Uni-FACEF CENTRO UNIVERSITÁRIO MUNICIPAL DE FRANCA

Felipe Jorge batarra

Gabriel David Assed

Gabriel Fernandes Pacheco

Richardy Gabriel Rodrigues

Samuel Schisari Lago

CONSTRUÇÃO DE SOFTWARE ORIENTADO COMPONENTE COM O PARADIGMA FUNCIONAL PARA UMA COMUNIDADE

UCE – PARADIGMAS DE PROGRAMAÇÃO II

FRANCA
2024

Felipe Jorge batarra

Gabriel David Assed

Gabriel Fernandes Pacheco

Richardy Gabriel Rodrigues

Samuel Schisari Lago

CONSTRUÇÃO DE SOFTWARE ORIENTADO COMPONENTE COM O PARADIGMA FUNCIONAL PARA UMA COMUNIDADE

UCE – PARADIGMAS DE PROGRAMAÇÃO II

Relatório de Unidade Curricular de Extensão – UCE de Paradigmas de Programação II, apresentado ao Departamento de Computação do *Uni-FACEF Centro Universitário Municipal de Franca*, para atender às disposições da Resolução nº 7 de 18 de dezembro de 2018, que estabelece as Diretrizes para a Extensão na Educação Superior.

Prof. Responsável: Prof. Dr. Daniel Facciolo Pires

FRANCA
2024

SUMÁRIO

1 INTRODUÇÃO

Este relatório apresenta o resultado do desenvolvimento de um software de gestão de pagamentos para a Sociedade Francana de Instrução e Trabalho para Cegos, uma organização dedicada a atender pessoas com deficiência visual. O projeto foi realizado com o objetivo de modernizar e otimizar as operações financeiras da instituição, superando as limitações por processos manuais que anteriormente resultavam em desorganização e ineficiência, impactando negativamente a execução de sua missão social.

A conclusão deste projeto representa um marco importante tanto para a organização quanto para a formação acadêmica da equipe. Alinhado aos **Objetivos de Desenvolvimento Sustentável (ODS)** das Nações Unidas, o software visa gerar impacto positivo em:

1. **ODS 1 (Erradicação da Pobreza)**: O sistema aprimora a eficiência operacional e maximiza o uso dos recursos financeiros, assegurando que cada contribuição alcance seu propósito, promovendo uma gestão mais eficaz e sustentável.
2. **ODS 9 (Inovação e Infraestrutura)**: Este projeto trouxe inovação tecnológica para a instituição, implementando soluções modernas que promovem maior eficiência, transparência e autonomia operacional.
3. **ODS 10 (Redução das Desigualdades)**: O software facilita a inclusão social, permitindo que a Sociedade Francana de Instrução e Trabalho para Cegos preste serviços mais acessíveis e organizados às pessoas com deficiência visual.

2 PROJETO DE UNIDADE CURRICULAR DE EXTENSÃO - UCE

2.1 OBJETIVO GERAL

O objetivo geral deste projeto foi desenvolver e implementar um software de gestão de pagamentos orientado a objetos para a Sociedade Francana de Instrução e Trabalho para Cegos, com backend desenvolvido em **Node.js** e frontend em **React.js**. O sistema foi concluído com sucesso, aprimorando a eficiência, transparência e autonomia operacional da instituição, promovendo uma gestão de recursos financeiros mais eficaz e impactando positivamente no suporte ao desenvolvimento humano e na inclusão social efetiva de pessoas com deficiência.

2.1.1 Objetivos específicos

1. **Desenvolver APIs Funcionais em Node.js**: Implementar interfaces de programação que suportem a comunicação eficiente entre o frontend e o backend, garantindo segurança e desempenho nas operações financeiras.
2. **Criar uma Interface Intuitiva com React.js**: Projetar e implementar uma interface de usuário acessível e responsiva, facilitando o uso do sistema por todos os stakeholders.
3. **Documentar a Arquitetura e os Componentes do Sistema**: Produzir uma documentação técnica completa, abrangendo os aspectos de design, funcionalidades e integrações para suporte à manutenção e evolução do sistema.
4. **Elaborar Manuais de Uso e Operação**: Desenvolver guias que simplifiquem a utilização do sistema pelos administradores e colaboradores da instituição.
5. **Garantir a Integração e Automação dos Processos Financeiros**: Automatizar processos manuais relacionados a pagamentos e transações financeiras, aumentando a eficiência e minimizando erros operacionais.

2.2 METODOLOGIA

O projeto seguiu uma metodologia prática e iterativa, fundamentada na imersão na realidade da comunidade atendida e na aplicação direta dos conceitos estudados em aula. As tecnologias utilizadas incluem **Node.js** para o desenvolvimento do backend e **React.js** para a criação do frontend. As etapas metodológicas foram organizadas da seguinte forma:

1. **Pesquisa Bibliográfica**: Utilização de literatura especializada em **Node.js**, **React.js**, e princípios de orientação a objetos, para fundamentar o desenvolvimento do software.
2. **Visitas de Campo e Entrevistas**: Realização de visitas à Sociedade Francana de Instrução e Trabalho para Cegos, com entrevistas aos responsáveis e usuários da instituição para coletar informações detalhadas sobre os processos atuais e necessidades operacionais.
3. **Entrevistas em Sala de Aula**: Condução de dinâmicas e entrevistas em sala de aula para simular cenários e coletar sugestões e feedbacks sobre o design e a funcionalidade do sistema, promovendo um alinhamento mais próximo com os objetivos do projeto.
4. **Coleta e Análise de Dados**: Levantamento de dados sobre os processos financeiros e administrativos da instituição, para o desenvolvimento de um sistema que atendesse às suas especificidades.
5. **Desenvolvimento e Testes**: Implementação do backend em **Node.js**, do frontend em **React.js**, e uso de conceitos de orientação a objetos para criar um software robusto. Foram realizados ciclos de testes iterativos para garantir a funcionalidade, segurança e acessibilidade do sistema.

3 ESTUDOS TEÓRICOS

Na presente seção, devem ser apresentados os materiais de estudo para a criação do software, como exemplos abaixo:

1) RocketSeat. DevLinks Disponível na web em https://www.rocketseat.com.br/devlinks. Último acesso: março de 2023
Concepções/estudos empregados: Quais os recursos foram estudados e como estes auxiliam na criação do site.

2) W3Schools. JavaScript Tutorial. Disponível na web em https://www.w3schools.com/js/default.asp. Último acesso: 03/2023
Concepções/estudos empregados: Quais os recursos foram estudados e como estes auxiliam na criação do site.

3) Guanabara, G. Curso de HTML5 e CSS3. Disponível na web em https://www.youtube.com/playlist?list=PLHz_AreHm4dlAnJ_jJtV29RFxnPHDuk9o. Último acesso: março de 2023
Concepções/estudos empregados: Quais os recursos foram estudados e como estes auxiliam na criação do site.

4 ATORES E ENTREVISTAS COM A COMUNIDADE

Com base nos dados coletados durante visitas e entrevistas, a Sociedade Francana de Instrução e Trabalho para Cegos, compreende os atores sociais envolvidos e os locais de coleta de dados.

Atores Sociais Identificados

1. **Cidinha - Presidente da Sociedade:**
   Responsável pela gestão geral da instituição, garantindo o cumprimento de sua missão social e representando a organização em suas relações com parceiros e financiadores.
2. **Giovana - Gestora Financeira da Sociedade:**
   Responsável pelos processos de pagamentos e controle financeiro da instituição, incluindo a gestão das contas bancárias e dos recursos provenientes de doações e repasses governamentais.

5 INTEGRAÇÃO DO FRONTEND E DO BACKEND DA SOLUÇÃO

1. Comunicação do React.js com o Back-End (Express)

O React.js se comunica com o back-end, que é feito em Express, através de uma arquitetura baseada em serviços. Em um componente React, utilizamos um serviço específico que faz chamadas HTTP usando o Axios para enviar requisições para as rotas do back-end. No exemplo de um serviço de ordens de pagamento, o serviço fará uma requisição para a URL correspondente no Express. O Express, por sua vez, utiliza o Prisma para interagir com o banco de dados e retornar os dados solicitados para o front-end. O React então manipula os dados, atualizando a interface conforme necessário.

2. Exemplo de Código Fonte

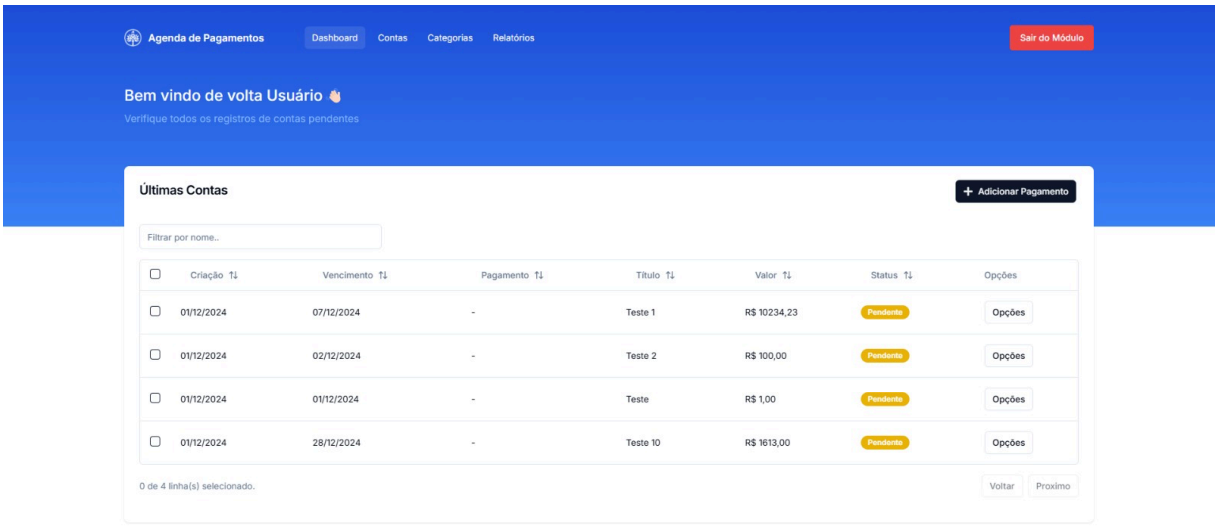
Aqui está um exemplo de como o React.js se comunica com o back-end para listar ordens de pagamento:

Frontend Componente React):

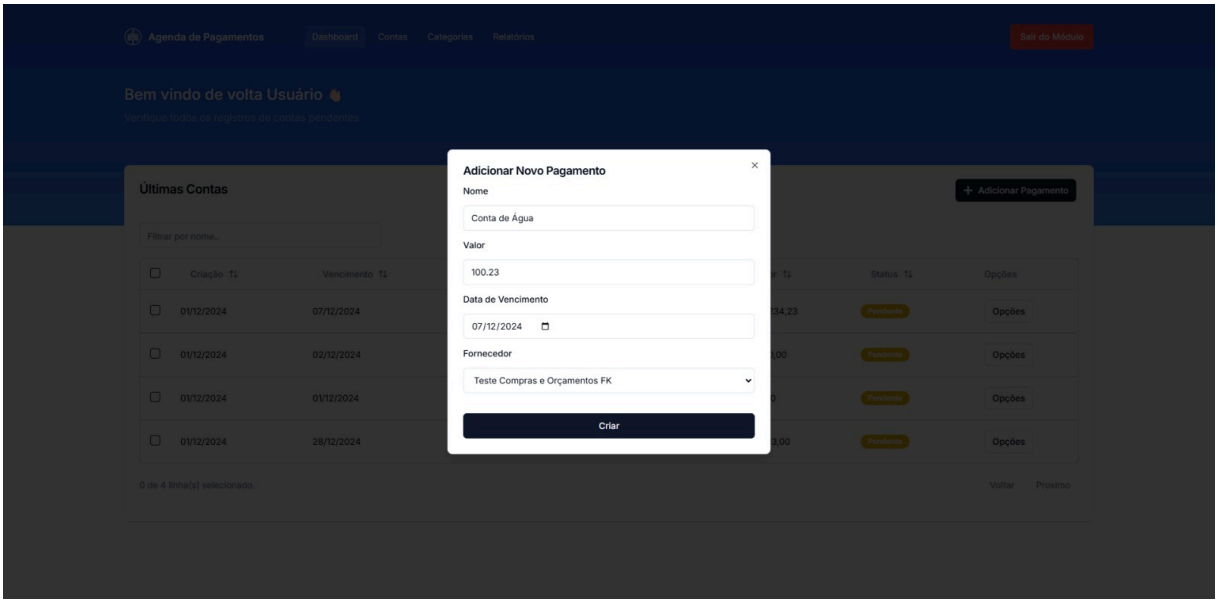
```
import React, { useEffect, useState } from 'react';

import { fetchPaymentOrders } from '../services/paymentService';

const PaymentOrdersList = () => {

  const [orders, setOrders] = useState([]);


  useEffect(() => {

    const getOrders = async () => {

      const response = await fetchPaymentOrders();

      setOrders(response.data);
```

```
    };

    getOrders();

  }, []);


  return (
    <div>
      <h1>Ordens de Pagamento</h1>
      <ul>
        {orders.map(order => (
          <li key={order.id}>
            {order.name} - {order.amount}
          </li>
        ))}
      </ul>
    </div>
  );
};


export default PaymentOrdersList;
```

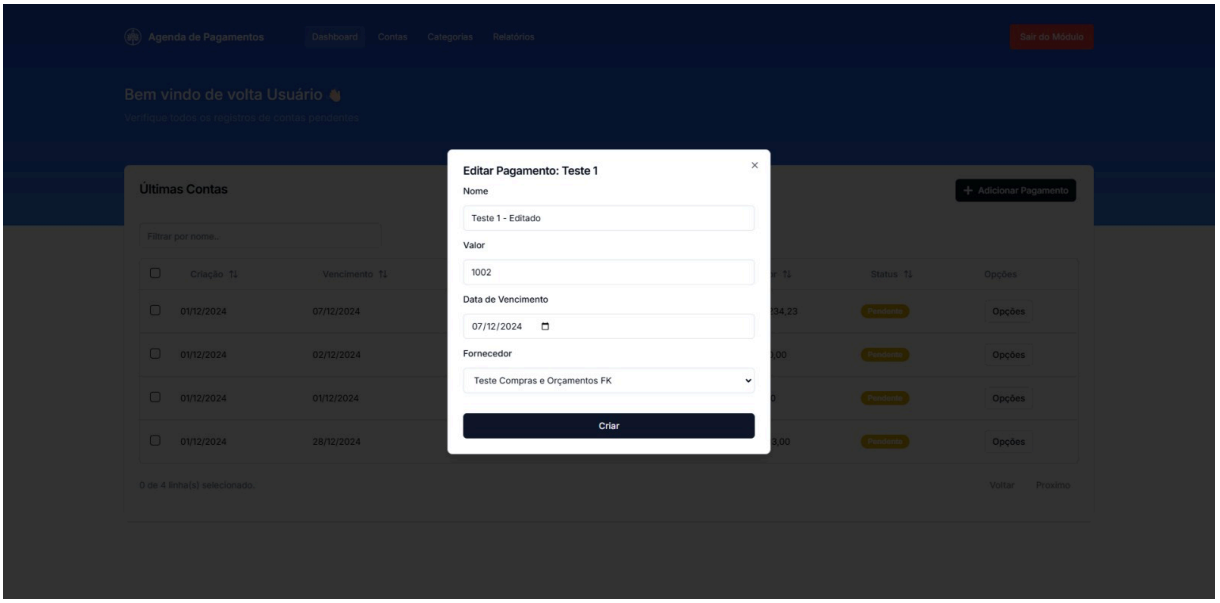
Tela principal, de visualização de pagamentos, utiliza a tabela tb_scheduled_payments do banco de dados para mostrar os pagamentos



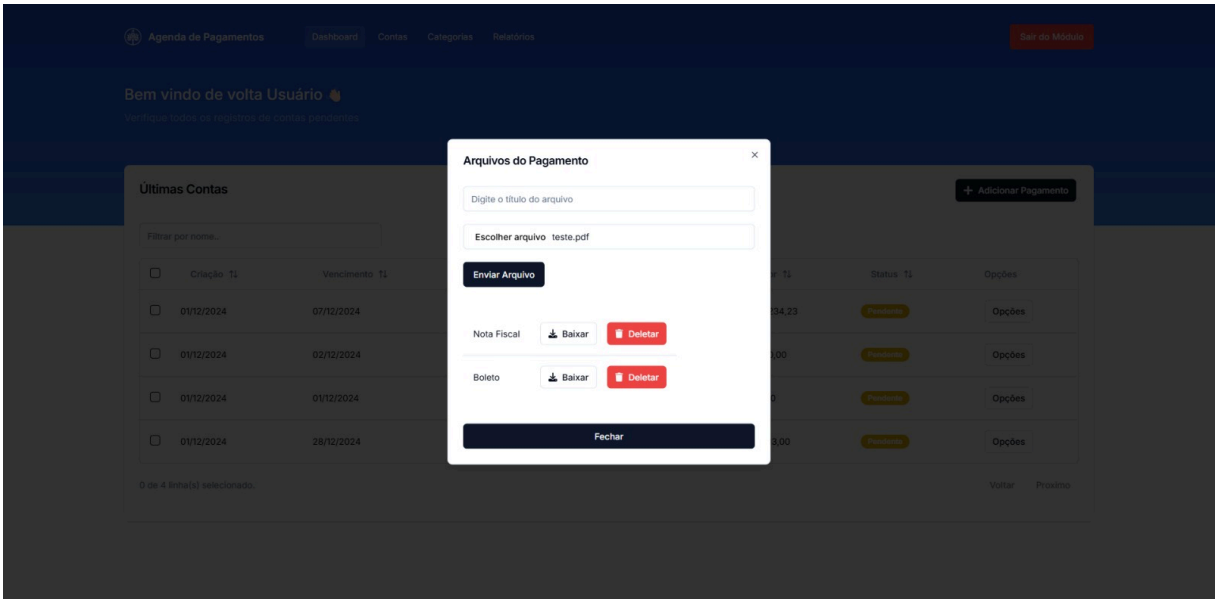
Modal de adicionar pagamento, permite o usuário criar um novo pagamento que é posteriormente adicionado na tabela tb_scheduled_payments



Modal de editar pagamento, pega um pagamento existente e permite o usuário fazer edições em alguns itens (somente enquanto o pagamento não for aprovado para o pagador)



Modal de arquivos de pagamento, permite visualizar os arquivos vinculados a um pagamento (boletos, notas fiscais, etc), fazer upload de um novo arquivo, baixar os arquivos listados e deletar algum arquivo (esse utiliza a tabela tb_scheduled_payments_files



Service Chamada ao Back-End com Axios):

```
import axios from 'axios';

export const fetchPaymentOrders = async () => {
  try {
    const response = await axios.get('https://api.sociedadedoscegos.com/payments');
    return response;
  } catch (error) {
    console.error('Error fetching payment orders:', error);
    throw error;
  }
};
```

Backend Express + Prisma:

```
import prisma from '../database/prisma';


export const getPaymentOrders = async (req, res) => {
  try {
    const orders = await prisma.paymentOrder.findMany();
    res.json(orders);
  } catch (error) {
```

```
    res.status(500).json({ error: 'Error fetching orders' });
  }
};
```

Backend (Express Routes):

```
import express from 'express';
import { getPaymentOrders } from '../controllers/paymentOrdersController';
const router = express.Router();
router.get('/payments', getPaymentOrders);
export default router;
```

Logo a baixo está o código-fonte do arquivo de services do front-end da agenda de pagamentos, que é o código que basicamente faz uma requisição para o nosso back-end e retorna uma resposta, para ser usado dentro dos nossos componentes:

```typescript
import axios from '../api';

export async function getPayments() {
  const { data } = await axios.get('/payments');

  if (!data.success) {
    return [];
  }

  return data.data;
}

export async function getNewPaymentData() {
  const { data } = await axios.get('/provider/prestador');

  if (!data.success) {
    return {
      suppliers: []
    };
  }

  return {
    suppliers: data.data
  };
}

export async function addPayment(data: any) {
  await axios.post('/payments', {
    title: data.title,
    value: Number(data.value),
    due_date: new Date(data.due_date).toISOString(),
    supplier_id: Number(data.supplier_id)
  });
}

export async function editPayment(paymentId: number, data: any) {
  await axios.put(`/payments/${paymentId}`, {
    title: data.title,
    value: Number(data.value),
    due_date: new Date(data.due_date).toISOString(),
    supplier_id: Number(data.supplier_id)
  });
}

export async function deletePayment(id: string) {
  await axios.delete(`/payments/${id}`);
}

export async function listFiles(paymentId: number) {
  const { data } = await axios.get(`/payments/${paymentId}/files`);

  if (!data.success) {
    return [];
  }

  return data.data;
}

export async function uploadFile(paymentId: number, formData: FormData) {
  await axios.post(`/payments/${paymentId}/files`, formData);
}

export async function deleteFile(fileId: number) {
  await axios.delete(`/payments/files/${fileId}`);
}

export function downloadFileURL(fileId: number) {
  return axios.defaults.baseURL + `/payments/files/${fileId}`;
}
```

3. Utilização do JWT

No nosso projeto, o JWT é utilizado para autenticar e autorizar o acesso às rotas do back-end. Quando o usuário faz login, o back-end gera um token JWT que é enviado ao front-end. Esse token é armazenado no front-end e é enviado em cada requisição subsequente para o back-end via cabeçalho *Authorization* com o prefixo *Bearer* . No lado do back-end, o token é verificado para garantir que o usuário tenha permissão para acessar os recursos solicitados. O middleware *verify-permissions.middleware.ts* é responsável por verificar o token e as permissões do usuário:

```
export function allowRoles(roles) {
  return async (req, res, next) => {
    const authorization = req.headers.authorization;
    if (!authorization) {
      res.status(401).send("Not Allowed");
      return;
    }
    const [_, token] = authorization.split(" ");
    const decoded = await jwtVerify(token, Utils.getUint8ArrayFromString(jwtSecret));
    if (!decoded.payload.sub) {
      res.status(401).send("Not Allowed");
      return;
    }
    const user = JSON.parse(decoded.payload.sub);
```

```
      req.user = user;


      if (roles.some((role) => req.user.roles.includes(role))) {
        next();
        return;
      }
      res.status(401).send("Not Allowed");
    };
  }
```

Esse processo garante que apenas usuários autenticados e com permissões adequadas possam acessar as rotas protegidas.

4. Utilização do BCrypt

O BCrypt é utilizado no nosso projeto para criptografar as senhas antes de armazená-las no banco de dados, garantindo maior segurança. Quando um usuário cria ou atualiza a senha, ela é a primeira criptografada usando o BCrypt, e somente a senha criptografada é salva no banco. Ao realizar o login, a senha fornecida pelo usuário é comparada com a versão criptografada no banco utilizando o método *bcrypt.compare()* . Isso evita o armazenamento de senhas em texto simples, protegendo as informações sensíveis dos usuários.

Exemplo de uso do BCrypt no arquivo *auth.controller.ts* :

```
// Hashing da senha ao criar ou atualizar o usuário

const securePassword = await bcrypt.hash(newUserData.password, 10);

// Comparação da senha fornecida com a senha armazenada

const isPasswordValid = await bcrypt.compare(password, user.password);
```

O BCrypt é importante porque ele usa um algoritmo de hashing seguro e adaptável, o que significa que é mais resistente a ataques de força bruta ou dicionário em comparação com métodos de criptografia tradicionais.

6 CONSIDERAÇÕES FINAIS

O projeto desta Unidade Curricular de Extensão teve como objetivo criar um sistema de gestão de pagamentos para a Sociedade Francana de Instrução e Trabalho para Cegos, utilizando programação orientada a objetos com **Node.js** no backend e **React.js** no frontend. Este projeto buscou aplicar os fundamentos de desenvolvimento de software para criar uma solução prática e funcional que suporte as atividades da entidade social, promovendo eficiência e transparência nos processos financeiros.

Os procedimentos realizados incluíram pesquisa bibliográfica, análise dos processos operacionais da instituição e desenvolvimento do sistema. O software foi projetado para atender às necessidades específicas da organização, possibilitando uma transição gradual de processos manuais para práticas digitais.

Possibilidades para futuras:

- Expansão do sistema com funcionalidades adicionais, como integração com plataformas contábeis ou automação da folha de pagamento.
- Melhoria da interface do usuário para torná-la ainda mais intuitiva, especialmente considerando a acessibilidade para pessoas com deficiência visual.
- Adaptação do sistema para outras organizações com características e necessidades semelhantes, ampliando seu impacto social.

Possíveis limitações:

- **Resistência à mudança** por parte de usuários acostumados a processos manuais. Embora o sistema tenha sido projetado para ser simples e acessível, a implementação de práticas digitais exige um período de adaptação adequado.

Em resumo, o projeto alcançou seus objetivos iniciais, servindo como uma valiosa oportunidade de aprendizado acadêmico e uma contribuição para a Sociedade Francana de Instrução e Trabalho para Cegos. Ele representa uma base sólida para futuras expansões e melhorias, alinhando-se ao propósito de inovação social e tecnológica.

7 CONTRIBUIÇÕES DA UCE PARA A FORMAÇÃO DISCENTE

A Unidade Curricular de Extensão (UCE) é essencial para complementar a formação acadêmica dos alunos, conectando teoria e prática. O projeto do sistema de gestão de pagamentos para a Sociedade permitiu aplicar conceitos teóricos em uma solução prática, demonstrando o impacto real da tecnologia na sociedade.

Os estudantes vivenciaram o papel de desenvolvedores de software, criando uma solução que atende às necessidades específicas da comunidade, o que evidencia a importância da responsabilidade social e ética em TI.

Em resumo, a UCE ofereceu uma experiência de aprendizado valiosa, enriquecendo as carreiras dos alunos e aprofundando sua compreensão sobre o papel da tecnologia na sociedade.