurança e Arquitetura . 2ª edição. Porto Alegre: Bookman, 2024. E-book. pág.21. ISBN 9788582606599. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788582606599/. Acesso em: 08 nov. 2024.

MANZANO, José Augusto N G. MySQL 5.5 Interativo: Guia Essencial de Orientação e Desenvolvimento. Rio de Janeiro: Érica, 2011. E-book. p.21. ISBN 9788536519449. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788536519449/. Acesso em: 22 Oct 2024.

MARTIN, Robert C. Desenvolvimento Ágil Limpo. Rio de Janeiro: Editora Alta Books, 2020. E-book. p.i. ISBN 9788550816890. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788550816890/. Acesso em: 05 nov. 2024.

MENDES, Rafael Pereira da Silva. "Estereótipo"; Brasil Escola. Disponível em: https://brasilescola.uol.com.br/sociologia/estereotipo.htm. Acesso em 02 de outubro de 2024.

MILANI, Alessandra Maciel P.; GONÇALVES, Anderson S.; PAES, Claudia A.; et al. Consultas em Bancos de Dados. Porto Alegre: SAGAH, 2021. E-book. p.233. ISBN 9786556900223. Disponível em:

https://integrada.minhabiblioteca.com.br/reader/books/9786556900223/. Acesso em: 22 Oct 2024.

SILVA, João C.; “Impacto das redes sociais no comportamento humano”; Digital Connection. Disponível em https://digitalconnection.pt/impacto-das-redes-sociais-no-comportamento-humano/. Acesso em 27 de setembro de 2024

ANEXOS

ANEXO A - Comando Select Inner Join de pacientes, hospitais e tratamentos

```
SELECT p.nomePaciente AS nomePaciente, h.nomeHospital AS nomeHospital, t.tipoTratamento AS tipoTratamento
FROM pacientes p
INNER JOIN pacientes_has_tratamentos pht ON p.idpacientes = pht.pacientes_idpacientes
INNER JOIN tratamentos t ON pht.tratamentos_idtratamentos = t.idtratamentos
INNER JOIN hospitais_has_tratamentos hht ON t.idtratamentos = hht.tratamentos_idtratamentos
INNER JOIN hospitais h ON hht.hospitais_idhospitais = h.idhospitais;
```

Fonte: Autores

ANEXO B - View Pacientes por Hospital e Tipo de Tratamento Necessário

nomePaciente	nomeHospital	tipoTratamento
Ana Silva	Hospital Municipal	Cirurgico
Maria Oliveira	Hospital Municipal	Cirurgico
Juliana Torres	Hospital Municipal	Cirurgico
Marcos Silva	Hospital Municipal	Cirurgico
Ana Silva	Hospital Geral	Cirurgico
Maria Oliveira	Hospital Geral	Cirurgico
Juliana Torres	Hospital Geral	Cirurgico
Marcos Silva	Hospital Geral	Cirurgico
Carlos Souza	Hospital das Clínicas	Quimioterapia
Luciana Mendes	Hospital das Clínicas	Quimioterapia
Eliana Carvalho	Hospital das Clínicas	Quimioterapia
Carlos Souza	Hospital Moinhos	Quimioterapia
Luciana Mendes	Hospital Moinhos	Quimioterapia
Eliana Carvalho	Hospital Moinhos	Quimioterapia

Fonte: Autores

ANEXO C - Stored Procedure “add_paciente”

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `add_paciente`(
IN cpfPaciente char(14),
IN dataNascimento DATE,
IN endereco varchar(60),
IN nomePaciente varchar(45),
IN cidade varchar(45))
BEGIN
INSERT INTO pacientes (cpfPaciente, dataNascimento, endereco, nomePaciente, cidade)
VALUES (cpfPaciente, dataNascimento, endereco, nomePaciente, cidade);
END
```

Fonte: Autores

ANEXO D - Stored Procedure “update_agendamento”

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `update_agendamento`(
IN a_idagendamentos int,
IN a_dataAgendamentos DATE,
IN a_horario TIME,
IN a_tipoAgendamento VARCHAR(45),
IN a_pacientes_idpacientes INT,
IN a_profissionais_idprofissionais INT)
BEGIN
UPDATE agendamentos
SET
    dataAgendamentos = coalesce(a_dataAgendamentos, dataAgendamentos),
    horario = coalesce(a_horario, horario),
    tipoAgendamento = coalesce(a_tipoAgendamento, tipoAgendamento),
    pacientes_idpacientes = coalesce(a_pacientes_idpacientes, pacientes_idpacientes),
    profissionais_idprofissionais = coalesce(a_profissionais_idprofissionais,profissionais_idprofissionais)
    WHERE idagendamentos = a_idagendamentos;
END
```

Fonte: Autores

ANEXO E - Stored Procedure “delete_hospital”

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `delete_hospital`(in h_idhospitais INT)
BEGIN
    delete from hospitais where h_idhospitais = idhospitais;
END
```

Fonte: Autores

ANEXO F - Trigger “hospitais_AFTER_INSERT”

```
CREATE DEFINER=`root`@`localhost` TRIGGER `hospitais_AFTER_INSERT` AFTER INSERT ON `hospitais` FOR EACH ROW BEGIN
INSERT INTO log_acoes (descricao) values (CONCAT('HOSPITAL ', new.nomeHospital, ' ADICIONADO'));
END
```

Fonte: Autores

ANEXO G - Trigger “profissionais_AFTER_DELETE”

```
CREATE DEFINER=`root`@`localhost` TRIGGER `profissionais_AFTER_DELETE` AFTER DELETE ON `profissionais` FOR EACH ROW BEGIN
INSERT INTO log_acoes (descricao) values (CONCAT('PROFISSIONAL: ', old.nomeProfissional, ' EXCLUIDO'));
END
```

Fonte: Autores

ANEXO H - Trigger “pacientes_AFTER_UPDATE”

```
CREATE DEFINER=`root`@`localhost` TRIGGER `pacientes_AFTER_UPDATE` AFTER UPDATE ON `pacientes` FOR EACH ROW BEGIN
if old.cpfPaciente <> new.cpfPaciente then
    INSERT INTO log_acoes set descricao = CONCAT('UPDATED CPF: ', old.cpfPaciente, ' TO ', new.cpfPaciente);
    end if;
if old.dataNascimento <> new.dataNascimento then
    INSERT INTO log_acoes set descricao = CONCAT('UPDATED DATA DE NASCIMENTO: ', old.dataNascimento, ' TO ', new.dataNascimento);
    end if;
if old.endereco <> new.endereco then
    INSERT INTO log_acoes set descricao = CONCAT('UPDATED ENDERECO: ', old.endereco, ' TO ', new.endereco);
    end if;
if old.nomePaciente <> new.nomePaciente then
    INSERT INTO log_acoes set descricao = CONCAT('UPDATED NOME DO PACIENTE: ', old.nomePaciente, ' TO ', new.nomePaciente);
    end if;
if old.cidade <> new.cidade then
    INSERT INTO log_acoes set descricao = CONCAT('UPDATED CIDADE: ', old.cidade, ' TO ', new.cidade);
    end if;
END
```

Fonte: Autores

ANEXO I - Select Tabela “log_acoes”

Result Grid | Filter Rows: | Edit:

id	descricao
79	PROFISSIONAL: Fernanda ADICIONADO
80	HOSPITAL Hospital Geral ADICIONADO
81	UPDATED CIDADE: Salvador TO Porto Alegre
82	UPDATED CIDADE: Hospital Geral TO Hospital Moinhos
83	HOSPITAL: Hospital Moinhos EXCLUIDO

Fonte: Autores

ANEXO J - Modelo de registro da API

```
const knex = require('knex')(require('../knexfile'));

const HospitalController = {
  // Método para listar hospitais
  async index(req, res) {
    const hospitais = await knex('hospitais');
    return res.json(hospitais);
  },

  // Método para criar um hospital
  async create(req, res) {
    const { nome, endereco } = req.body;
    await knex('hospitais').insert({ nome, endereco });
    return res.status(201).json({ message: 'Hospital cadastrado com sucesso!' });
  },

  // Método para atualizar um hospital
  async update(req, res) {
    const { id } = req.params;
    const { nome, endereco } = req.body;

    await knex('hospitais').where({ id }).update({ nome, endereco });
    return res.json({ message: 'Hospital atualizado com sucesso!' });
  },

  // Método para deletar um hospital
  async delete(req, res) {
    const { id } = req.params;
    await knex('hospitais').where({ id }).del();
    return res.json({ message: 'Hospital removido com sucesso!' });
  }
};
```

ANEXO K - Código para envio de mensagem por meio do Twilio

```
const knex = require('knex')(require('../knexfile'));
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const { sendSMS } = require('../twilio'); // Importa o módulo Twilio

const AuthController = {
  // Método para registrar um funcionário
  async register(req, res) {
    const { nome, email, senha, telefone } = req.body;
    const hashedPassword = await bcrypt.hash(senha, 10);

    try {
      // Registra o novo funcionário no banco
      await knex('funcionarios').insert({ nome, email, senha: hashedPassword, telefone });

      // Envia um SMS ao funcionário após o registro
      await sendSMS(telefone, `Olá ${nome}, seu registro foi concluído com sucesso!`);

      return res.status(201).json({ message: 'Funcionário cadastrado com sucesso!' });
    } catch (error) {
      console.error('Erro ao registrar funcionário:', error);
      return res.status(500).json({ error: 'Erro ao registrar funcionário' });
    }
  },
```