CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE FRANCA

“Dr. THOMAZ NOVELINO”

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

SAMARA SILVA

CEPOWARE

SISTEMA DE ORGANIZAÇÃO E ADMINISTRAÇÃO DE SERVIÇOS

Trabalho de Graduação apresentado à Faculdade de Tecnologia de Franca - “Dr. Thomaz Novelino”, como parte dos requisitos obrigatórios para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Me. Claudio Eduardo Paiva

FRANCA/SP

2022

CEPOWARE

Sistema de Organização e Administração de Serviços

Samara Silva [1]

Resumo

A administração pode ser definida como o ato de gerenciar negócios, recursos ou pessoas com o objetivo de alcançar metas previamente especificadas. Sendo assim, a proposta deste trabalho é detalhar o desenvolvimento de um software para a administração de recursos, dados, finanças e clientes, visando auxiliar uma empresa de aplanação de cepos para a produção de calçados com os registros de seus serviços e finanças. A utilização de um software de controle em vez de registros físicos impressos ou escritos à mão contribui para a facilidade de uso, segurança e integridade das informações de uma empresa. A fim de cumprir com os requisitos estabelecidos pela empresa cliente foi realizado um estudo aprofundado sobre as principais ferramentas de elicitação de requisitos, assim como uma pesquisa sobre as ferramentas e recursos de desenvolvimento de software disponíveis para o ambiente *web*. O sistema Cepoware foi desenvolvido com uso de JavaScript, HTML e CSS para a criação da interface com o usuário, que foi anexada a um banco de dados SQLite. O software resultante é capaz de receber novos dados referentes aos serviços providenciados pela empresa, clientes e pagamentos, armazená-los, notificar o usuário sobre serviços ou pagamentos pendentes e calcular receitas e despesas. A interface do sistema possui um *layout* amigável e intuitivo. O programa pode ainda auxiliar a empresa com uma estimativa sobre o quanto ela pode faturar considerando todos os serviços ainda não pagos e pendentes.

Palavras-chave: Sistema Web. Cepo. Finanças. JavaScript. Administração.

Abstract:

Management can be defined as the act of managing businesses, resources or people, with the aim of achieving predefined goals. Therefore, the purpose behind this work is detailing the development of a software program for the administration of resources, data, finances and customers, aiming to help a chopping block planing company for shoemaking with the recording of it's services and finances. Utilizing a management software instead of a physical printed or handwritten record contributes to the ease of use, security and integrity of a company's information. In order to comply with the requirements established by the client company an in-depth study was carried out on the main requirement elicitation tools, as well as a research on the software development tools and resources available for the web environment. The Cepoware system was developed with JavaScript, HTML and CSS to create the user interface, which was then attached to a SQLite database. The resulting software is able of receiving new data regarding the services provided by the company, it's customers and payments, storing them, notifying the user of pending services or payments and calculating incomes and expenses. The system interface has a friendly and intuitive

1 Graduanda em Análise e Desenvolvimento de Sistemas pela Fatec Dr Thomaz Novelino – Franca/SP. Endereço eletrônico: samarasilvacandy@gmail.com

layout. The program can also aid the company with an estimate of how much it can earn considering all services that have yet to be paid and services that are still pending.

Keywords: *Web System. Chopping Block. Finances. Javascript. Management.*

1 Introdução

No processo de produção de calçados é necessário cortar as peças do sapato. Esse corte é realizado por meio da máquina balancim, onde são posicionados o material a ser cortado e, embaixo do mesmo, uma tábua de cepo para evitar que, tanto o material quanto as facas de corte do balancim sejam danificadas. O cepo é um material feito de polipropileno que serve como base de impacto durante o corte. Após ser utilizado diversas vezes o cepo fica danificado e entortado e precisa de manutenção para ser utilizado novamente com segurança.

A empresa cliente deste projeto de software é uma empresa especializada em providenciar manutenção para cepos de balancim. Ela aplaina o cepo para ser reutilizado pelas fábricas de calçado, que podem assim prosseguirem com a produção sem o risco de o cepo quebrar e/ou danificar seus materiais e máquinas de corte. Contudo, a empresa ainda não possui um software para registrar seus serviços realizados e finanças.

Possuir uma forma confiável de registrar o trabalho da empresa é importante e, de acordo com Eckford (2016, *online*), “a documentação inspira confiança em seus negócios, para os possíveis investidores e para seus clientes.”. Para o autor, a documentação mantém uma empresa consistente em seus serviços, protege sua integridade, provem informações importantes para a tomada de decisões e providencia oportunidades para a empresa. Por isso, manter uma documentação atualizada e cumprir metas baseadas nessa documentação passa uma boa imagem para os clientes, enquanto perder metas ou ser uma empresa desorganizada passa a imagem contrária.

O software Cepoware tem como objetivo facilitar o trabalho de documentação dos funcionários da empresa, auxiliando no processo de registro de seus clientes, serviços, gastos e finanças. Esse objetivo é alcançado com o cadastro dessas informações em um banco de dados e do resgate deles quando necessário. Além disso, o software emite notificações ao usuário para mantê-lo atento às datas de entrega de serviços e recebimento de pagamentos.

O sistema foi desenvolvido utilizando a linguagem de programação JavaScript com Node.JS, HTML e CSS.

2 Levantamento de Requisitos

O processo de Levantamento de Requisitos é uma etapa crucial do desenvolvimento de software. Antes de se desenvolver um produto, é necessário compreender corretamente quais são as necessidades do cliente.

Sommerville (2003) propõe um processo genérico de levantamento e análise de requisitos que contém os seguintes passos:

- Compreensão do domínio: os analistas devem desenvolver seus conhecimentos sobre o domínio da aplicação, compreendendo o negócio e empresa no qual seu software está inserido para determinar qual o problema a ser resolvido.
- Coleta de requisitos: é necessário interagir com os *stakeholders* do sistema para descobrir seus requisitos, a compreensão do domínio se desenvolve mais durante essa coleta.
- Classificação: a classificação considera o conjunto não estruturado dos requisitos e os organiza em grupos coerentes.
- Resolução de conflitos: quando múltiplos *stakeholders* estão envolvidos, os requisitos poderão apresentar conflitos. Deve-se solucionar esses conflitos de modo a satisfazer o maior número possível de *stakeholders*.
- Definição das prioridades: em qualquer conjunto de requisitos, alguns serão mais importantes do que outros. Esse estágio envolve interação com os *stakeholders* para a definição dos requisitos mais importantes.
- Verificação de requisitos: os requisitos são verificados para descobrir se estão completos e consistentes e se estão em concordância com o que os *stakeholders* desejam do sistema.

A seguir, é apresentado o processo utilizado para realizar a elicitação dos requisitos funcionais do software Cepoware e os diagramas gerados.

2.1 Elicitação e Especificação dos Requisitos:

O sistema Cepoware foi pensado e construído através do contato direto com os *stakeholders*: trabalhadores da microempresa, incluindo o dono. Foram utilizadas técnicas de entrevistas fechadas e abertas, etnografia e criação de cenários para que os trabalhadores pudessem visualizar o uso do software com maior facilidade.

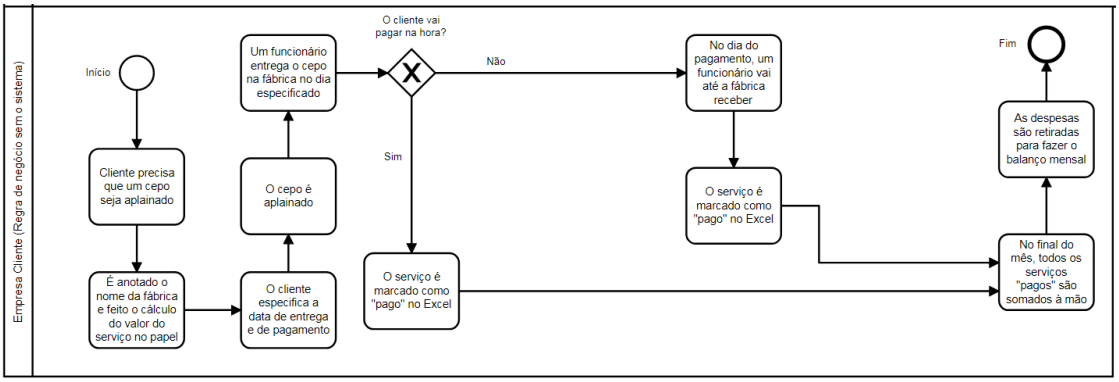
A etnografia foi utilizada para melhor compreender o ambiente, a cultura e o modo de trabalho da empresa. Portanto o local de trabalho foi visitado e os processos realizados pelos funcionários durante o dia a dia foram observados, anotados e também executados pela autora do software antes da definição dos requisitos. Isso foi feito para que estes, quando definidos, se encaixassem o mais precisamente possível com as necessidades observadas.

Os *stakeholders* requerem um projeto baseado na *web*, com a cor temática verde, em que possam registar e consultar clientes, serviços, gastos, ganhos, o lucro total da empresa e receber notificações sobre datas de entrega e datas de pagamento de serviços.

2.2 BPMN

O BPMN (*Business Process Model and Notation)* é uma representação dos processos realizados durante o dia a dia de uma empresa, possibilitando um melhor entendimento de suas dependências e facilitando a atualização e implementação de um software que satisfaça as necessidades do negócio. Ele auxilia o usuário final a compreender as funções, o uso do software e seu papel nos processos diários da corporação. Na figura 1 é apresentado o modelo dos negócios da empresa cliente, sem incluir o software Cepoware:

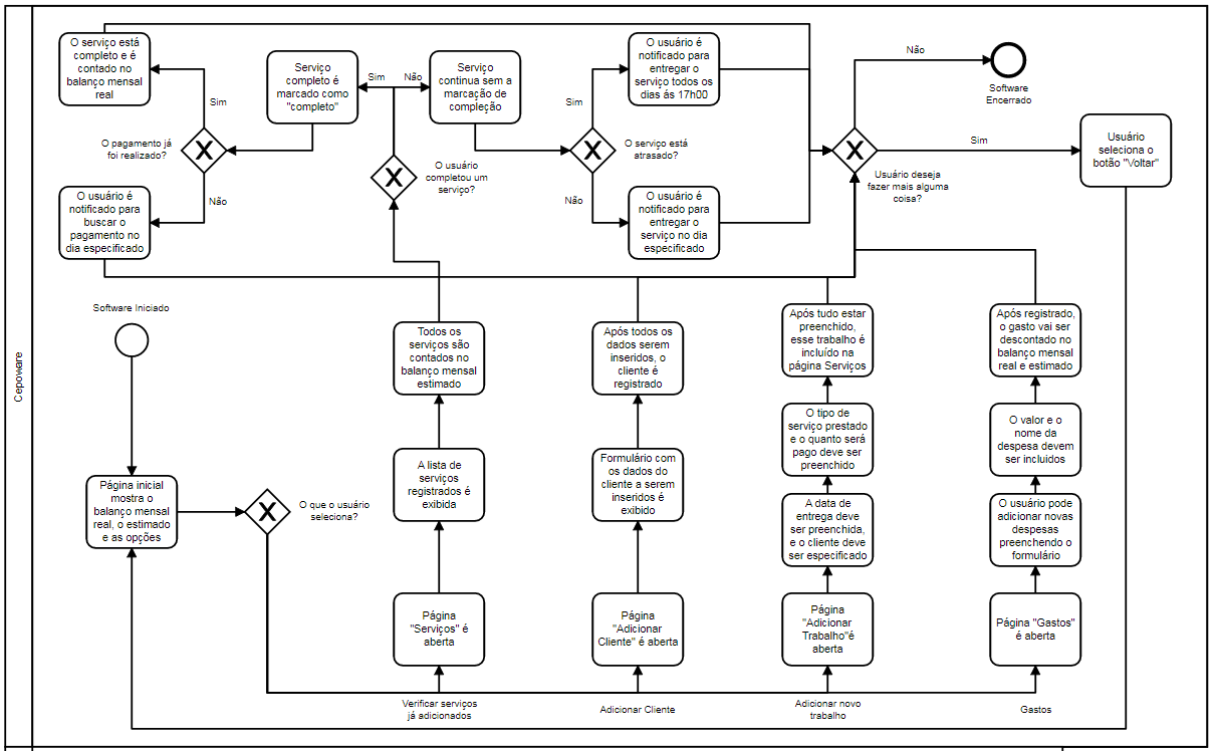
Figura 1: BPMN da empresa cliente



Fonte: Autora

Na figura 2 é apresentado o modelo BPMN do funcionamento do software Cepoware.

Figura 2: BPMN do software Cepoware



Fonte: Autora

2.3 Requisitos Funcionais

De acordo com Valente (2020, p.79), os Requisitos Funcionais se referem “ao que um sistema deve fazer”, ou seja, suas funcionalidades. Eles devem ser especificados em linguagem natural e direta. No Quadro 1 encontram-se os requisitos funcionais do software Cepoware.

Quadro 1 – Requisitos Funcionais do Sistema Cepoware

RF001- Calcular o lucro real.	Categoria: (X) Oculto ( )Evidente	Prioridade: ( ) Altíssima (X) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve calcular o lucro real da empresa baseado nos serviços que já foram pagos e nas despesas.		
RF002- Calcular o lucro estimado.	Categoria: (X) Oculto ( ) Evidente	Prioridade: ( ) Altíssima ( ) Alta ( ) Média

		(X) Baixa
Descrição: O sistema deve calcular o lucro estimado da empresa, baseado em todos os serviços registrados (pagos, não pagos, pendentes, atrasados e completos) e nas despesas, gerando uma estimativa de quanto dinheiro a empresa possuirá se todos os serviços pendentes forem finalizados e pagos.		
RF003- Criar botões, ex: "Administrar Serviços", "Administrar Clientes", "Administrar Gastos".	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve ter botões através dos quais o usuário pode acessar as páginas do software e suas respectivas funcionalidades.		
RF004- Inserir valores numéricos.	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve permitir que o usuário insira valores numéricos nas páginas de "Gastos", "Adicionar Serviços" e "Adicionar Cliente".		
RF005- Inserir textos.	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deverá permitir a inserção de dados em forma de texto nas páginas de "Adicionar Gastos", "Adicionar Serviços" e "Adicionar Cliente". O usuário irá inserir textos no software para registrar informações sobre o serviço\cliente\gasto em questão.		
RF006- Criar lista de serviços.	Categoria: ( ) Oculto (X) Evidente	Prioridade: (X) Altíssima ( ) Alta ( ) Média ( ) Baixa
Descrição: O sistema deve criar uma lista de fichas com os serviços registrados pelo usuário, uma embaixo da outra, e exibi-la na aba de Serviços. Cada ficha contém o tipo de serviço, valor do serviço, sua data de entrega e sua data de pagamento.		
RF007- Notificar usuário sobre entrega de serviços	Categoria: ( ) Oculto (X) Evidente	Prioridade: ( ) Altíssima (X) Alta ( ) Média ( ) Baixa
Descrição: O software deve mandar uma notificação ao usuário no dia em que o cepo deve ser pego ou entregue, com uma hora de antecedência. Caso o serviço esteja atrasado, o usuário será notificado todos os dias às 17h00.		
RF008- Notificar usuário sobre a coleta de pagamentos	Categoria: ( ) Oculto (X) Evidente	Prioridade: ( ) Altíssima (X) Alta ( ) Média ( ) Baixa
Descrição: O software deve mandar uma notificação ao usuário no dia em que o pagamento deve ser coletado, com uma hora de antecedência.		

Fonte – Autora

2.4 Requisitos Não Funcionais

Valente (2020, p.79) explica que os Requisitos Não Funcionais devem ser especificados de forma quantitativa, usando-se de métricas como: confiabilidade (% de disponibilidade, tempo médio entre falhas), usabilidade (tempo necessário de treinamento para os usuários), robustez (tempo para recuperação do sistema após falhas, probabilidade de perca de dados durante uma falha), entre outros.

Tais requisitos não funcionais são restrições nas funções e serviços oferecidos pelo sistema e podem incluir restrições de tempo, restrições no processo de desenvolvimento e restrições impostas por normas (SOMMERVILLE, 2003). No Quadro 2 encontram-se os requisitos não funcionais do sistema Cepoware. O Quadro 3 mostra a matriz de rastreabilidade entre requisitos funcionais e requisitos não funcionais do software.

Quadro 2 – Requisitos Não Funcionais do sistema

RNF 001 - Paleta de cores	As cores das páginas do software devem ser primariamente na cor verde.	Tipo Produto	(X) Desejável ( ) Obrigatório	(X)Permanente ( ) Transitório
RNF 002 - Notificações	As notificações devem ser enviadas ao usuário com pelo menos uma hora de antecedência.	Tipo Usabilidade	( ) Desejável (X) Obrigatório	(X)Permanente ( ) Transitório
RNF 003 - Compatibilidade Android	O software deve ser compatível com sistemas Android da marca Samsung.	Tipo Compatibilidade	( ) Desejável (X) Obrigatório	(X)Permanente ( ) Transitório
RNF 004 - Programação em Java Script	Deverá ser usado principalmente o JavaScript para construção do sistema.	Tipo Produto	(X) Desejável ( ) Obrigatório	(X)Permanente ( ) Transitório
RNF 005 – Minimalismo	O software e suas funcionalidades devem ser confeccionados de forma minimalista para que o usuário consiga utilizar o produto com maior facilidade.	Tipo Produto	(X) Desejável ( ) Obrigatório	(X)Permanente ( ) Transitório
RNF 006 - Sistema na Web	O sistema deve funcionar na web, para que o usuário possa acessá-lo através de um *browser*.	Tipo Usabilidade	(X) Desejável ( ) Obrigatório	(X) Permanente ( ) Transitório

Fonte – Autora

Quadro 3 - Matriz de Rastreabilidade 1

	RF001	RF002	RF003	RF004	RF005	RF006	RF007	RF008
RNF001	X	X	X	X	X	X	X	X
RNF002							X	X
RNF003	X	X	X	X	X	X	X	X
RNF004	X	X	X	X	X	X	X	X
RNF005	X	X	X	X	X	X	X	X
RNF006	X	X	X	X	X	X	X	X

Fonte – Autora

2.5 Regras de Negócio

As regras de negócio são diretrizes da empresa que definem ou restringem ações, especificando limites para operações. Elas demonstram como as tarefas da empresa devem ser conduzidas (RIBEIRO, 2020). No Quadro 4 encontram-se as regras de negócio da empresa cliente e no Quadro 5 está a matriz de rastreabilidade relacionando as regras de negócio com os requisitos funcionais do software Cepoware.

Quadro 4 – Regras de Negócio da empresa cliente.

RN001 – Custo dependente do Tipo de Serviço
Descrição: Cepos pequenos custam menos para aplainar do que cepos grandes, colagens de cepo custam mais do que cepos grandes. A criação de novos cepos custa mais do que um cepo grande, porém menos que uma colagem.
RN002 – Desconto para novos clientes
Descrição: Não é cobrado o mesmo valor para todos os clientes. Novos clientes ganham descontos para todos os serviços.
RN003 – Custo de busca e de entrega
Descrição: É cobrada uma taxa a mais se o cliente preferir que o cepo seja transportado por um funcionário da empresa.
RN004 – Prazo de entrega do serviço
Descrição: Dependendo de quando um cliente pedir por um serviço, o cepo pode ser entregue somente dias depois, em vez de no mesmo dia ou no dia seguinte.
RN005 – Desconto dependente de quantidade
Descrição: Dependendo da quantidade de cepos que um cliente queira que sejam aplainados por vez, um desconto pode ser negociado.

Fonte – Autora

Quadro 5 – Matriz de Rastreabilidade 2

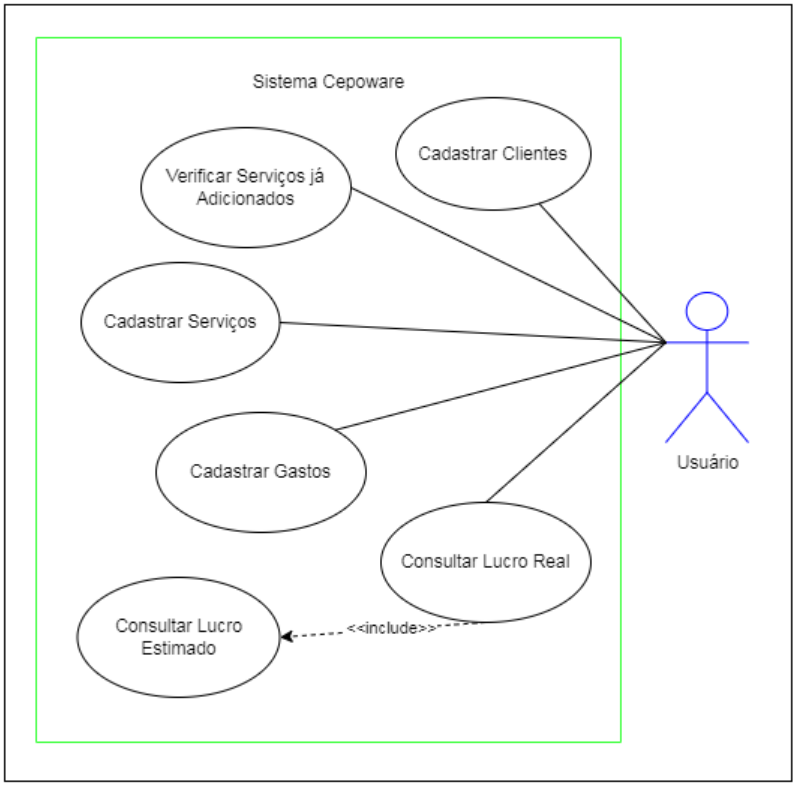
	RF001	RF002	RF003	RF004	RF005	RF006	RF007	RF008
RN001	X	X	X	X	X	X		
RN002	X	X	X	X	X	X		
RN003	X	X	X	X	X	X		
RN004			X			X	X	X
RN005	X	X	X	X	X	X		

Fonte – Autora

2.6 Casos de Uso

Os casos de uso são documentos de especificação de requisitos, representando-os de forma textual. Eles incluem descrições detalhadas na perspectiva de um ator usuário do software, sendo este uma entidade externa ao sistema (VALENTE, 2020). A Figura 3 apresenta o diagrama de casos de uso do sistema e no Quadro 6 está a matriz de rastreabilidade entre casos de uso e requisitos funcionais.

Figura 3 – Diagrama de casos de uso do sistema Cepoware



Fonte – Autora

Quadro 6 - Matriz de Rastreabilidade 3

	RF001	RF002	RF003	RF004	RF005	RF006	RF007	RF008
UC001			X	X	X	X		
UC002	X	X	X	X	X	X	X	X
UC003	X	X	X	X	X	X		
UC004	X	X				X		
UC005	X	X	X			X	X	X

Fonte – Autora

Do Quadro 7 ao Quadro 12 encontram-se as especificações dos Casos de Uso do software Cepoware.

Quadro 7 – Caso de Uso: Cadastrar novo cliente

Caso de Uso – Cadastrar novo cliente	
ID	UC 001
Descrição	O usuário cadastra um novo cliente.
Ator Primário	Usuário
Pré-condição	Não possui
Cenário Principal	1. O usuário entra no sistema e seleciona o botão “Administrar Clientes” 2. Na página “Administrar Clientes”, o usuário seleciona “Cadastrar novo Cliente” 3. O usuário preenche o campo do nome da empresa 4. Insere também o(s) telefone(s) e o endereço da empresa 5. Aperta o botão de “Salvar Cliente” 5. O sistema registra o novo cliente no banco de dados.
Pós-condição	Para UC 002 – Adicionar serviços
Cenário Alternativo	*3a – Em vez de colocar o nome da empresa, o usuário insere o nome de um funcionário da empresa.
Inclusão	Não possui
Extensão	Não possui

Fonte – Autora

Quadro 8 – Caso de Uso: Adicionar Serviço

Caso de Uso – Adicionar Serviço	
ID	UC 002
Descrição	O usuário recebeu um novo serviço de um cliente e vai registrá-lo no sistema.
Ator Primário	Usuário
Pré-condição	UC 001 – Cadastrar novo cliente

Cenário Principal	1. O usuário entra no sistema e seleciona o botão “Administrar Serviços” 2. Na página “Administrar Serviços”, o usuário seleciona “Adicionar novo serviço” 3. Escolhe o tipo de serviço (Ex: Se é um cepo grande, colagem, cepo redondo etc.) dentre os tipos já cadastrados 4. Insere o valor do serviço em reais 5. Escolhe o cliente que pediu o serviço dentre os clientes já cadastrados 6. Insere a data de entrega 7. Aperta o botão de “Salvar Serviço” 8. O sistema registra o serviço no banco de dados. Ele agora vai aparecer na aba Serviços 9. Nos dias especificados, o sistema manda uma notificação para o usuário entregar o serviço e\ou receber seu pagamento
Pós-condição	Não possui
Cenário Alternativo	6a. Especifica que não há data de entrega, pois o serviço já foi completado com antecedência
Inclusão	Não possui
Extensão	Não possui

Fonte – Autora

Quadro 9 – Caso de Uso: Registrar Gastos

Caso de Uso – Registrar gastos	
ID	UC 003
Descrição	O usuário insere os gastos da empresa no sistema para que possa ser feito o cálculo do lucro real.
Ator Primário	Usuário
Pré-condição	Não possui
Cenário Principal	1. O usuário entra no sistema e seleciona o botão “Administrar Gastos” 2. Na página “Administrar Gastos”, o usuário seleciona “Adicionar novo gasto” 3. Insere o nome do gasto (Ex: Cola, energia, amolação de facas, manutenção de máquinas) 4. Insere o valor do gasto 5. Aperta o botão “Salvar Gasto” 6. O sistema registra o gasto no banco de dados. Ele agora aparece na lista de gastos da página 7. O sistema agora contabiliza esse gasto no cálculo do lucro total
Pós-condição	UC 004 – Consultar Lucro Real
Cenário Alternativo	Não possui

Inclusão	Não possui
Extensão	Não possui

Fonte – Autora

Quadro 10 – Caso de Uso: Consultar Lucro Real

Caso de Uso – Consultar Lucro Real	
ID	UC 004
Descrição	O usuário verifica o lucro real da empresa através do software.
Ator Primário	Usuário
Pré-condição	UC 003 – Registrar Gastos, UC 005 – Verificar lista de serviços já adicionados
Cenário Principal	1.O usuário abre o software e vê o lucro real calculado através dos serviços já pagos.
Pós-condição	Não possui
Cenário Alternativo	Não possui
Inclusão	UC 006
Extensão	Não possui

Fonte – Autora

Quadro 11 – Caso de Uso: Verificar lista de serviços já adicionados

Caso de Uso – Verificar lista de serviços já adicionados	
ID	UC 005
Descrição	O usuário verifica todos os serviços pendentes, atrasados e completos da empresa através de uma lista no software.
Ator Primário	Usuário
Pré-condição	UC 002 – Registrar Serviços
Cenário Principal	1. O usuário entra no sistema e seleciona o botão “Administrar Serviços” 2. O sistema gera uma lista dos serviços registrados 3. O usuário encontra um dos serviços pendentes na lista 4. O usuário marca o serviço pendente como “completo”
Pós-condição	UC 004 – Consultar Lucro Real
Cenário Alternativo	4a – Em vez de finalizar o serviço, ele apenas verifica a data de entrega 5 – O sistema notifica o usuário para que o serviço seja entregue no dia correto 6 – O usuário marca o serviço pendente como “completo” após finalizá-lo
Inclusão	Não possui
Extensão	Não possui

Fonte – Autora

Quadro 12 – Caso de Uso: Consultar o Lucro Estimado

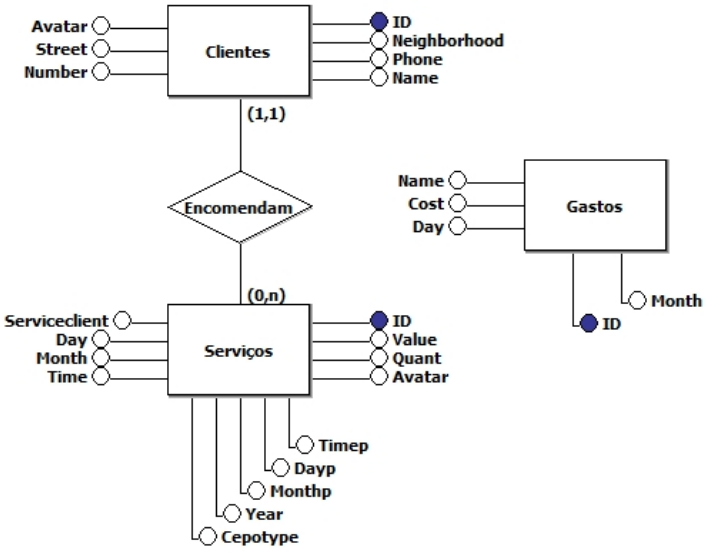
Caso de Uso – Consultar o Lucro Estimado	
ID	UC 006
Descrição	O usuário verifica o lucro estimado da empresa através do software.
Ator Primário	Usuário
Pré-condição	UC 003 – Registrar Gastos, UC 005 – Verificar lista de serviços já adicionados
Cenário Principal	1.O usuário abre o software e vê o lucro estimado calculado através dos serviços pagos e não pagos da empresa.
Pós-condição	Não possui
Cenário Alternativo	Não possui
Inclusão	Não possui
Extensão	Não possui

Fonte – Autora

2.7 Diagrama Entidade-Relacionamento

Os diagramas entidade-relacionamento são representações das informações armazenadas por um dado sistema ou empresa, criados antes do banco de dados físico como forma de visualizar as estruturas e o escopo do banco para depois colocá-lo em prática (OLIVEIRA, 2002). A Figura 4 traz a representação conceitual do banco de dados do software Cepoware.

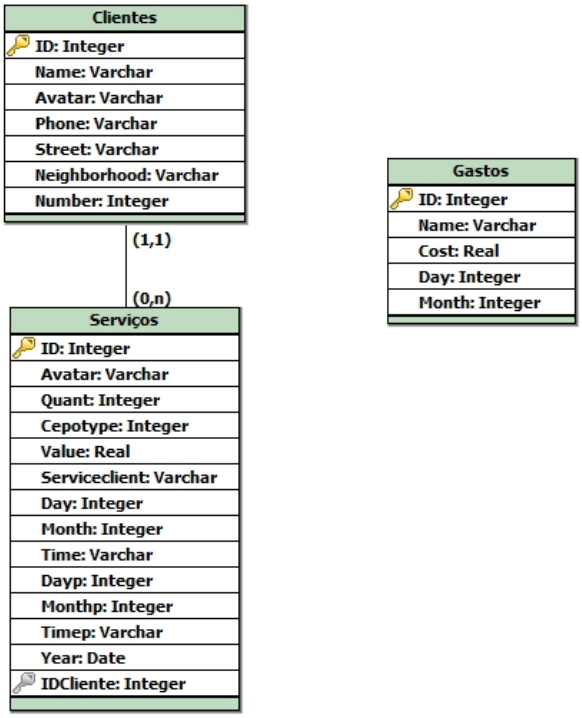
Figura 4 – Diagrama do modelo conceitual do banco de dados do sistema Cepoware



Fonte: Autora

A Figura 5 representa um diagrama do modelo lógico do banco dados, incluindo maiores detalhes.

Figura 5 – Diagrama do modelo lógico do banco de dados do sistema Cepoware



Fonte: Autora

3 Ferramentas e Métodos ou Desenvolvimento

Nesta seção estão descritas as linguagens e ferramentas utilizadas para o desenvolvimento deste projeto de software e seu método de desenvolvimento com exemplos de funcionalidades e do código fonte.

3.1 JavaScript

O JavaScript é uma das linguagens mais populares no mercado atual e compõe grande parte dos sites na internet. De acordo com Silva (2010) ela é utilizada em milhões de páginas web no mundo todo para validar formulários, detectar objetos e adicionar funcionalidades interativas. Ela foi escolhida por sua versatilidade e compatibilidade com diversas plataformas. Criada para servir como uma linguagem que “colaria” componentes como imagens e *plugins*, o JavaScript é bem adequado para um projeto baseado no desenvolvimento *web.*

3.2 HTML

A linguagem HTML (*Hyper Text Marking Language*) tem sido empregada para o desenvolvimento de sites da web desde os primórdios da internet e, ainda hoje, é vastamente utilizada. Os códigos compondo o *frontend* do software Cepoware foram construídos com HTML. Ele foi criado com base nos *links* de hipertexto que conectam as páginas da *web* entre si, estabelecendo uma estrutura de navegação entre elas e assim sendo chamado de “Linguagem de Marcação de Hiper Texto”. (MILLETO; BERTAGNOLLI, 2014).

3.3 CSS

O CSS (*Cascading Style Sheet*) foi utilizado para estilizar os elementos do site, criando componentes visuais juntamente com o HTML. Criado em 1996, o CSS se popularizou em massa e domina as páginas da internet moderna, pois permite a criação de páginas atrativas e interativas com estilos personalizados pelo desenvolvedor, customizando textos, imagens, listas, gráficos e assim em diante (MILLETO; BERTAGNOLLI, 2014). O uso de CSS para definir cores, fontes, alinhamentos e elementos gráficos é um dos fatores que diferencia os sites modernos dos primeiros sites que foram criados apenas com o uso do HTML.

3.4 SQLite

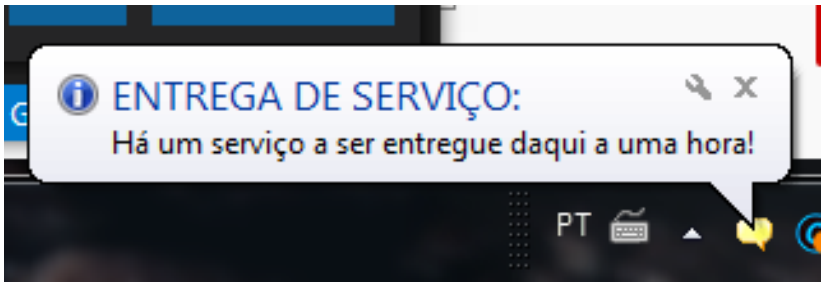
O SQLite é considerado um banco de dados leve e rápido e que vem embutido em computadores e aparelhos *mobile*. Seu formato de arquivo é estável e pode facilmente ser transferido de uma plataforma para outra, tendo compatibilidade com versões antigas. Por conta da necessidade de desenvolver este projeto com pouca experiência, esse banco de dados foi escolhido por ter uma documentação acessível e ser intuitivo, além de ser uma escolha popular para o armazenamento de dados por longos períodos. O código fonte do SQLite é de domínio público e livre para ser utilizado para quaisquer propósitos (BHOSALE; PATIL; PATIL, 2015).

3.5 Node-Notifier

O Node-Notifier é uma biblioteca que permite a criação de alertas. Esses alertas podem ser utilizados para oferecer um *feedback* para o usuário ou para solicitar informações (ADAOBI, 2020). As notificações funcionam até mesmo em versões anteriores ao Windows 11, algo necessário neste projeto para um bom funcionamento nos computadores da microempresa cliente, que utilizam principalmente o Windows na sua versão 7.

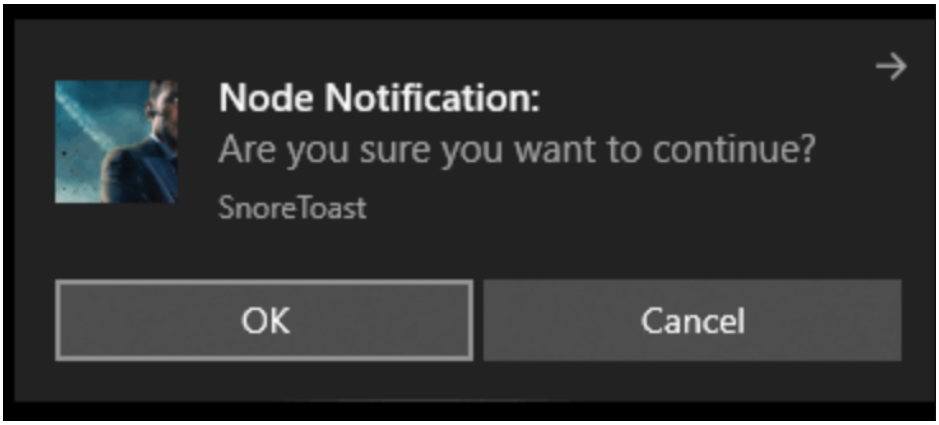
Os alertas também funcionam em aparelhos *mobile* e podem ser customizados de diversas formas para uma melhor experiência. A versão do Node-Notifier utilizada no projeto é a 10.0.0. As figuras a seguir mostram exemplos de alertas gerados pelo Node-Notifier, no Windows 7 e Windows 10, respectivamente.

Figura 6 – Exemplo de alerta do Node-Notifier no Windows 7



Fonte – Autora

Figura 7 – Exemplo de alerta do Node-Notifier no Windows 10



Fonte – NPMJS, Node-Notifier Package, 2022

Cada alerta do Node-Notifier é tratado como um objeto e pode ser customizado livremente. Uma mensagem simples pode ser composta apenas pelo seu título, mensagem e som. O código fonte responsável pela criação de um exemplo de alerta pode ser visualizado na Figura 8.

Figura 8 – Função de abertura do software Node-Notifier

```
notifier.notify({
        title: "Bem-Vindo ao Cepoware",
        message: "Serviços abertos",
        sound: true
    });
```

Fonte – Autora

3.6 Node JS

O Node JS é um software de código aberto debutado em 2009, e serve para que a linguagem JavaScript seja executada diretamente no servidor. Utilizando-se do interpretador V8, ele conversa diretamente com a máquina virtual do Google Chrome, permitindo um ganho de performance e que aplicações *web* sejam escritas com apenas uma linguagem, tanto na parte do cliente quanto na parte do servidor (CANTELON et al., 2014). Ele foi utilizado em conjunto com o *framework* Express para criar as funcionalidades do software Cepoware por conta de sua adaptabilidade, grande comunidade ativa de desenvolvedores e a possibilidade de se utilizar os mesmos códigos em diversas plataformas. Neste projeto a versão utilizada do Node JS foi a 12.19.0.

3.7 Métodos ou Desenvolvimento

O software foi desenvolvido em ciclos semanais de acordo com um cronograma. Primeiramente foram definidas as funcionalidades necessárias fazendo uso de entrevistas com o funcionário designado como *Product Owner* e demais funcionários da empresa. Com as informações coletadas foi construído um protótipo das páginas do software utilizando-se o Figma.

O protótipo foi apresentado para alguns dos futuros usuários, que fizeram sugestões de alterações no seu *layout*, como mudanças nos tamanhos e cores de certos componentes das telas. Tais solicitações foram consideradas e registradas como requisitos antes que as páginas fossem codificadas de fato em HTML e CSS.

O *frontend* do software foi desenvolvido primeiro, para depois as funcionalidades serem inseridas.

Para o armazenamento de dados foi utilizado o SQLite. O banco de dados é criado e administrado através do próprio VSCode. As funções de cadastrar clientes,

serviços, gastos, calcular o lucro e gerar notificações interagem diretamente com o banco de dados e com as suas tabelas. A Figura 9 mostra um exemplo de Função que retorna uma lista de todos os clientes cadastrados.

Figura 9 – Função que retorna uma lista de todos os clientes cadastrados

```
async function pageClients(req, res) {
    const filters = req.query

    const query = `
        SELECT clients.*
        FROM clients
    `
    //Caso haja erro na hora da consulta ao banco de dados
    try {
        const db = await Database
        const clients = await db.all(query)

        return res.render('clients.html', { clients, filtros, filters, or-
ders })

    } catch (error) {
        console.log(error)
    }
}
```

Fonte – Autora

Pela constante “query”, são enviados comandos para o SQLite, que os interpreta e retorna dados das tabelas inseridas no banco de dados. A Figura 10 mostra exemplos de códigos executados no banco de dados SQLite.

Figura 10 - Exemplo que inclui as funções “SUM” e “ROUND” do SQLite:

```
const query = `
        SELECT ROUND((SELECT SUM(value) FROM services WHERE paid = 1) -
        (SELECT SUM(cost) FROM expenses), 2)
    `
    const query2 = `
        SELECT ROUND((SELECT SUM(value) FROM services) -
        (SELECT SUM(cost) FROM expenses), 2)
    `
```

Fonte – Autora

O *backend* levou mais tempo para ser desenvolvido do que o *frontend,* devido a uma maior necessidade de adaptação e estudos sobre servidores, rotas e interação

com o banco de dados. A Figura 11 mostra um exemplo de código utilizado para cadastrar um cliente.

Figura 11 – HTML da página de Cadastro de Clientes

```
<div id="container">
        <header class="page-header">
            <div class="top-bar-container">
                <a href="/clients">
                <img src="/images/icons/back.png" alt="Voltar"></a>
                <img src="/images/logo.png" alt="Cepoware">
            </div>
            <div class="header-content">
                <strong>Cadastro de Clientes:</strong>
                <p>Preencha os campos para inserir o novo cliente.</p>
            </div>
        </header>
        <main>
            <form action="/save-clients" id="create-client" method="POST">
                <fieldset>
                    <legend>Dados do cliente:</legend>
                    <div class="input-block">
                        <label for="name">Nome completo:</label>
                        <input name="name" id="name" required>
                    </div>
                    <div class="input-block" id="phonenumber">
                        <label for="phone">Telefone <small> (Somente números):
</small>
                            <button type="button" id="add-phone"> + Novo Tele-
fone</button>
                        </label>
                        <input name="phone" id="phone" type="number" required>
                    </div>
                </fieldset>
                <fieldset id="address-items">
                    <legend>Endereço:</legend>
                    <div class="address-item">
                    <div class="input-block">
                        <label for="street">Rua:</label>
                        <input name="street" id="street" required>
                    </div>
                    <div class="input-block">
                        <label for="neighborhood">Bairro:</label>
                        <input name="neighborhood" id="neighborhood" required>
                    </div>
                    <div class="input-block">
                        <label for="number">Número:</label>
                        <input name="number" id="number" required>
```

```
                    </div>
                    </div>
                </fieldset>
                <fieldset>
                <div class="input-block">
                    <label for="avatar">Link do ícone:<small> *Opcional (Com-
ece com https://):</small></label>
                    <input name="avatar" id="avatar" type="url">
                </div>
                </fieldset>
            </form>
            <footer>
                <p>
                    <img src="/images/icons/warning.svg" alt="Aviso Im-
portante">
                    Importante! <br>
                    Preencha todos os dados.
                </p>
                <button type="submit" form="create-client">Salvar Cadas-
tro</button>
            </footer>
        </main>
    </div
```

Fonte – Autora

Ao se cadastrar um novo cliente, o botão “submit” envia os dados para a função de cadastro mostrada na Figura 12.

Figura 12 – Função *saveClients*

```
async function saveClients(req, res) {
    const createRegistry = require('./database/createRegistry');

    const clientValue = {
        name: req.body.name,
        avatar: req.body.avatar,
        phone: req.body.phone,
        street: req.body.street,
        neighborhood: req.body.neighborhood,
        number: req.body.number
    };

    try {
        const db = await Database;
        await createRegistry(db, { clientValue });

        return res.redirect("/clients");
```

```
    } catch (error) {
        console.log(error);
    }
}
```

Fonte – Autora

A função *saveClients* manda os dados inseridos para o banco de dados através da função *createRegistry* e redireciona o usuário para a página que contém a lista de clientes já inseridos. A Figura 13 mostra a função *createRegistry*.

Figura 13 – Função *createRegistry*

```
module.exports = async function(db, { clientValue }) {
    //Inserir dados na table de clients
    const insertedClient = await db.run(`
        INSERT INTO clients (
            name,
            avatar,
            phone,
            street,
            neighborhood,
            number
        ) VALUES (
            "${clientValue.name}",
            "${clientValue.avatar}",
            "${clientValue.phone}",
            "${clientValue.street}",
            "${clientValue.neighborhood}",
            "${clientValue.number}"
        );`
    );
};
```

Fonte – Autora

Assim, o cliente é inserido no banco de dados e pode ser encontrado pelo usuário imediatamente.

Este processo de cadastro segue o mesmo padrão para inserção de gastos e serviços, contendo uma página de coleta de informações, uma função para o processamento dos dados e redirecionamento do usuário, e uma função para a inserção no banco de dados.

Além disso, o cadastro de serviços possui funções extras para que alertas sejam enviados ao usuário quando o horário de pagamento ou entrega se aproxima. A Figura 14 mostra um exemplo de função de alerta criada através do node-notifier.

Figura 14 – Função de alerta de entrega Node-Notifier

```
for (var i = 0, len = services.length; i < len; i++) {
       if (services[i][paid] == 0 && services[i][time] >= hour && ser-
vices[i][day] == date && services[i][month] == mes && ser-
vices[i][year] == ano){
                notifier.notify({
                title: "ENTREGA DE SERVIÇO:",
                message: "Há um serviço a ser entregue daqui a uma hora! ",
                sound: true
                });
       }
}
```

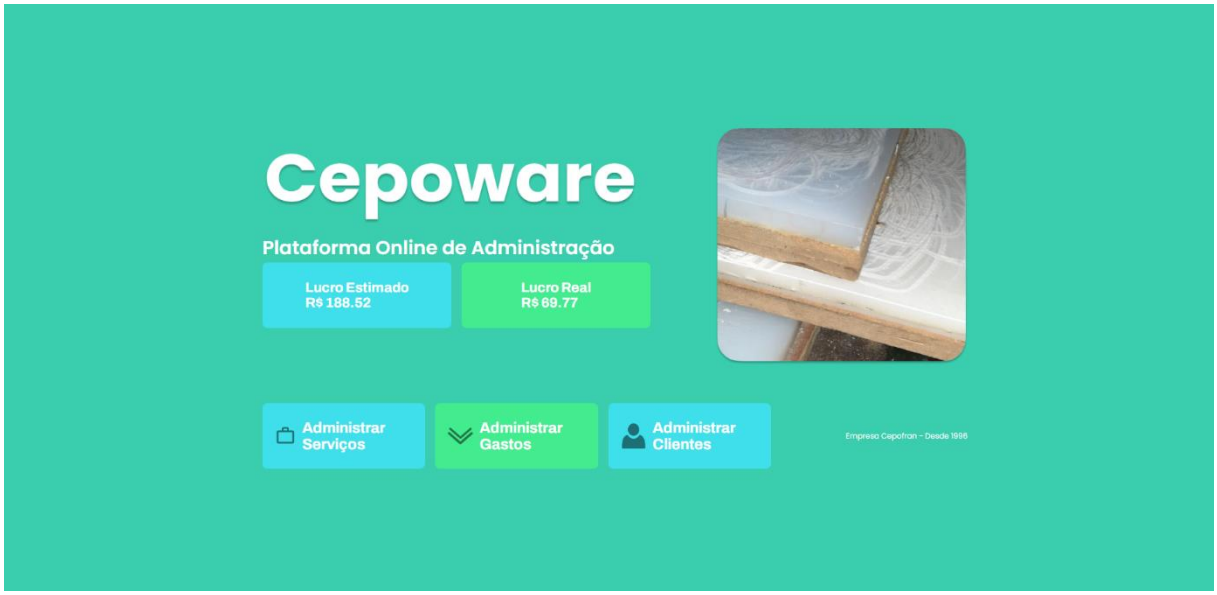
Fonte – Autora

4 Resultados e Discussão

Após entrevistar os funcionários da empresa, realizar a elicitação de requisitos e documentá-los, definir todas as ferramentas e os métodos de desenvolvimento, foram elaboradas as telas apresentadas da Figura 15 a Figura 30, assim como breves explicações de suas funcionalidades.

A Figura 15 exibe a página inicial do software quando executado em modo *desktop*. Nesta tela estão disponíveis os caminhos para Administração de Serviços, Administração de Gastos e Administração de Clientes.

Figura 15 – Página Inicial Desktop



Fonte – Autora

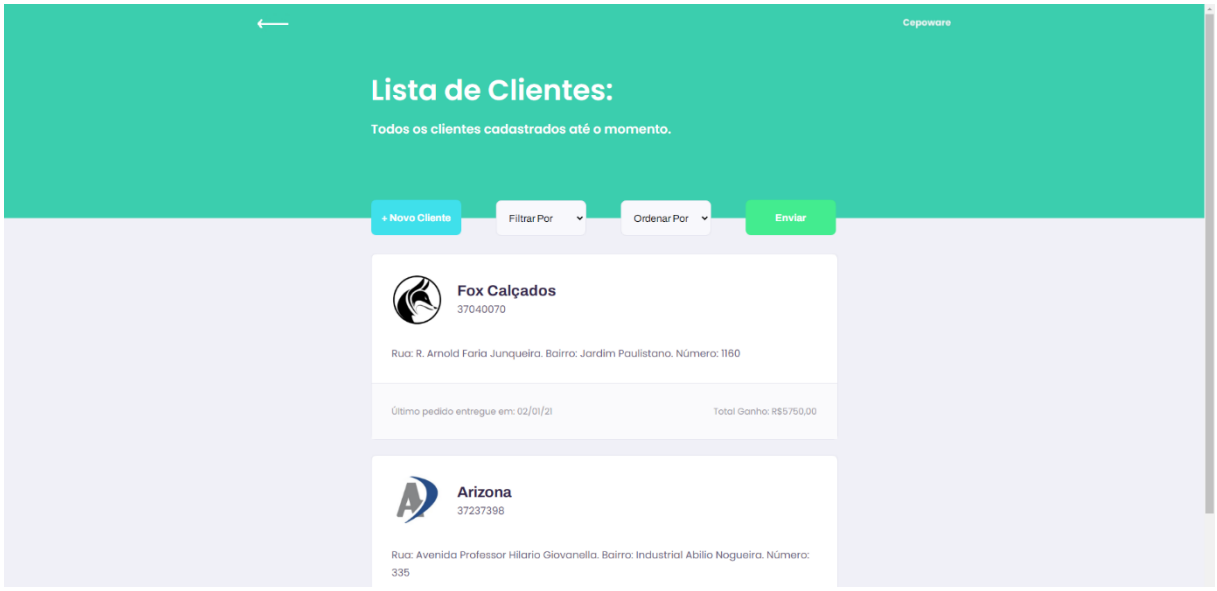
A Figura 16 exibe a página inicial do Cepoware em modo *mobile*.

Figura 16 – Página Inicial Mobile

Fonte – Autora

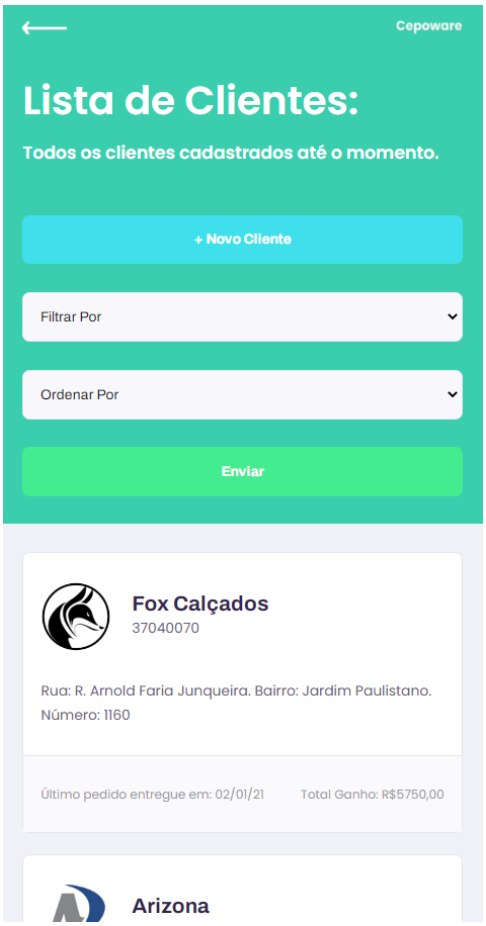
A Figura 17 apresenta a lista de clientes acessada pelo botão “Administrar Clientes” em sua versão *desktop*. Nesta tela é possível visualizar os clientes cadastrados e acessar a página para inserção de novos clientes. A Figura 18 demonstra a versão *mobile* da mesma funcionalidade.

Figura 17 – Lista de Clientes



Fonte – Autora

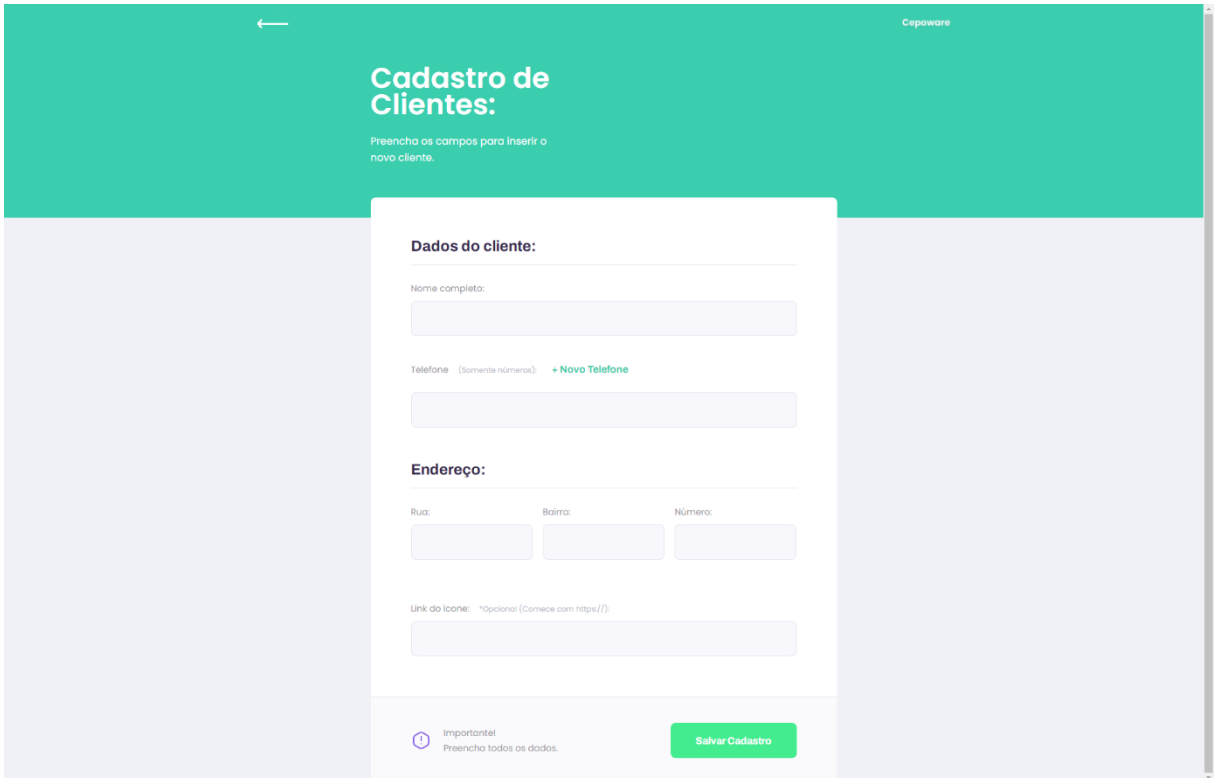
Figura 18 – Lista de Clientes Mobile



Fonte – Autora

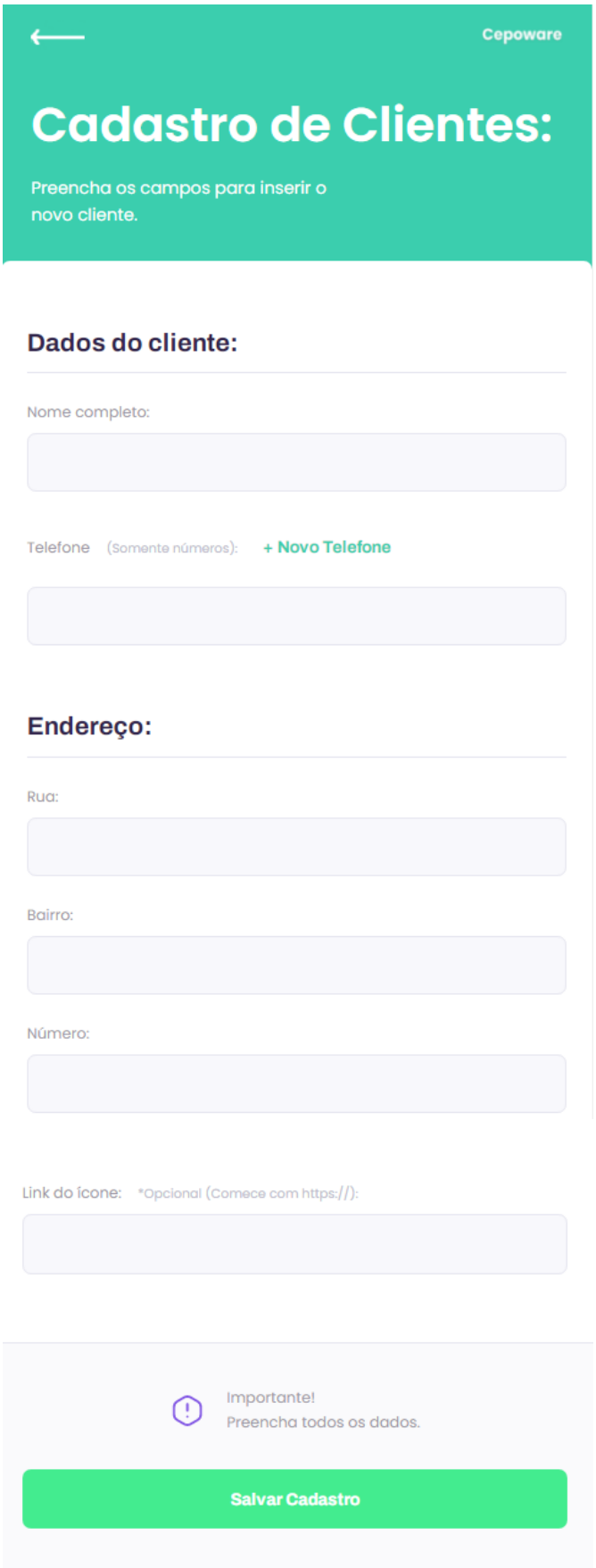
Através do botão “Novo Cliente”, o usuário pode acessar a página “Cadastro de Clientes”. Ao preencher os dados (Nome, Telefone, Rua, Bairro, Número e, opcionalmente, um avatar) pode-se registrar um novo cliente no banco de dados. Essas páginas são mostradas na versão *desktop* (Figura 19) e na versão *mobile* (Figura 20).

Figura 19 – Cadastro de Clientes



Fonte – Autora

Figura 20 – Cadastro de Clientes Mobile



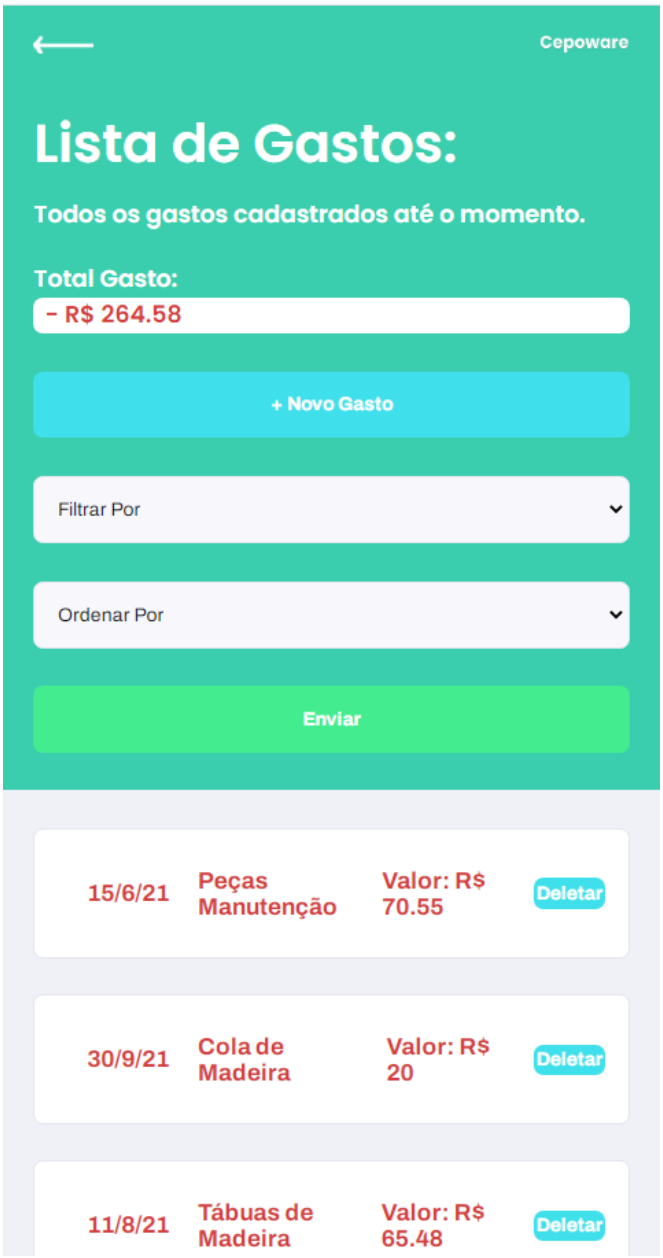
Fonte – Autora

Na Figura 21 (versão *desktop*) e Figura 22 (versão *mobile*) está ilustrada a página de listagem de gastos acessada pelo botão “Administrar Gastos” na página inicial. Nesta tela o usuário pode visualizar cada gasto registrado no sistema, bem como o total gasto pela empresa.

Figura 21 – Lista de Gastos

Fonte – Autora

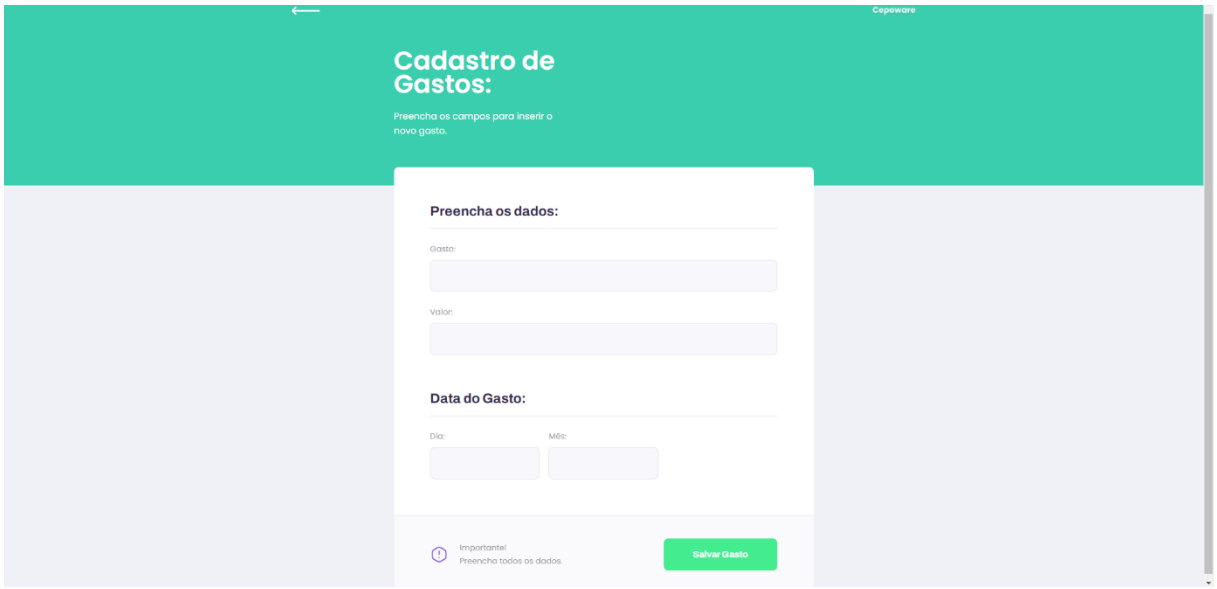
Figura 22 – Lista de Gastos Mobile



Fonte – Autora

Ao selecionar o botão “+ Novo Gasto” o usuário acessa a página de cadastro de gastos e, ao preencher os dados desta tela (descrição da despesa, valor, dia e mês) pode-se incluir esse registro no banco de dados. Assim, esse valor pode ser descontado dos ganhos para calcular o lucro da empresa. Nas Figura 23 e Figura 24 são ilustradas as páginas de cadastro de gastos em formato *desktop* e *mobile*, respectivamente.

Figura 23 – Cadastro de Gastos



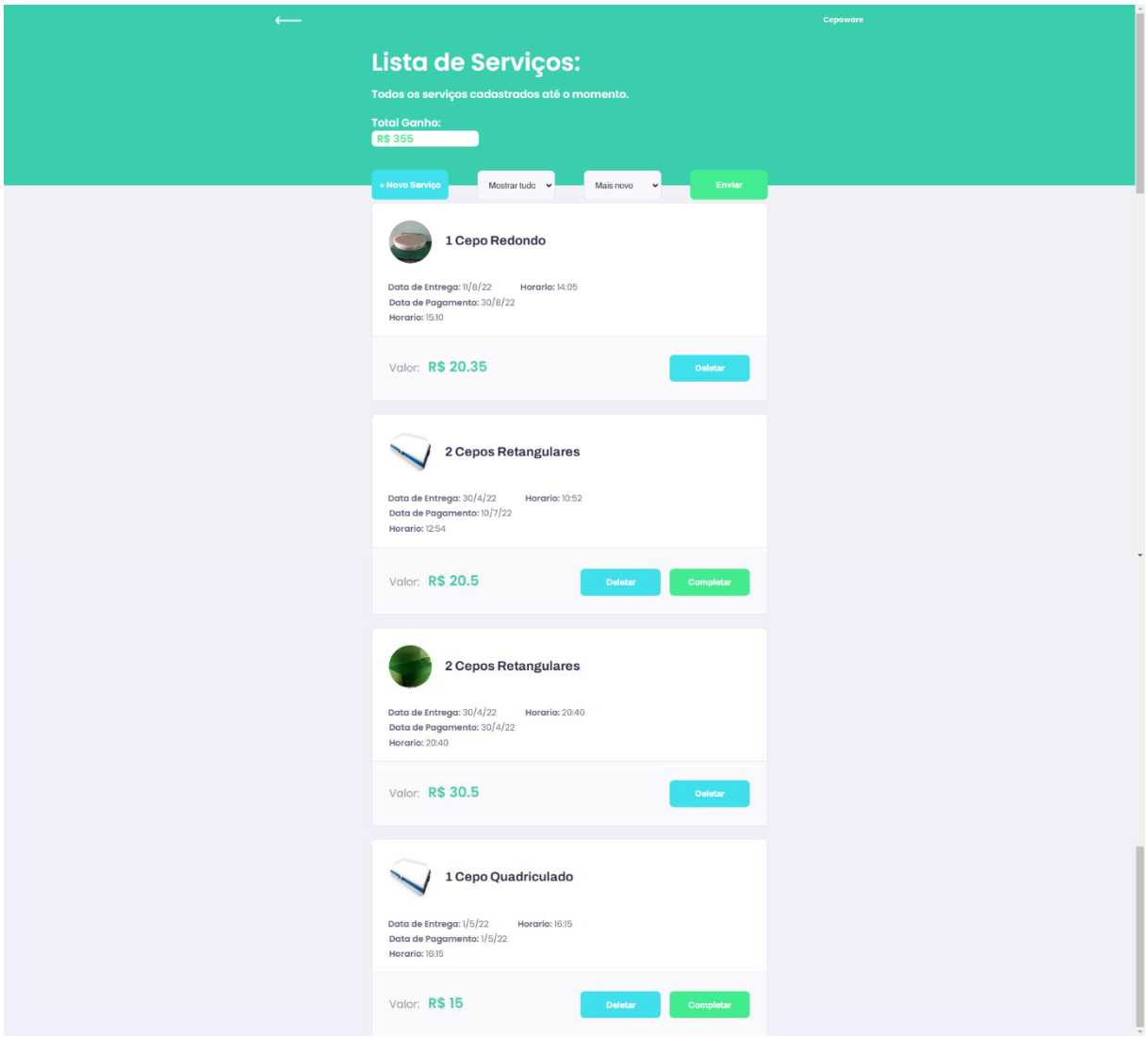
Fonte – Autora

Figura 24 – Cadastro de Gastos Mobile

Fonte – Autora

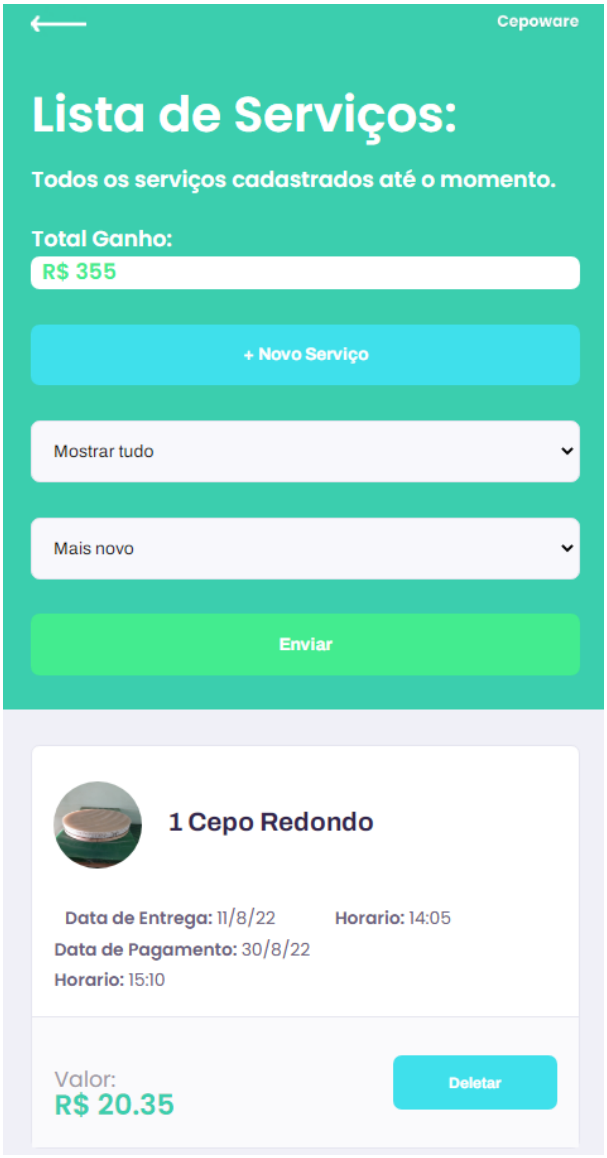
Na Figura 25 encontra-se a página de listagem de serviços em modo *desktop*, enquanto nas figuras 26 e 27 encontra-se a página de listagem de serviços em modo *mobile*. Nesta tela, o usuário pode ver os serviços cadastrados, sejam estes pendentes ou concluídos. Pode-se ainda ver o total recebido pela empresa, considerando-se serviços já pagos e serviços ainda não pagos.

Figura 25 – Listagem de Serviços



Fonte – Autora

Figura 26 – Listagem de Serviços Mobile Parte 1



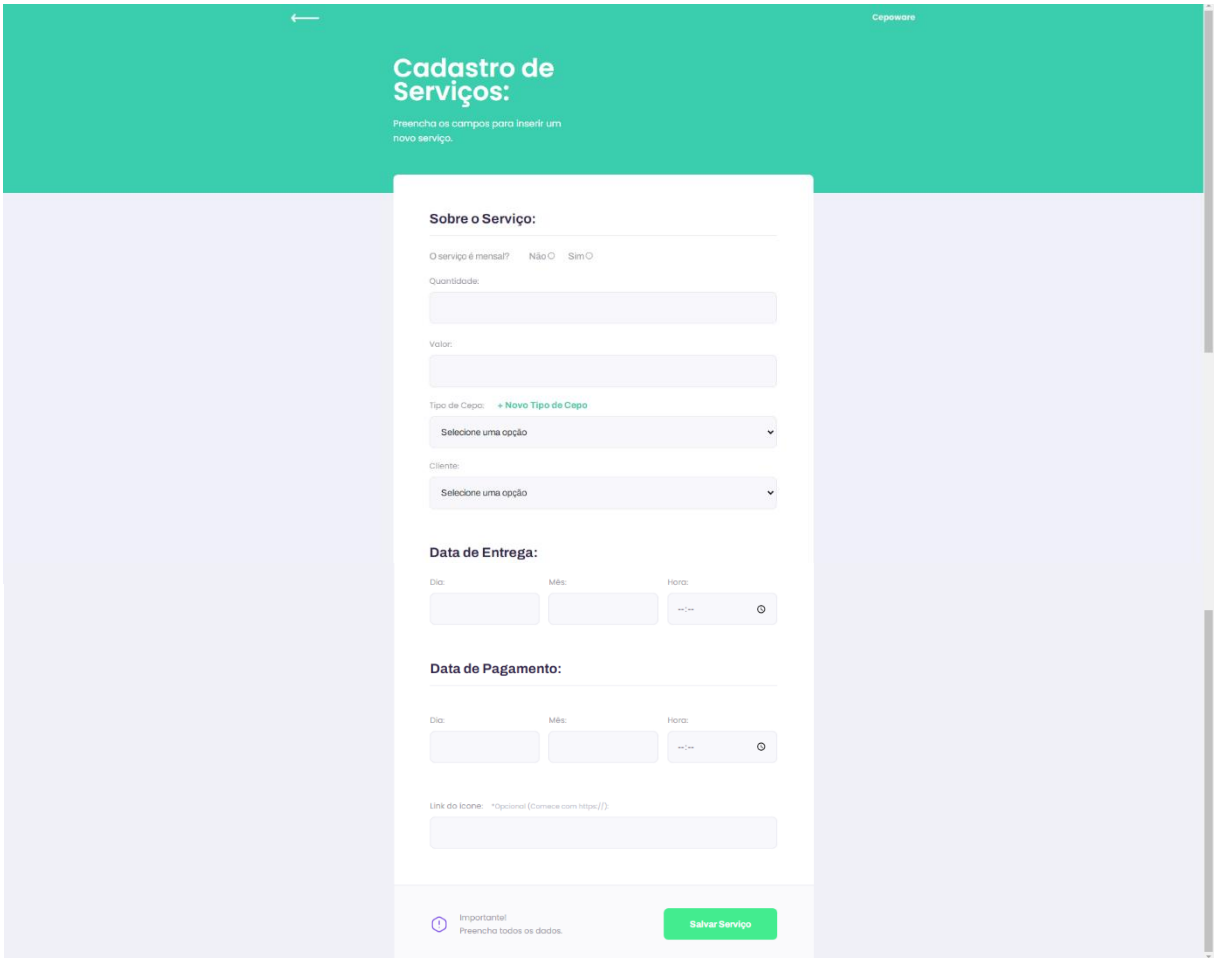
Fonte – Autora

Figura 27 – Listagem de Serviços Mobile Parte 2

Fonte – Autora

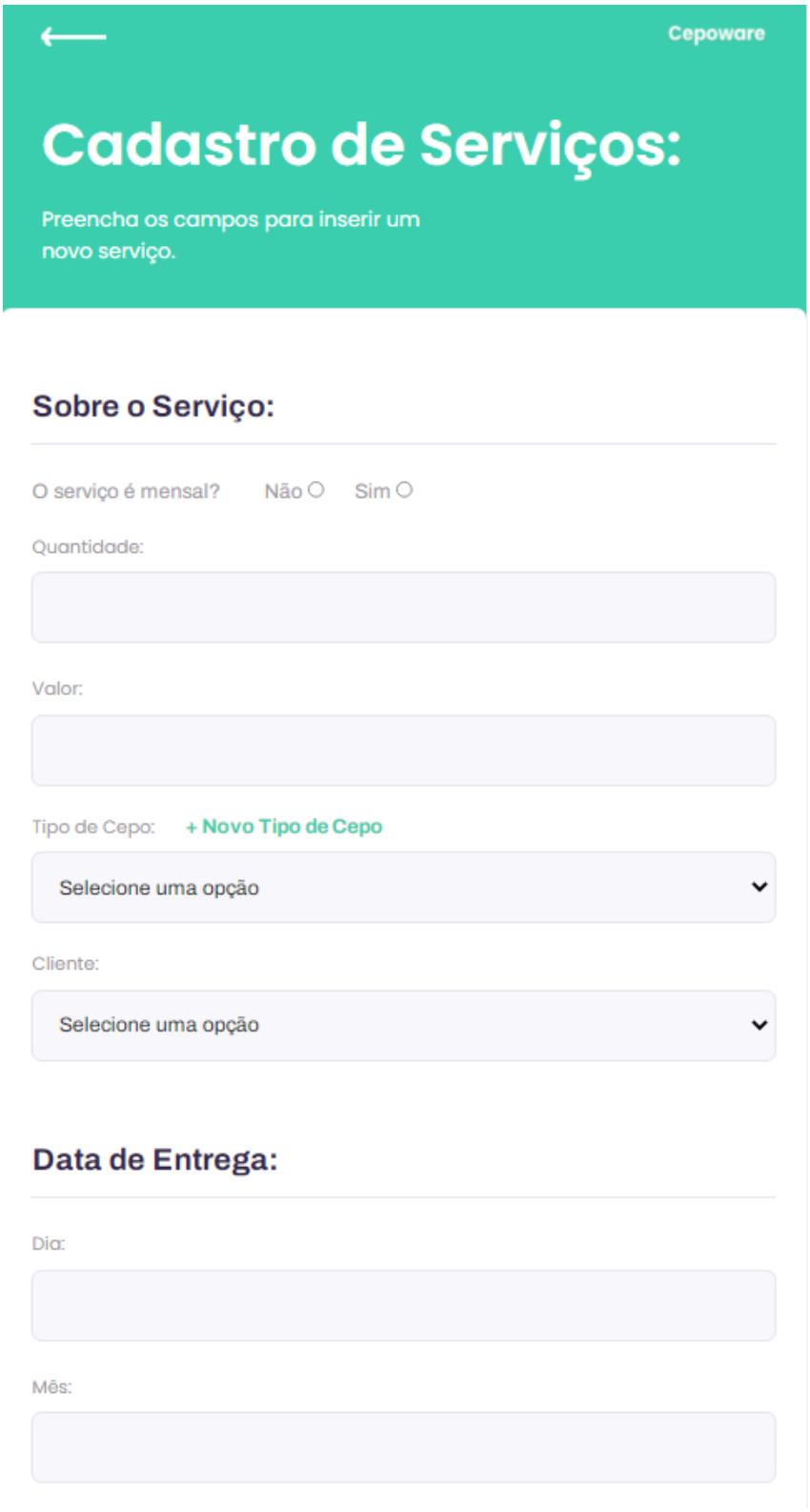
As telas exibidas na Figura 28, Figura 29 e Figura 30 são acessadas pelo botão “+ Novo Serviço”. Ao se preencher os dados sobre a quantidade de cepos, o valor total do serviço, o tipo de cepo, o cliente e as datas de entrega e pagamento, o usuário pode registrar um novo serviço. Assim, o trabalho realizado pode ser armazenado no sistema e ser contabilizado no lucro real e estimado.

Figura 28 – Cadastro de Serviços



Fonte – Autora

Figura 29 – Cadastro de Serviços Mobile Parte 1



Fonte – Autora

Figura 30 – Cadastro de Serviços Mobile Parte 2

Fonte – Autora

O software foi apresentado para a empresa após a confecção das telas. Os funcionários apreciaram o visual simples, a facilidade dos cadastros e a listagem informativa dos serviços realizados. O projeto se encontra em fase de testes na empresa e está sendo utilizado no dia a dia dos trabalhadores. Novas sugestões e melhorias estão sendo coletadas para o desenvolvimento de futuras versões e para a melhora contínua do software.

Considerações finais

Um ambiente de trabalho bem organizado, em que as datas combinadas com os clientes são respeitadas e atendidas é essencial para o bom desempenho de uma empresa.

Este projeto buscou apresentar o desenvolvimento de um software *web* responsivo, que possa ser utilizado em aparelhos *desktop* e *mobile* capaz de registrar clientes, pedidos, datas de entrega, datas de pagamento e gastos. Com a capacidade de verificar os serviços e gastos da empresa em um só lugar, o usuário reduz a possibilidade de esquecer datas importantes, perder clientes, ou perder tempo procurando registros de serviços já concluídos.

Ao concluir este projeto, é possível notar que o software Cepoware de fato permite o armazenamento de serviços, gastos, clientes e realiza os cálculos do lucro real e do lucro estimado, além de incluir alertas sobre serviços que devem ser entregues e pagamentos a receber.

Versões futuras do software poderão incluir novos tipos de alertas, como aqueles para alertar funcionários sobre clientes que não frequentam a empresa há muito tempo. Tais alertas permitirão que os empregados verifiquem a razão da ausência do cliente na empresa a fim de executar ações que minimizem a perda de clientes. Alertas já implementados também poderão ser melhorados para serem mais específicos e precisos.

Uma versão futura ainda poderá contar com controle de serviços mensais, nos quais a empresa cadastra um único serviço de manutenção para os cepos de um cliente com datas mensais especificas de pagamento e entrega. Funcionalidades para ordenação, busca e filtros de serviços farão com que o usuário encontre o que procura com maior facilidade.

Os cadastros também possuirão maior quantidade de detalhes em versões posteriores, incluindo o CNPJ das empresas clientes, seus proprietários e nome fantasia. Funcionalidades para que sejam gerados relatórios, históricos de serviços e gastos e notas fiscais também estão sendo planejadas.

O desenvolvimento deste projeto permitiu grandes ganhos de conhecimento sobre ferramentas e conceitos antes não utilizados pela autora, auxiliando em seu crescimento pessoal, profissional e acadêmico.

Referências

ADAOBI, ANIUCHI. **Sending Notifications with node-notifier (Mac, Windows, Linux)**, 30/12/2020. Disponível em: <https://stackabuse.com/sending-notifications-with-node-notifier-mac-windows-linux/>. Acesso em: 17.abr.2022.

BHOSALE, S. T.; PATIL, Miss Tejaswini; PATIL, Miss Pooja. Sqlite: Light database system. **International Journal of Computer Science Mobile Computing**, v. 44, n. 4, p. 882-885, 2015. Disponível em: <https://ijcsmc.com/docs/papers/April2015/V4I4201599a35.pdf>. Acesso em: 17.abr.2022

CAMPOS, ANDRE L. N.; OLIVEIRA, TOACY. Software Processes with BPMN: An Empirical Analysis. In: HEIDRICH, Jens et al. **Product-Focused Software Process Improvement.** PROFES 2013. Lecture Notes in Computer Science, vol 7983. 1ª ed. Berlin: Springer, 2013. 339p.

CANTELON, Mike et al. **Node. js in Action**. 1ª ed. Greenwich: Manning, 2014, 416p.

CASCADING STYLE SHEETS HOME PAGE, **What is CSS?,** 16/06/2021. Disponível em: <https://www.w3.org/Style/CSS/Overview.en.html>. Acesso em: 21.out.2021.

ECKFORD, Matthew. **The importance of documentation in business operations and compliance**, 19/05/2018. Disponível em: <https://www.tmf-group.com/en/news-insights/articles/2016/may/importance-of-documentation/>. Acesso em: 14.apr.2022.

MDN WEB DOCS, **HTML: Linguagem de Marcação de Hipertexto** , 27/09/2021 . Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML> . Acesso em: 24.out.2021.

MILETTO, Evandro Manara; BERTAGNOLLI, Silvia de Castro. **Desenvolvimento de Software II:** Introdução ao Desenvolvimento Web com HTML, CSS, JavaScript e PHP-Eixo: Informação e Comunicação. 1ª ed. Rio Grande do Sul: Bookman, 2014, 276p.

NPMJS, **Node-Notifier Documentation**, 28/01/2022. Disponível em: <https://www.npmjs.com/package/node-notifier> Acesso em: 17.apr.2022.

OLIVEIRA, CELSO HENRIQUE PODEROSO DE. **SQL Curso Prático**. 1ª ed. São Paulo: Novatec, 2002. 272p.

RIBEIRO, RENATO. **Entenda o que são as regras de negócio e como ajudam seu software!** 06/10/2020, Disponível em: < https://www.iugu.com/blog/regras-de-negocio> . Acesso em: 02.nov.2021.

SILVA, MAURÍCO SAMY. **JavaScript, Guia do Programador:** Guia completo das funcionalidades da linguagem Javascript. 1ª ed. São Paulo: Novatec, 2010. 600p.

SOMMERVILLE, I. **Engenharia de Software**. 9ª ed. São Paulo: Pearson Addison Wesley, 2003. 543p.

SQLITE, **What is SQLite?**, 2021/06/2018. Disponível em: <https://www.sqlite.org/index.html> Acesso em: 17.out.2021.

VALENTE, MARCO TULIO. **Engenharia de Software Moderna:** Princípios e Práticas para Desenvolvimento de Software com Produtividade, 2020, 408p.