RAFAEL NOGUEIRA RABELO

DESENVOLVIMENTO WEB EM UMA STARTUP DE ENERGIA SOLAR

LAVRAS – MG

2023

RAFAEL NOGUEIRA RABELO

DESENVOLVIMENTO WEB EM UMA STARTUP DE ENERGIA SOLAR

Relatório de estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Sistemas de Informação, para a obtenção do título de Bacharel.

Prof. Dr. Bruno de Abreu Silva

Orientador

LAVRAS – MG

2023

RAFAEL NOGUEIRA RABELO

DESENVOLVIMENTO WEB EM UMA STARTUP DE ENERGIA SOLAR

Relatório de estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Sistemas de Informação, para a obtenção do título de Bacharel.

APROVADA em 24 de Fevereiro de 2023.

Prof. Dr. Bruno de Abreu Silva	UFLA
Prof. Dr. Paulo Afonso Parreira Junior	UFLA
Prof. Dra. Renata Teles Moreira	UFLA

Prof. Dr. Bruno de Abreu Silva
Orientador

LAVRAS – MG
2023

Dedico à toda minha família, principalmente minha mãe Maria Celeste Nogueira de Melo e meu pai Edilberto Honorato Rabelo. Dedico também a todos os meus amigos.

AGRADECIMENTOS

A princípio, agradeço à minha mãe e ao meu pai por todo apoio e todo esforço para oferecer uma vida de qualidade para mim e meu irmão.
Agradeço também a todos os meus familiares, por todo o apoio e desejos de sucesso.
Agradeço a todos os meus amigos, presentes em bons e maus momentos, além de me instigarem sempre nos objetivos profissionais.
Agradeço ao Prof. Dr. Bruno de Abreu Silva por seus ensinamentos diante do curso e por aceitar me orientar e me ajudar no processo de produção deste documento.
E agradeço à Universidade Federal de Lavras, e a todos os seus funcionários, e principalmente todos os professores que tive a oportunidade de ter sido aluno.

RESUMO

O setor de energia solar no Brasil tem experimentado um crescimento significativo nos últimos anos, saindo de 14 GW de potência instalada para 22 GW entre 2021 e 2022. Crescimento esse que pode ser atribuído a diversos fatores, como a diminuição dos custos de painéis fotovoltaicos, aumento dos incentivos governamentais, e o crescente custo da energia convencional. Junto desse aumento da base instalada de usinas solares vem também uma grande demanda para monitoramento e gestão dessas usinas. O objetivo deste trabalho é relatar, do ponto de vista do desenvolvedor, como foi a participação do estagiário no desenvolvimento do sistema GDash na startup Sharenergy. Sistema que se trata de uma plataforma para monitoramento de sistemas fotovoltaicos, auditoria de faturas das concessionárias de energia elétrica e envio de relatórios automáticos a clientes. O estagiário entrou no projeto no início da formulação de uma equipe de desenvolvimento própria, visto que a criação e desenvolvimento inicial haviam sido feitos por uma empresa terceirizada, havendo uma base pronta, mas muita coisa não funcional ou a ser feita. Durante o desenvolvimento do projeto, usou-se o Framework Scrum. O projeto foi desenvolvido em React para o *frontend*, Node para o *backend*, ambos sob a plataforma Meteor.js e MongoDB para o banco de dados, e teve a atuação do estagiário nessas três áreas.

Palavras-chave: Energia Solar. Framework Scrum. React. Node. Meteor. MongoDB.

ABSTRACT

The solar energy sector in Brazil has experienced significant growth in recent years, going from 14 GW of installed power to 22 GW between 2021 and 2022. This growth can be attributed to several factors, such as decreasing costs of photovoltaic panels, increasing government incentives, and the rising cost of conventional energy. Along with this increase in the installed base of solar plants, there is also a great demand for monitoring and managing these plants. The objective of this work is to report, from the developer's point of view, how the intern's participation was in the development of the GDash system in the startup Sharenergy. System that is a platform for monitoring photovoltaic systems, auditing utility bills and sending automatic reports to customers. The intern joined the project at the beginning of the formulation of its own development team, since the creation and initial development had been carried out by a third-party company, with a ready base, but a lot of things that were not functional or to be done. During the development of the project, the Scrum Framework was used. The project was developed in React for the frontend, Node for the backend both under the Meteor.js platform and MongoDB for the database, and had the work of the intern in these three areas.

Keywords: Solar energy. Framework Scrum. React. Node. Meteor. MongoDB.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto

De acordo com Sol (2022), o Brasil é um país que tem um potencial enorme para produção de energia solar, por possuir um nível de incidência solar acima da média em relação a outros países, como por exemplo, seu local menos ensolarado tem uma incidência maior do que o local mais ensolarado da Alemanha. Além disso, com o passar dos anos se teve um aumento do período de seca no país, devido às mudanças climáticas vindas do aquecimento global, fazendo com que as usinas hidrelétricas, que compõem a principal forma de geração na matriz energética brasileira, tivessem uma queda na produção como relatado em G1 (2021), tendo a energia solar como uma grande opção para complementar a geração durante esse período.

Segundo ABSOLAR (2022), a potência instalada de usinas fotovoltaicas no Brasil já ultrapassa 22 GW, um crescimento de 57% em relação ao ano de 2021. Tendo 69% dessa potência instalada vindo da geração distribuída, que é composta por sistemas fotovoltaicos em residências, comércios, entre outros estabelecimentos, em um todo de 1.463.241 de sistemas.

Essas usinas de geração distribuída ficam em sua maioria na responsabilidade de empresas chamadas de integradoras, que fazem o projeto, regularização do sistema, instalação, manutenção e um acompanhamento pós-venda, fazendo um monitoramento do sistema e auditoria da compensação de créditos da concessionária de energia elétrica. Porém esse pós-venda acaba sendo um problema para muitos desses profissionais, onde cada marca de equipamento solar tem sua própria aplicação para monitoramento, além de que toda a auditoria tem que ser feita manualmente. Diante disso, observa-se que existe um potencial em centralizar e automatizar esse acompanhamento feito por essas empresas.

1.2 Soluções atuais para a monitoramento de usinas solares

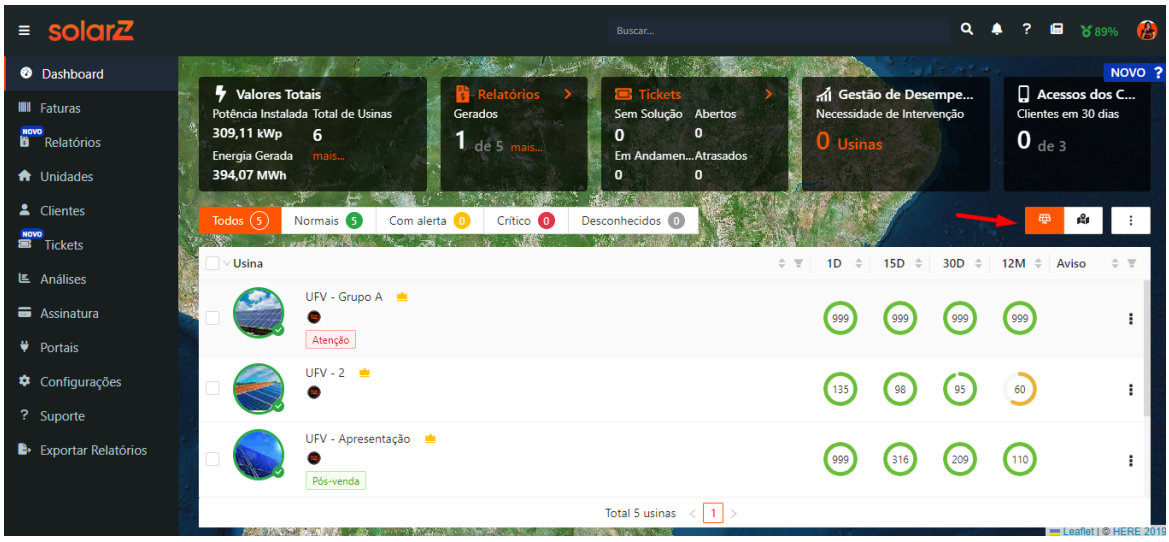
O monitoramento de usinas solares é feito originalmente através do *software* disponibilizado por cada fabricante de inversor solar, assim tendo diversos softwares diferentes para fazer esse acompanhamento.

Com gradativamente mais fabricantes produzindo esses equipamentos solares, se criou um grande problema, pois cada *software* atuava de uma maneira diferente. Além de tudo, cada vez mais eram necessárias análises mais profundas daqueles dados obtidos, como por exemplo a auditoria das faturas que os clientes recebem da concessionária de energia elétrica, comparando

o que consta na fatura com o que foi reportado pelo *software* como produzido pela usina, onde todo esse processo tinha que ser feito manualmente para cada usina. Foi então que surgiram as plataformas que unificam esses dados de diversos *softwares*.

Um espaço unificado para o monitoramento de diversas fabricantes é a plataforma SolarZ [1]. Trata-se de um sistema de monitoramento de múltiplas usinas solares, com suporte a diversos fabricantes de inversores solares, e download e leitura automática de faturas de algumas concessionárias elétricas, com seu principal diferencial sendo o foco no dono da usina, possibilitando que o integrador personalize um aplicativo para esse cliente, gere relatórios de consumo, etc. Ou seja, o integrador paga uma assinatura da plataforma, onde ele consegue ter acesso a dados em tempo real e alertas das usinas que ele presta suporte, e ao mesmo tempo consegue disponibilizar para o dono da usina um aplicativo personalizado para que ele também possa monitorar sua usina e acompanhar *tickets* de suporte e manutenção. Abaixo, a Figura 1.1 ilustra o *Dashboard* da SolarZ, onde é mostrado na parte superior um resumo total de todas usinas, com energia gerada, *tickets* abertos, relatórios, etc, na parte de baixo a lista de usinas daquele usuário, e na lateral esquerda o menu de navegação.

Figura 1.1 – *Dashboard* SolarZ



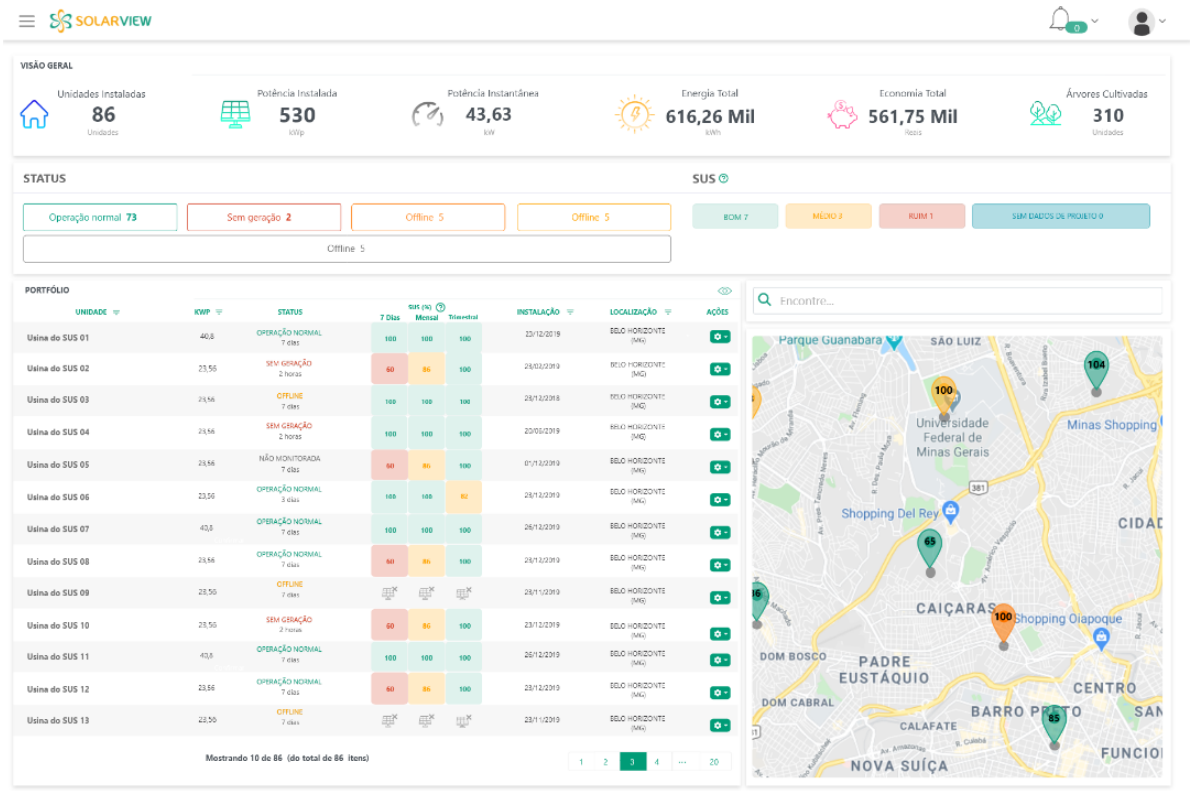
Fonte: SolarZ (2023)

Outro serviço de monitoramento unificado é o SolarView Pro[2]. Inicialmente a SolarView era somente uma fabricante de coletores de dados universal, que funciona junto aos inversores solares transmitindo os dados medidos pelo equipamento, posteriormente eles criaram o SolarView Pro, um serviço para monitoramento não só de seus coletores como também de

[1] https://solarz.com.br/
[2] https://solarview.com.br/svpro/

equipamentos de outras fabricantes, tendo praticamente as mesmas funcionalidades descritas do SolarZ. A Figura 1.2 ilustra o *Dashboard* do SolarView Pro, onde é mostrado na parte superior um resumo total de todas usinas, com energia total, economia total, número de usinas, etc, e na parte inferior, a direita um mapa com a localização das usinas, e na esquerda a lista de usinas daquele usuário.

Figura 1.2 – *Dashboard* SolarView Pro



Fonte: SolarView (2023)

No Brasil, até o momento do desenvolvimento do projeto, nenhum desses sistemas estavam totalmente consolidados, tendo a oportunidade para entrada de um novo sistema, principalmente oferecendo uma auditoria automática das faturas, onde as faturas são obtidas diretamente do sistema da concessionária de energia e comparadas com os dados registrados pelos inversores.

1.3 Empresa

O estágio foi feito na empresa Sharenergy[3], uma empresa pequena, por volta de 10 funcionários, sendo uma integradora especializada em projetos, instalação, manutenção e acompanhamento de sistemas fotovoltaicos.

A empresa trabalha na parte de tecnologia com Metodologia de Desenvolvimento Ágil com base no *Framework* Scrum, porém de forma adaptada para o contexto da empresa.

O único projeto da empresa na área de tecnologia é o GDash[4], uma plataforma por assinatura para monitoramento de usinas fotovoltaicas, reunindo informações de diversas marcas de equipamentos solares, e trazendo análises dessas usinas como, total economizado, auditoria das faturas, etc.

1.4 Objetivos

Um dos objetivos deste relatório é mostrar o desenvolvimento do GDash, que visa atuar neste cenário de usinas fotovoltaicas com principal objetivo de criar uma plataforma universal para gestão e monitoramento. O sistema funciona reunindo informações de diversas marcas, além de facilitar atividades de gestão como auditoria de faturas e gerenciamento de chamados, sendo o único projeto em que o estagiário atuou. O estágio iniciou-se em janeiro de 2021 e foi até outubro de 2021.

Do ponto de vista da sua atuação no GDash, os objetivos eram desenvolver o sistema para ser lançada uma versão preliminar até abril de 2021, para testes em empresas parceiras, e continuar melhorando e adicionando funcionalidades de acordo com o *feedback* recebido, para o lançamento oficial no segundo semestre do mesmo ano.

Do ponto de vista do crescimento do estagiário, os objetivos do estágio foram:

- Vivenciar suas primeiras experiências no meio profissional;
- Aprimorar os conhecimentos técnicos adquiridos durante a formação acadêmica, e adquiridos externamente por meio de cursos e estudos próprios;
- Contribuir na empresa, buscando o crescimento profissional, pessoal, além do interpessoal.

[3] <https://www.sharenergy.com.br/>
[4] <https://www.gdash.com.br/>

O restante deste relatório está organizado da seguinte maneira: no Capítulo 2, são apresentados os principais conceitos envolvidos na área em que este trabalho se encontra, além dos componentes técnicos, como *frameworks* e bibliotecas, as linguagens de programação e alguns atributos e características que as compõem e que contribuíram para o desenvolvimento. O Capítulo 3 traz a descrição do sistema pronto e apresenta imagens de algumas telas, explicações de funcionalidades e seus fluxos dentro do sistema. O Capítulo 4 encerra o relatório com considerações finais sobre o projeto, o estágio e a visão do estagiário sobre o papel da Universidade durante essa etapa de estágio.

2 CONCEITOS E TECNOLOGIAS UTILIZADAS

Neste Capítulo, são apresentados os conceitos, tecnologias, ferramentas e *frameworks* que foram usados para auxiliar no desenvolvimento do sistema.

2.1 Banco de Dados

Date (2003) define que "um banco de dados é uma coleção de dados persistentes utilizados pelos sistemas de aplicação de uma empresa". Ou seja, banco de dados é um conjunto de registros que descrevem entidades do mundo real, como pessoas, eventos ou itens, salvos e utilizados por um software. Existem diversos tipos de bancos de dados, os mais comuns são o relacionais e não relacionais.

Os bancos de dados relacionais são compostos por tabelas, com colunas que especificam os dados a serem armazenados, dados esses que são armazenados como linhas na tabela, como ilustrado no Quadro 2.1. Outra importante característica desse tipo de banco de dados é o uso de relacionamentos entre as tabelas, utilizando um identificador para referenciar um registro em outra tabela.

Os bancos de dados não relacionais são qualquer banco de dados que não seguem o modelo relacional. Mas se destacam principalmente dois tipos, os de chave e valor, que como sugere o nome são coleções de pares com um identificador e um dado a ser salvo, e os orientados a documentos, que diferentemente dos bancos relacionais não necessitam de uma estrutura uniforme, podendo ter registros com tipos de valores diferentes ou até campos a mais, como podemos ver na Figura 2.1.

Quadro 2.1 – Exemplo de banco de dados relacional

Pessoas			
ID	Nome	Data de Nascimento	CPF
1	Lucas Cuckerson	10/01/1976	111.111.111-11
2	Gemerson Alves	12/10/1998	222.222.222-22
3	Gabriel Lilhead	03/06/2002	333.333.333-33
...	...	...	...

Fonte: Autor

Figura 2.1 – Exemplo de banco de dados não relacional orientado a documentos

```
[
  {
    _id: 1,
    nome: "Lucas Cuckerson",
    data_nascimento: "10/01/1976",
    cpf: "999.999.999-99",
  },
  {
    _id: 2,
    nome: "Gemerson Gonçalves",
    data_nascimento: ISODate("1998-10-12T00:00:00Z"),
    cpf: 99999999999,
  },
  {
    _id: 3,
    nome: "Jonas Nascimento",
    data_nascimento: "03/06/2002",
    cpf: "999.999.999-99",
    sexo: "M",
  },
]
```

Fonte: Autor

2.2 MongoDB

MongoDB[1] é um banco de dados não relacional orientado a documentos de código aberto, criado pela 10gen(atual MongoDB Inc) em 2007, sendo usado principalmente por conta de sua flexibilidade e eficiência com grandes volumes de dados como descrito por Docs (2023a).

Bancos de dados orientados a documentos são feitos para armazenar dados semiestruturados, como por exemplo, XML (*Extensible Markup Language*) ou JSON (*JavaScript Object Notation*), ao contrário dos bancos de dados relacionais que utilizam tabelas. De acordo com Docs (2023b), no MongoDB são utilizados documentos chamados de BJSON (*Binary JSON*), semelhantes ao JSON porém com mais tipos de dados e representação em binário, dando assim mais desempenho e flexibilidade. A Figura 2.2 exibe um exemplo de um documento de um usuário salvo no MongoDB, descritos por "nome do campo: dado salvo", os dados sendo uma chave identificadora, nome, dividido em primeiro nome e último nome, data de nascimento, data de falecimento, lista de contribuições e número de visualizações.

Diferentemente de bancos de dados relacionais, o MongoDB não utiliza o SQL (*Structured Query Language*) para suas consultas, ele utiliza uma linguagem própria chamada de MQL (*MongoDB Query Language*) que possibilita o uso de expressões regulares, intervalos e até o

[1] <https://www.mongodb.com/>

Figura 2.2 – Exemplo de documento do MongoDB

```
var mydoc = {
               _id: ObjectId("5099803df3f4948bd2f98391"),
               name: { first: "Alan", last: "Turing" },
               birth: new Date('Jun 23, 1912'),
               death: new Date('Jun 07, 1954'),
               contribs: [ "Turing machine", "Turing test", "Turingery" ],
               views : NumberLong(1250000)
             }
```

Fonte: Docs (2023b)

uso de javascript em funções de agregação. A Figura 2.3 exibe um exemplo de uso da linguagem MQL, onde são buscados os pedidos feitos pelo cliente com nome "Jack Beanstalk", e na Figura 2.4 a mesma operação em SQL, podemos ver que a busca em MQL é bem mais simples e fácil de se entender.

Figura 2.3 – Exemplo de uso da linguagem MQL

```
0  // Query on embedded documents
1
2  const cursor = db.collection('orders').find({
3    'customer.name': 'Jack Beanstalk'
4  });
5
```

Fonte: MongoDB (2023)

Figura 2.4 – Exemplo de uso da linguagem SQL

```
1 SELECT o.* FROM customers c JOIN orders o ON c.id = o.customerId WHERE name = "Jack Beanstalk";
```

Fonte: Autor

2.3 Javascript

Como definido por MDN (2023b), Javascript é uma linguagem de programação leve, interpretada e baseada em objetos, que compõem as três principais tecnologias WEB junto com HTML e CSS. Foi criada em 1995 como parte do navegador Netscape sendo posteriormente integrada a praticamente todos os navegadores. Inicialmente sua principal função era tornar as páginas mais dinâmicas, que na época eram completamente estáticas com todas ações tendo que recorrer ao servidor, atualmente o javascript é uma linguagem multiplataforma e já está presente em diversas áreas e tecnologias como *backend*, *desktop* e *mobile*.

Outro fator importante da linguagem é sua estrutura multiparadigma, que apresenta características de linguagens baseadas em protótipos, funcionais e orientadas a objetos, assim tendo uma grande flexibilidade na forma de ser utilizada.

Em 1996 a Netscape submeteu a linguagem a ECMA (*European Computer Manufacturers Association*) a fim de criar uma padronização com colaboração de grandes empresas como a Microsoft[2], e assim surgiu o padrão ECMAScript. Segundo MDN (2023b), em 2012 todos navegadores modernos já tinham suporte ao ECMAScript 5.1 e os mais antigos a pelo menos o ECMAScript 3, e atualmente são lançadas novas especificações do ECMAScript anualmente.

A Figura 2.5 exibe um exemplo de classe em javascript que descreve um retângulo, recebendo sua altura e largura na inicialização e tendo um método para retornar a área calculada da figura geométrica.

Figura 2.5 – Exemplo de classe em Javascript

```
class Retangulo {
    constructor(altura, largura) {
      this.altura = altura; this.largura = largura;
    }
  //Getter
    get area() {
        return this.calculaArea()
    }

    calculaArea() {
        return this.altura * this.largura;
    }
}

const quadrado = new Retangulo(10, 10);

console.log(quadrado.area);
```

Fonte: MDN (2023a)

2.4 Node.js

Node.js é um ambiente de execução javascript de código aberto e multiplataforma, sendo utilizado principalmente na criação de aplicações web. Foi criado por Ryan Dahl em 2009

[2] <https://www.microsoft.com/>

utilizando o interpretador javascript V8, desenvolvido pela Google[3] para o navegador Google Chrome, com uma arquitetura assíncrona e orientada a eventos.

Uma das principais características da arquitetura orientada a eventos do Node é sua orquestração utilizando o *Event Loop* e a *Worker Pool*, que existem devido ao seu comportamento *single-thread* que tem o intuito de diminuir os recursos computacionais utilizados. Segundo Node.js (2023b), o *Event Loop* executa os *callbacks* javascript registrados pelos eventos e as requisições assíncronas não-bloqueantes feitas por esses *callbacks*. Já a Worker Pool tem o objetivo de lidar com as tarefas trabalhosas, como tarefas muito intensivas na CPU ou no I/O.

A Figura 2.6 exibe um trecho de código em Node.js, onde é criado um servidor na porta 3000 que retorna o texto "Hello World"em qualquer requisição feita.

Figura 2.6 – Trecho de código Node.js

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Fonte: Node.js (2023a)

2.5 React

React é uma biblioteca javascript para criação de interfaces interativas, criada pela empresa Meta (antigo Facebook) em 2011 para utilização interna, e se tornando código aberto em 2013. Segundo React (2023a), os três principais diferenciais do react são:

- Criação de interfaces declarativas que fazem com que o código seja mais previsível e simples de depurar;
- Estrutura de código baseada em componentes que facilitam o encapsulamento e separação entre lógica e interface;

[3] <https://www.google.com/>

- Facilidade de utilizar os conhecimentos aprendidos em outras áreas como em *mobile* utilizando React Native.

A Figura 2.6 exibe um exemplo de componente em React, que renderiza uma lista de compras com o título "*Shopping List For*"com o nome recebido como propriedade concatenado ao final.

Figura 2.7 – Trecho de código React

```
class ShoppingList extends React.Component {
  render() {
    return (
      <div className="shopping-list">
        <h1>Shopping List for {this.props.name}</h1>
        <ul>
          <li>Instagram</li>
          <li>WhatsApp</li>
          <li>Oculus</li>
        </ul>
      </div>
    );
  }
}

// Example usage: <ShoppingList name="Mark" />
```

Fonte: React (2023b)

2.6 Meteor.js

Meteor.js é um *framework* web *fullstack* de código aberto mantido pela empresa Meteor Software. Ele foi escrito em Node.js e e encapsula as três principais camadas do desenvolvimento web:

- Banco de dados, integrando-se com o MongoDB;
- *Backend*, utilizando Node.js e o protocolo *Distributed Data Protocol (DDP)* para comunicação;
- *Frontend*, que pode ser utilizado com as bibliotecas javascript React, Vue, Blaze e Svelte.

Segundo Meteor (2021), DDP é um protocolo cliente-servidor que utiliza o padrão de comunicação *publish-subscribe*, em que o servidor atua como *publisher* criando e enviando eventos, que são recebidos por um ou mais clientes que fizeram *subscribe*. Ele atua com SockJS

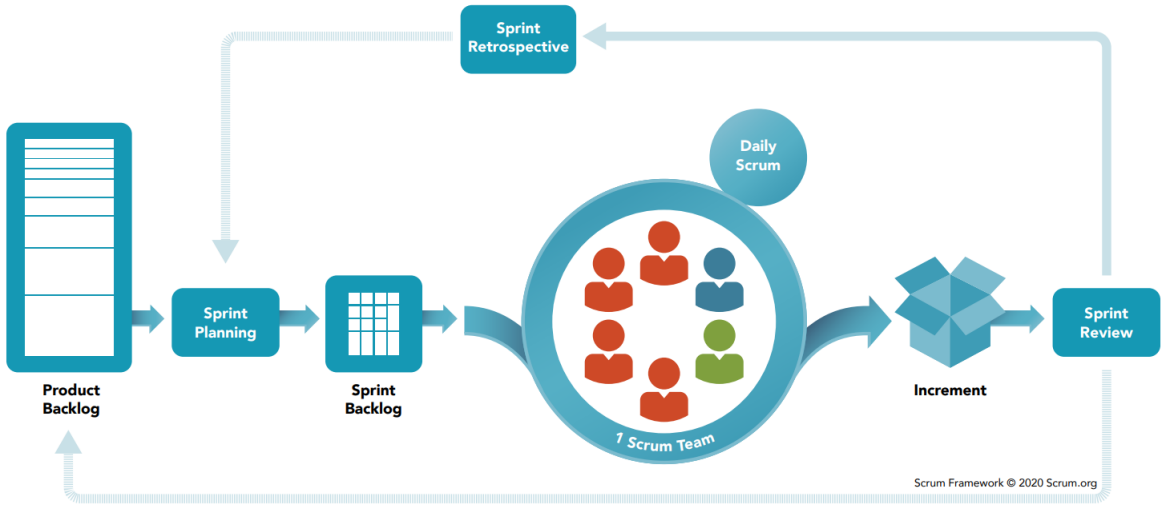
ou WebSockets como meio de comunicação no baixo nível, transmitindo os dados em formato *JavaScript Object Notation (JSON)* e *Extended JavaScript Object Notation (EJSON)*. A utilização do protocolo DDP tem como objetivo facilitar a comunicação entre cliente e servidor, principalmente em situações onde se tem a necessidade da atualização dos dados no cliente a cada mudança ocorrida no banco de dados.

2.7 Framework Scrum

Cruz (2018) define que "Scrum é um *framework* para desenvolver e manter produtos complexos que também pode ser utilizado no gerenciamento ágil de projetos que se destinam também à criação de produtos". Ou seja, é um *framework* com grande flexibilidade, podendo ser usado no desenvolvimento de software, hardware, marketing, etc.

A Figura 2.8 ilustra toda a estrutura da Scrum que será descrita posteriormente, mostrando o fluxo de seus eventos e artefatos.

Figura 2.8 – Estutura do Scrum



Fonte: Scrum.org (2023)

Schwaber e Sutherland (2013) simplificam essa definição dizendo que, a ideia do Scrum é que um pequeno grupo de pessoas consiga tratar problemas complexos de maneira adaptativa, com foco na entrega de valor e na previsibilidade do prazo de desenvolvimento. Esse grupo de pessoas são divididas em 3 responsabilidades:

- *Product Owner*: é a pessoa responsável por maximizar o valor do produto, e principalmente por gerenciar o *Backlog* do Produto, que será explicado a seguir;

- Time de Desenvolvimento: é um grupo de profissionais que realizam o trabalho que compõe o incremento ao final da Sprint, um ciclo de trabalho de período determinado que produz algum incremento e valor ao produto em sua conclusão;

- *Scrum Master*: é a pessoa responsável por garantir a aplicação do Scrum, desde a teoria, as práticas e as regras, servindo os outros grupos do Scrum.

Outro elemento da estrutura bem característico do Scrum são seus artefatos, que representam o trabalho ou o valor para o fornecimento de transparência, sendo 3 artefatos:

- Incremento: é soma da coleção de itens do *Backlog* que devem ser completados durante a Sprint, com os incrementos das Sprints anteriores;

- *Backlog* do Produto: é uma lista dinâmica de necessidades do produto como mudanças a serem feitas, novas funcionalidades e correções, que vão sendo alteradas com o andar do projeto;

- *Backlog* da Sprint: é uma lista de itens do *Backlog* do Produto que são escolhidos para a Sprint, ou seja, são as modificações que formam um novo incremento ao final da Sprint.

Outra característica do Scrum são seus eventos e rotinas definidas, que fazem parte da Sprint. A Sprint é o principal componente do *framework*, tendo a duração definida no início, podendo durar até um mês, e sempre se iniciando uma nova após o encerramento da outra. Elas também apresentam componentes fixos que não podem ser alterados após o início, como metas de qualidade e objetivo. Essas Sprints são compostas pelos eventos:

- *Planning*: é a reunião inicial da Sprint, com participação colaborativa de toda equipe, definindo as metas, duração e objetivos de cada integrante;

- *Daily*: são reuniões diárias com tempo máximo de 15 minutos, onde o trabalho feito até o momento é inspecionado e é previsto o trabalho que irá ser feito até a próxima reunião, além de ser também relatado qualquer problema ou empecilho encontrado;

- *Review*: é a reunião feita no final da Sprint para inspecionar todo o incremento feito naquele período, e já pensando no que pode ser desenvolvido na próxima Sprint;

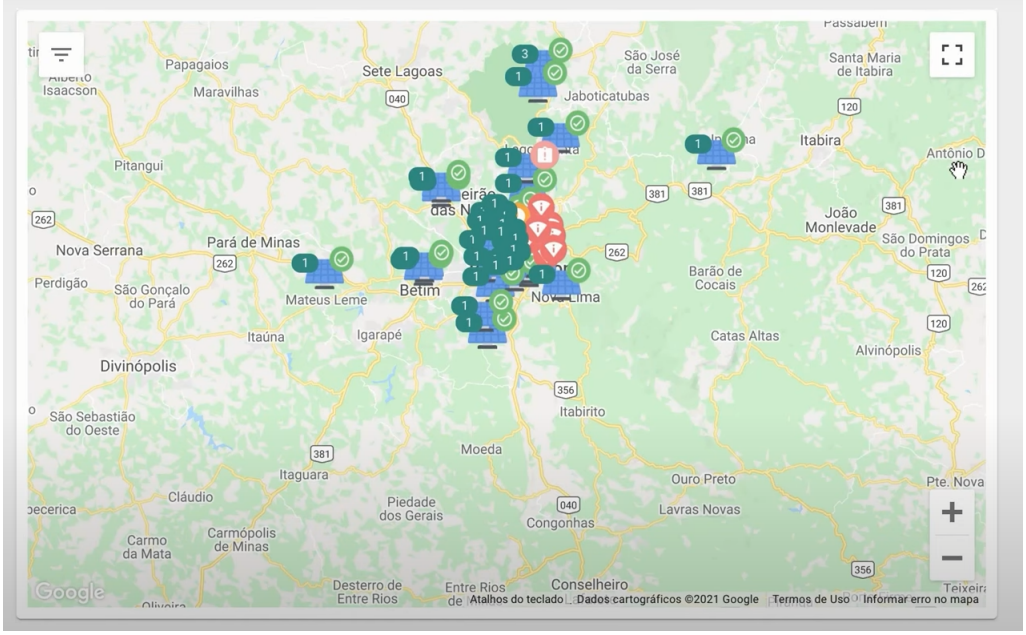
- *Retrospectiva*: é uma reunião que ocorre após a Review e antes do início da próxima Sprint, onde é feita uma avaliação dos indivíduos e processos da Sprint passada, buscando pontos que foram bem e outros que podem melhorar.

2.8 Google Maps

Google Maps é uma plataforma de mapas online multiplataforma, que oferece diversas funcionalidades como rotas de tráfego, imagens de satélite e visão 360º das ruas. Foi criada inicialmente pela empresa Where 2 Technologies, sendo adquirida em 2004 pela Google[4]. Em dezembro de 2022 a versão *mobile* foi o quinto aplicativo mais usado em aparelhos móveis nos EUA, sendo usado por 58% dos usuários, segundo Comscore (2022).

A Figura 2.9 é um exemplo de mapa onde são marcadas as usinas de acordo com as coordenadas daquele lugar.

Figura 2.9 – Mapa de usinas utilizado no GDash



Fonte: GDash (2023)

2.8.1 Geolocation

Geolocation é uma das ferramentas que fazem parte da plataforma Google Maps, fazendo a conversão de endereços em coordenadas geográficas ou o inverso, sendo muito utilizada para posicionar marcadores no mapa através de endereços transformando uma entrada como "1600 Amphitheatre Parkway, Mountain View, CA"em latitude 37.423021 e longitude -122.083739.

[4] <https://www.google.com/>

3 ATIVIDADES DESENVOLVIDAS

Neste Capítulo, será explorado o desenvolvimento do GDash durante o período do estágio. O desenvolvimento contribuiu para os objetivos iniciais propostos, principalmente tratando-se de aplicar aquilo que já foi visto durante o período de graduação, e adquirir conhecimentos diante das necessidades que foram surgindo.

Inicialmente, foi dado um tempo para que o estagiário se familiarizasse com o sistema, seguindo algumas instruções escritas pelos antigos responsáveis pelo projeto de como criar um módulo novo, novas páginas etc. Além disso, por a empresa ter uma parceria com a Amazon Web Services (AWS), o estagiário teve a oportunidade de fazer um treinamento online de duração de um dia chamado AWS Solutions Training for Partners: Foundations - Technical[1].

Após esse período de treinamento, o estagiário foi alocado para a equipe de tecnologia da empresa responsável pelo GDash. Essa equipe era formada por 1 desenvolvedor, posteriormente mais um desenvolvedor entrou no projeto, 2 estagiários, e um dos sócios da empresa que não atuava de maneira técnica mas de certa forma como Scrum Master e Product Owner.

O desenvolvimento do projeto foi gerenciado e entregue somente conforme alguns eventos do Scrum. Foram feitas as *plannings* a cada início de uma nova *sprint*, que durava de uma a duas semanas; as *dailys* ocorriam inicialmente na parte da manhã e depois foi alterado para parte da tarde, e as *reviews* após o período de execução da *sprint*, feitas rapidamente junto das *plannings*.

3.1 GDash

O sistema GDash surgiu a partir da ideia de um dos fundadores da empresa, que viu os problemas que eles enfrentavam no pós-venda na própria empresa, os principais sendo, manter credenciais de diversos portais de monitoramento de cada marca de inversor, ter de baixar e auditar manualmente os dados das faturas emitidas pelas concessionárias de energia elétrica comparando com os dados medidos nos portais e por fim ter que gerar manualmente os relatórios de produção e economia que são enviados aos clientes mensalmente.

A partir disso foi contratada uma empresa terceirizada chamada Synergia[2], situada na Universidade Federal de Minas Gerais (UFMG), para o desenvolvimento de um sistema que sanasse esses problemas. Paralelamente também foram contratados profissionais do Centro

[1] <https://aws.amazon.com/pt/partners/training/>
[2] <https://www.synergia.dcc.ufmg.br/>

Federal de Educação Tecnológica (CEFET) para a criação de um dispositivo de coleta de dados em arduino chamado de Datalogger, para utilização via redes móveis em locais afastados, visto que a maioria dos inversores se conectava somente por WIFI, posteriormente esse projeto do Datalogger foi deixado de lado por conta do baixo retorno comparado ao custo e trabalho.

O sistema entregue pela Synergia foi entregue, produzido sobre um *boilerplate*[3] da empresa em MeteorJS, com autenticação com diferentes *roles*[4], parte das páginas prontas, e uma versão preliminar do algoritmo que faz o download, a leitura e auditoria das faturas. Além do MeteorJS o sistema utiliza Node.JS no *backend*, React no *frontend* e MongoDB como banco de dados. Esse sistema é dividido em duas partes, uma API chamada tambem de Datalogger, onde é feita a busca dos dados em tempo real dos inversores e o GDash em si com o resto das funcionalidades, construído com 2 tipos de usuário em mente, o integrador, que é o pagante da assinatura e o principal foco do sistema, e o dono da usina normalmente referido somente como cliente.

Essa obtenção de dados em tempo real dos inversores é a base principal do sistema, sendo através desses dados que são gerados os relatórios, os gráficos de produção das usinas, o valor economizado e a base para auditoria das faturas. Isso é feito com um algoritmo que roda em um intervalo de 15 minutos (é o tempo padrão mas algumas marcas de inversores apresentam um intervalo diferente), seguindo o fluxo:

1. A API Datalogger busca no banco de dados todas usinas cadastradas juntamente com seus inversores;

2. São buscadas no banco de dados as credenciais do portal relacionado àquele inversor, cadastradas pelo usuário;

3. É acessada a API específica de cada marca de inversor utilizando as credenciais;

4. São obtidos os dados do inversor daquele determinado intervalo de tempo, como temperatura, tensão, corrente, frequência e potência. Algumas marcas oferecem dados de minuto a minuto, outras somente de 5 em 5 minutos, e por aí vai;

5. Esses dados são formatados e salvos para se enquadrar no padrão utilizado pelo sistema.

[3] Base de código genérica utilizada como forma de acelerar o desenvolvimento de diversos projetos construídos pela empresa, tendo funções como login, cadastro e geração de formulários.

[4] São cargos determinados que são atribuídos aos usuários e utilizados na autenticação para definir restrições de acesso a uma função ou página.

O diferencial do GDash com os outros sistemas citados na Seção 1.2, é que ele reúne praticamente todas atividades necessárias para um integrador no pós-venda. Gerando e enviando relatórios mensais para os clientes via *email*, tendo um espaço para gerenciamento de chamados de suporte e manutenção das usinas, notificando alertas de funcionamento dos inversores em caso de algum erro ou problema e fazendo auditoria das faturas, que no caso de faturas da CEMIG[5] são baixadas e lidas automaticamente todo mês.

Além dessas funções focadas no usuário pagante, ele também fornece uma versão do sistema para acesso do cliente, onde o integrador insere as informações e o endereço de *email* de um cliente e vincula esse usuário a uma ou mais usinas, a partir disso é enviado um email para esse endereço, onde ele consegue escolher sua senha. Esse usuário então consegue ter acesso às informações em tempo real de suas usinas como produção, economia e potência e suas faturas.

Em relação a precificação, a plataforma tem um teste gratuito de 14 dias com acesso a todas funcionalidades de maneira ilimitada, e após isso ela atua na modalidade de assinatura com o preço variando de acordo com o número de usinas como visto na Figura 3.1

3.2 Fluxo de Cadastro

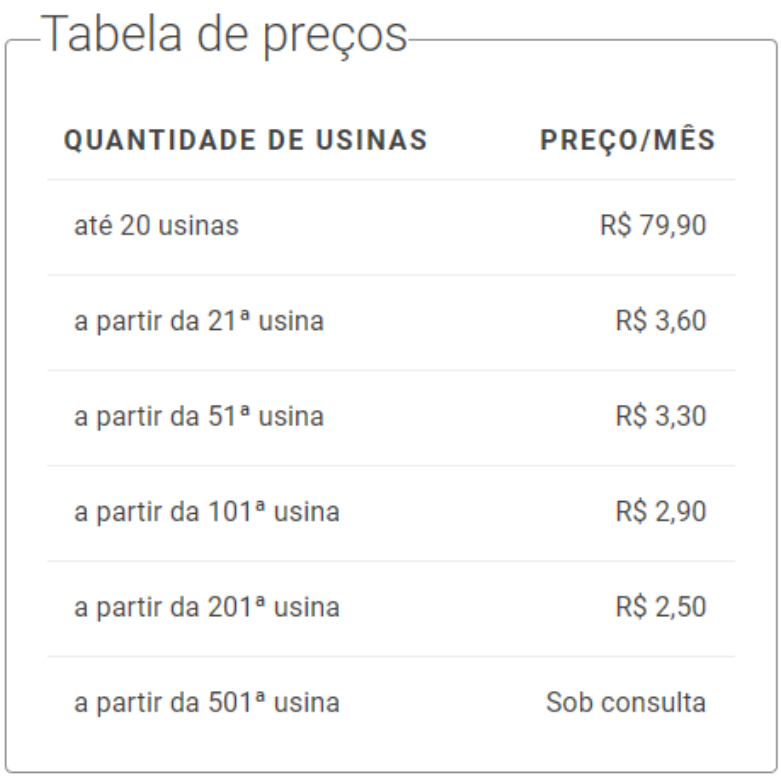
Uma das primeiras tarefas do estagiário foi a reformulação do fluxo de cadastro da plataforma, que anteriormente só tinha um modelo de usuário com os campos de email e senha, sem nenhuma página de cadastro. Foi definido então dois tipos de usuários diferentes, e para isso foram criados três fluxos de cadastro, um para integradores e dois para clientes, sendo definidos da seguinte maneira:

- Usuário Integrador: Preenchimento do formulário na *homepage* -> Contato da equipe de atendimento e envio do link de cadastro -> Preenchimento dos dados e finalização do cadastro.

- Usuário Cliente via formulário: Integrador abre a aba Clientes de uma usina -> Preenche o email do cliente -> Caso o cliente já exista, as informações são resgatadas, caso contrário é aberto um formulário para o preenchimento manual -> Opcionalmente são preenchidos os dados das instalações no nome desse cliente.

[5] <https://www.cemig.com.br/>

Figura 3.1 – Precificação da Plataforma GDash



Fonte: GDash (2023)

- Usuário Cliente via importação: Integrador abre a página de clientes -> Arrasta o arquivo do tipo csv para o campo demarcado.

3.3 Telas de cadastro

As telas de cadastro tem diferenças entre os tipos de usuários, porém os dois necessitam das mesmas informações como base, somente tendo alguns campos a mais entre eles, porém essas informações dependem do tipo de conta, pessoa física ou jurídica:

Física: Nome completo, email, CPF, telefone e endereço.

Jurídica: Razão social, email, CNPJ; inscrição estadual, telefone e endereço.

3.3.1 Cadastro do integrador

O cadastro de integrador atualmente acontece de forma mais restrita, seguindo um controle feito pela equipe de atendimento. Isso foi escolhido para ter uma postura mais intimista com o consumidor, buscando entender suas necessidades e ouvindo qualquer sugestão ou problema encontrado com esses primeiros usuários.

Primeiramente temos um formulário, como visto na Figura 3.2, que deve ser preenchido na *homepage*, página essa que funciona de maneira independente da plataforma. A partir desses dados, a equipe entra em contato com o interessado e envia um link para aí sim fazer o cadastro na plataforma.

Figura 3.2 – Formulario da *homepage*

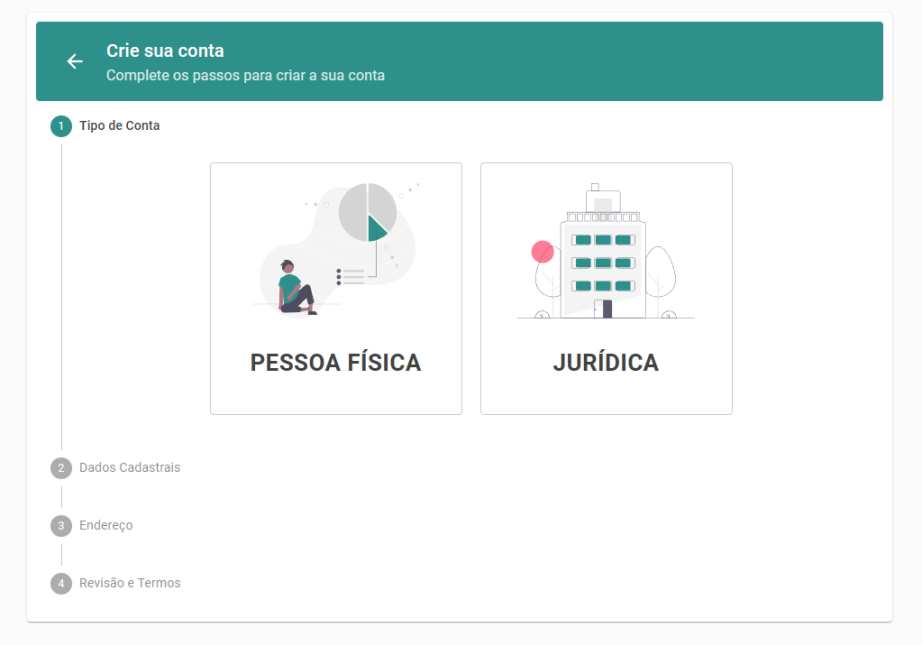
Fonte: GDash (2023)

Ao abrir esse link o consumidor tem acesso a um cadastro dividido em 4 etapas. Na primeira etapa ele deve selecionar o tipo de conta, tendo as opções de pessoa física ou jurídica, como observado na Figura 3.3.

Após selecionar o tipo de conta, o formulário avança para a etapa 2, como visto na Figura 3.4, onde estão os campos das informações base ditas anteriormente, esses campos são pré-preenchidos com os dados que foram inseridos no formulário da *homepage*. Além disso, nessa etapa é feita a escolha da senha da conta, que possui algumas restrições como número de caracteres, por exemplo. E também é feita uma verificação se o email inserido já está cadastrado no sistema.

O usuário então avança para etapa 3 do cadastro, onde estão os campos de endereço, como visto na Figura 3.5. O primeiro campo de CEP serve como forma de facilitar o pre-enchimento, visto que com ele são buscados e pré-preenchidos os outros dados de endereço

Figura 3.3 – Etapa 1 de cadastro do integrador



Fonte: GDash (2023)

com a API do ViaCEP[6], porém também é possível preencher manualmente caso tenha algum problema.

Por fim, ao chegar na etapa 4, é mostrada uma revisão de tudo que foi preenchido e devem ser aceitos os termos e compromisso para finalização do cadastro, como mostrado na Figura 3.6.

3.3.2 Cadastro de clientes

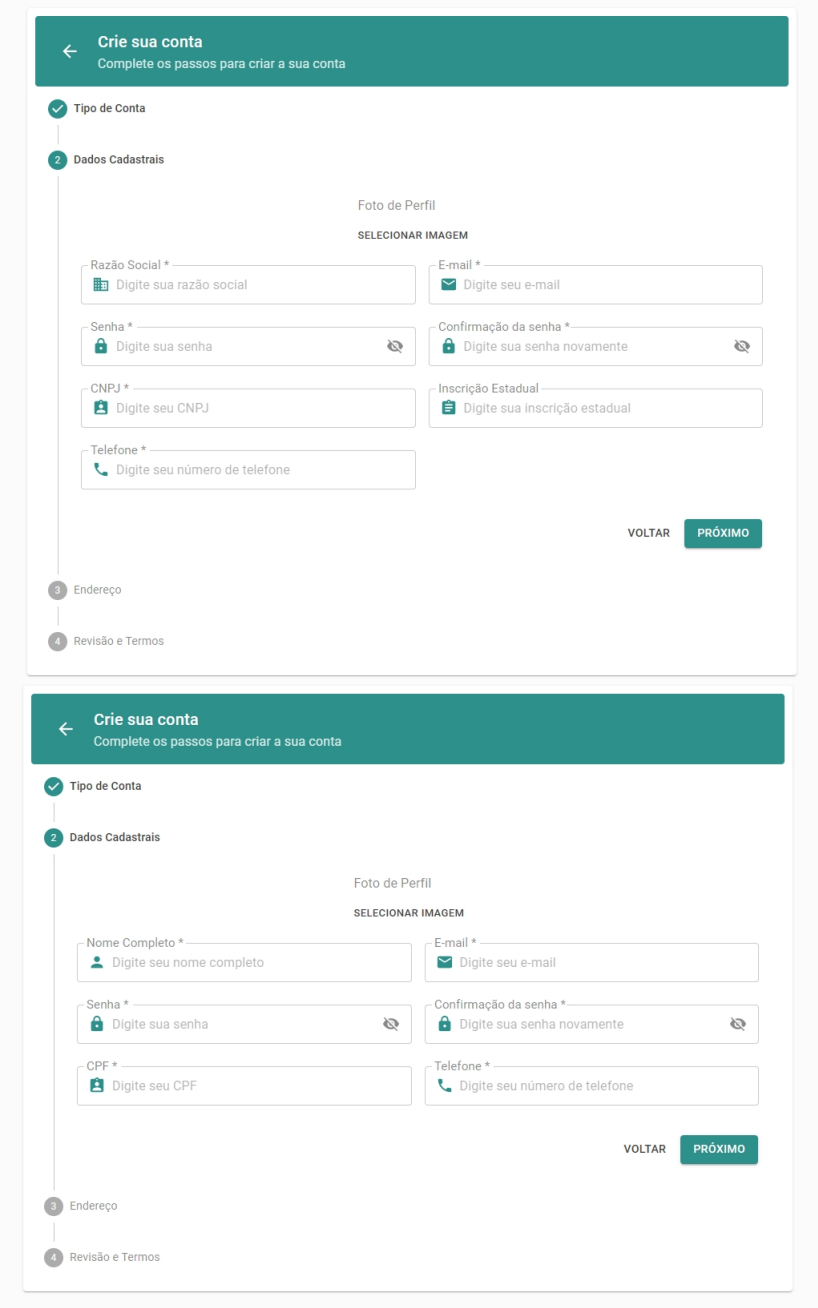
O cadastro de clientes pode ser feito de duas formas atualmente, adicionando manualmente um cliente de cada vez, ou em forma de lotes através de um arquivo do tipo csv, podendo inserir diversos clientes de uma vez.

3.3.2.1 Cadastro manual

Da forma manual, o integrador deve entrar na tela da usina relacionada ao cliente que quer cadastrar, ir até a aba Clientes e clicar no botão de criar. Irá abrir então o formulário de cadastro de cliente, dividido em 2 etapas, a primeira tem somente o campo de email, como pode ser visto na Figura 3.7.

[6] <https://viacep.com.br/>

Figura 3.4 – Etapa 2 de cadastro do integrador

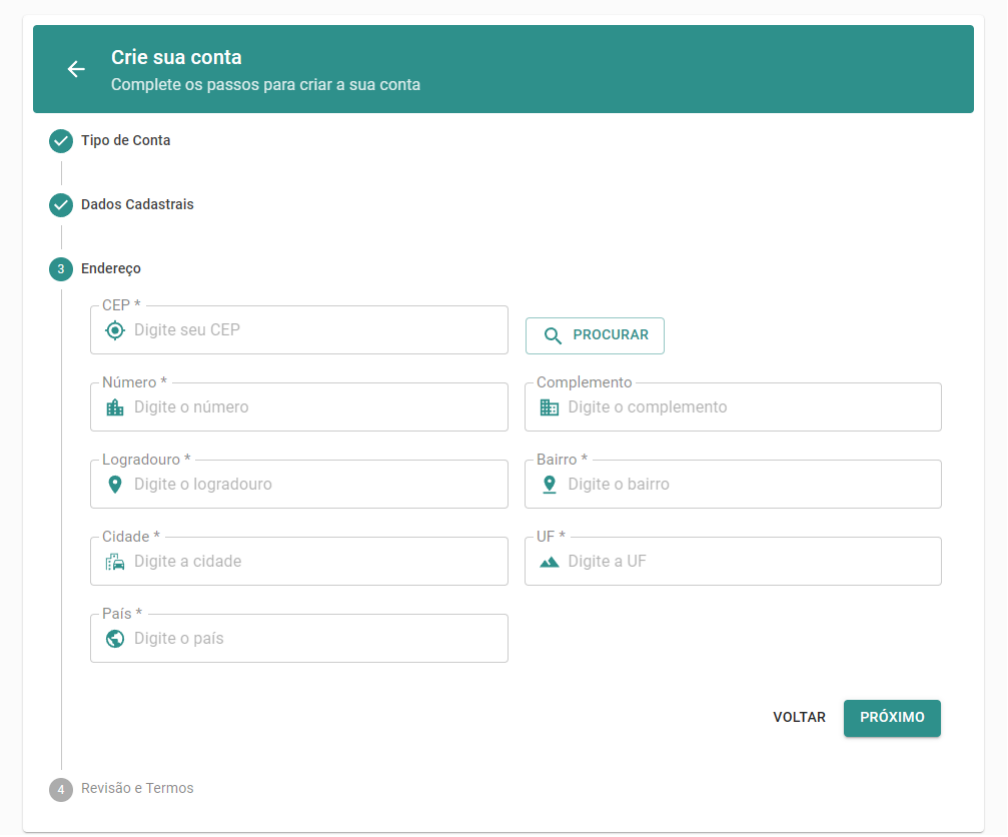


Fonte: GDash (2023)

Caso seja preenchido o email de um cliente já cadastrado, irá aparecer sua foto de perfil, nome e alguns números do CPF, como na Figura 3.8, sendo possível prosseguir para associação do cliente com aquela usina.

Caso o email não esteja cadastrado, será mostrado um botão para criação de um novo usuário, como na Figura 3.9. Clicando neste botão um modal irá se abrir, visto na Figura 3.10, similar ao cadastro de integrador, porém nesse caso a sessão de endereço é opcional e não existem os campos de senha.

Figura 3.5 – Etapa 3 de cadastro do integrador



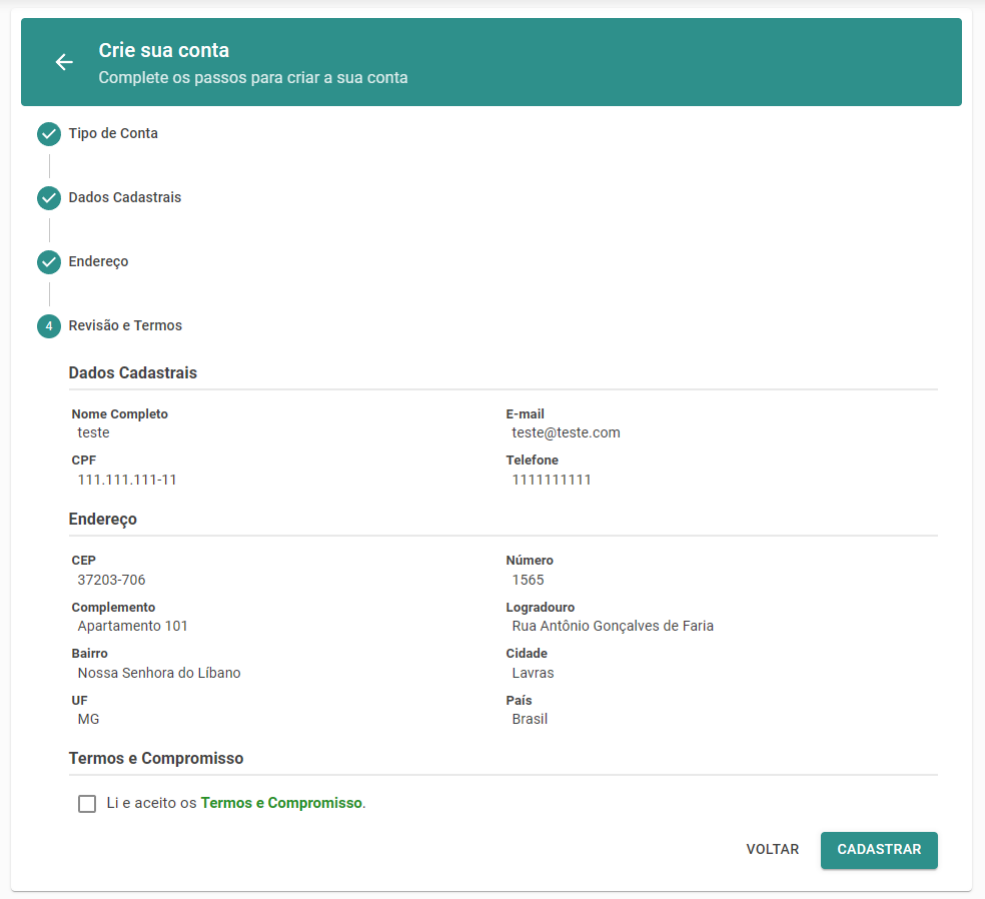
Fonte: GDash (2023)

Ao finalizar o cadastro de um novo cliente ou prosseguir ao inserir o email de um existente, o formulário avança para etapa 2, como vista na Figura 3.11, onde apresenta um resumo de algumas informações desse cliente, e opcionalmente possibilita a inserção de instalações e dos dados utilizados para realização do download e auditoria das faturas, como concessionária e número de cliente. Após a conclusão, o cliente receberá um email onde ele poderá fazer a escolha da senha de acesso, aceitar os termos e finalizar o cadastro.

3.3.2.2 Cadastro em lotes

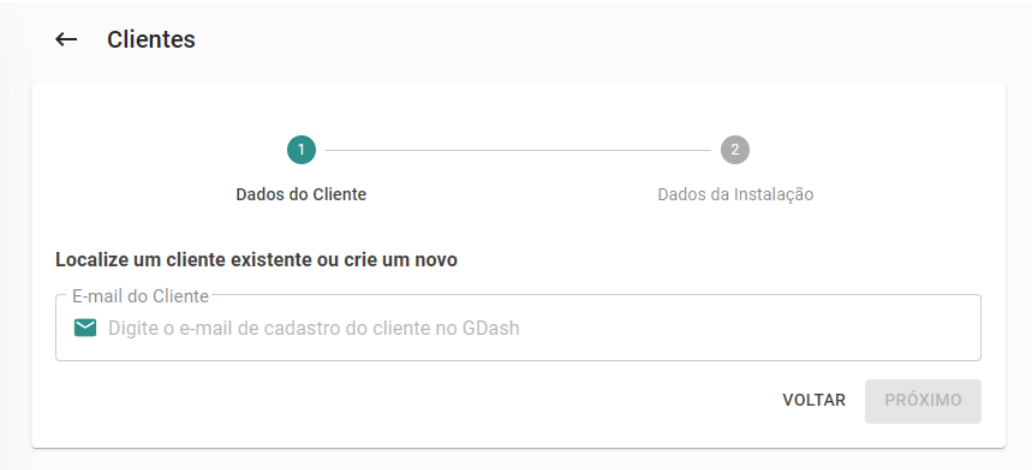
O cadastro em lotes surgiu de uma demanda dos integradores, que necessitavam de uma forma de cadastrar múltiplos clientes de uma vez, principalmente na configuração inicial na plataforma. Nessa forma, é disponibilizado um arquivo csv(*Comma-Separated Values*) para o integrador com todas colunas que devem ser preenchidas, essas sendo as mesmas dos campos da inserção manual. A partir dessa tabela preenchida, o integrador deverá arrastar esse arquivo até o campo de importação presente na página de clientes, como pode ser visto na Figura 3.12. Após a leitura do arquivo, é aberto um modal onde o integrador deverá vincular cada cliente a

Figura 3.6 – Etapa 4 de cadastro do integrador



Fonte: GDash (2023)

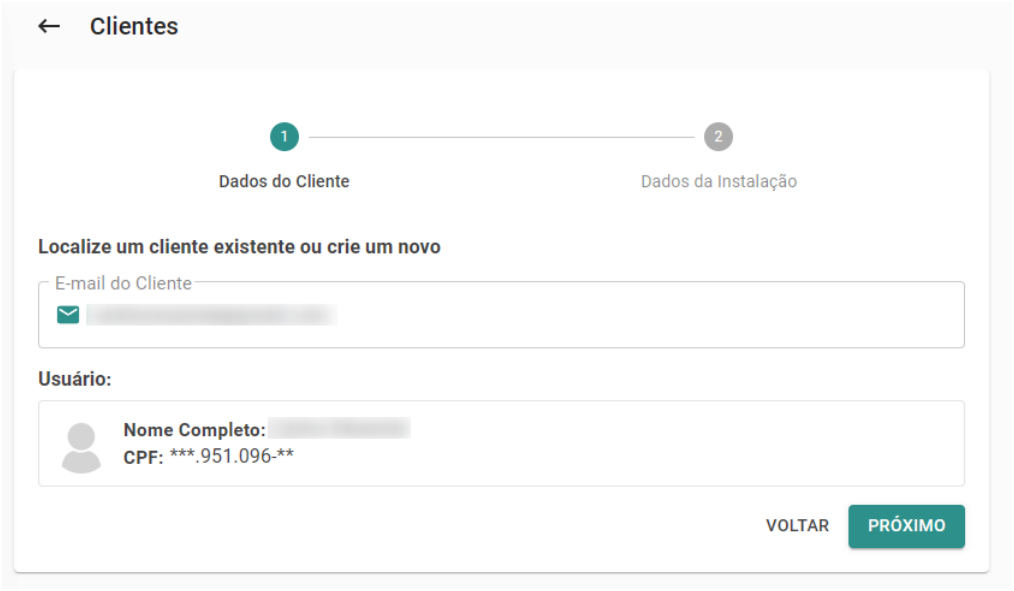
Figura 3.7 – Etapa 1 do cadastro de clientes



Fonte: GDash (2023)

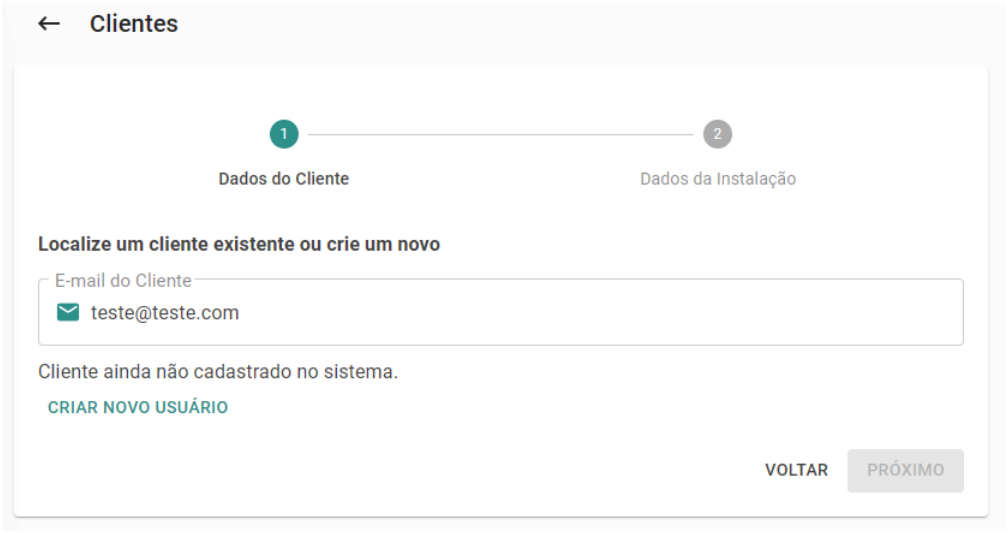
uma usina, e concluindo isso, será enviado um email para os clientes para que eles possam fazer a escolha da senha de acesso, aceitar os termos e finalizar o cadastro.

Figura 3.8 – Cadastro de cliente já existente



Fonte: GDash (2023)

Figura 3.9 – Cadastro de novo cliente

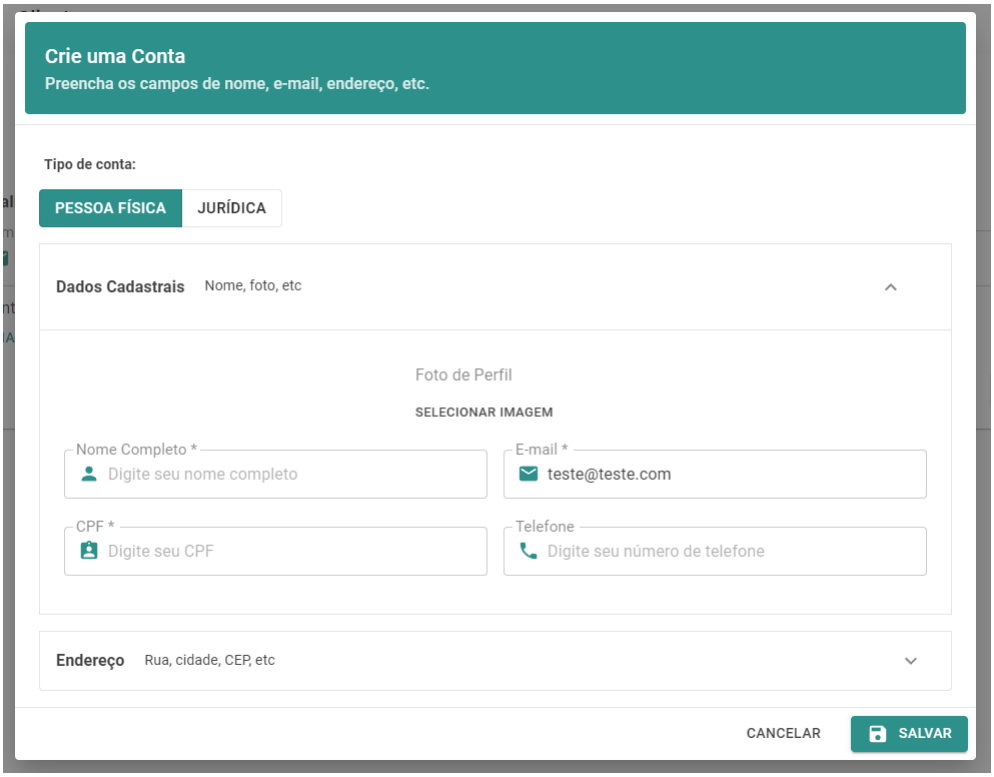


Fonte: GDash (2023)

3.4 Mapa das usinas

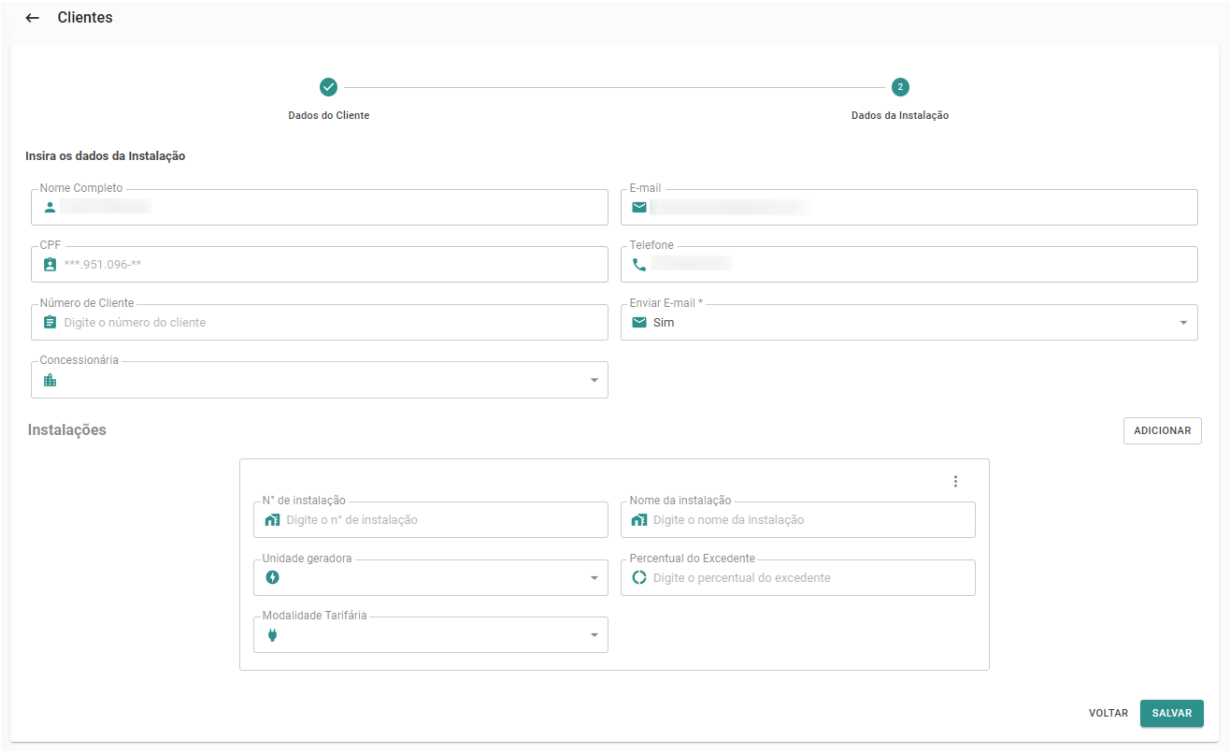
Outra importante tarefa feita pelo estagiário foi a criação total do mapa das usinas, que reúne todas usinas cadastradas pelo integrador em um mapa, servindo com um resumo, mostrando seus status atual e número de chamados abertos, como visto na Figura 2.9.

Figura 3.10 – Modal de cadastro de novo cliente



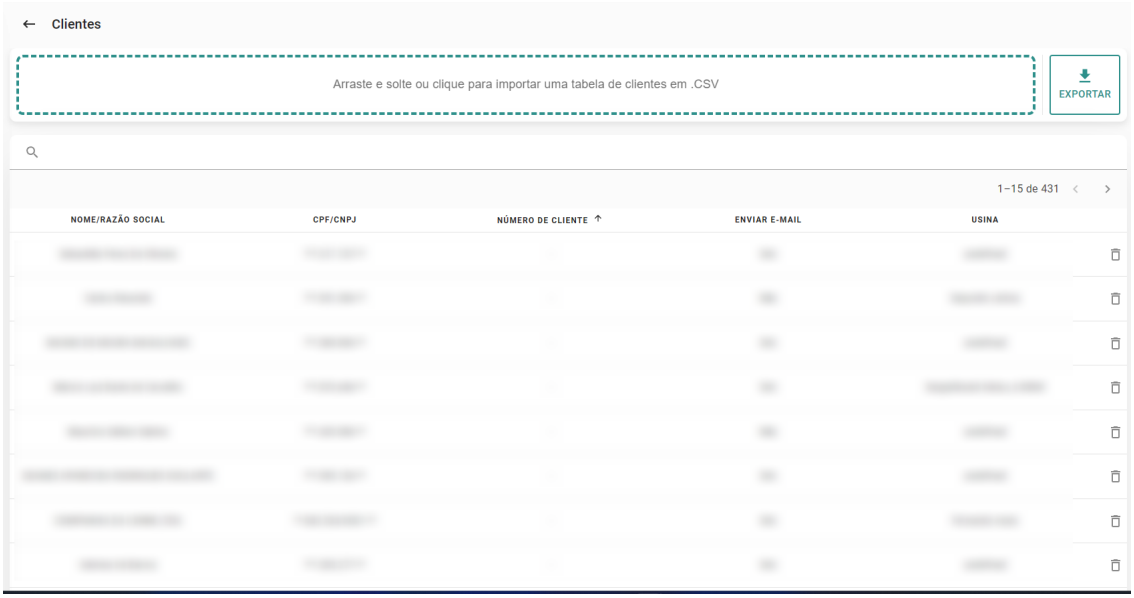
Fonte: GDash (2023)

Figura 3.11 – Etapa 2 do cadastro de clientes



Fonte: GDash (2023)

Figura 3.12 – Importação de clientes



Fonte: GDash (2023)

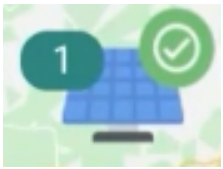
3.4.1 Obtenção das coordenadas

O primeiro passo para a exibição do mapa é a obtenção das coordenadas das usinas, visto que a maioria dos portais das marcas de inversores fornece apenas o endereço. Foi escolhido então a utilização da API Geocoding para conversão desse endereço em coordenadas de latitude e longitude, a princípio toda vez que o mapa era aberto esses endereços eram convertidos, mas devido a baixa eficiência e custo, isso foi mudado, sendo convertido apenas quando a usina é cadastrada, salvando essas coordenadas junto a ela.

3.4.2 Exibição dos marcadores

Com as coordenadas das usinas são criados marcadores posicionados no Google Maps de acordo com a latitude e longitude de cada usina, como visto na Figura 3.13.

Figura 3.13 – Marcador do mapa



Fonte: GDash (2023)

Esses marcadores são compostos por 3 elementos, um ícone de painel solar que serve como a principal forma de simbolizar a usina, o balão à esquerda com a quantidade de chama-

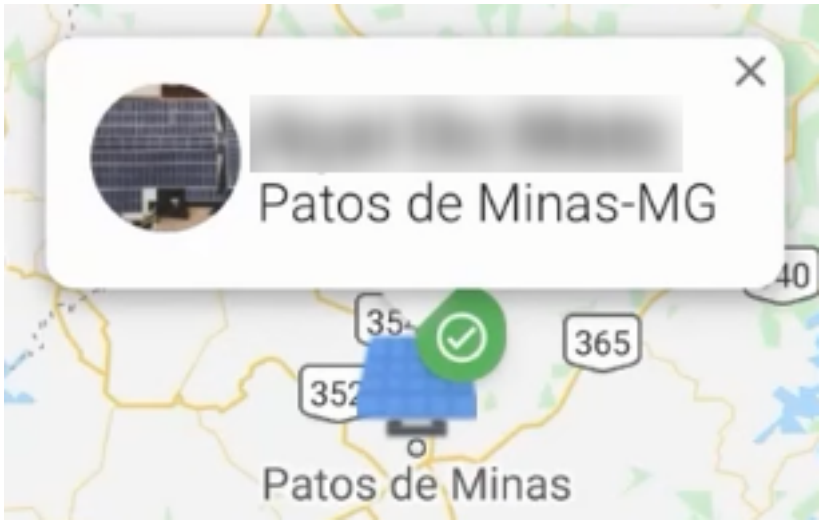
dos abertos naquela usina e a direita o ícone correspondente ao status da usina. Como visto na Figura 3.14, existem 5 tipos de status: nenhum problema encontrado, parte dos inversores offline, um ou mais inversores em alarme, todos inversores offline e sem inversores cadastrados.

Figura 3.14 – Tipos de status da usina

Fonte: GDash (2023)

Os marcadores quando clicados abrem um *popup*, observado na Figura 3.15, onde são mostradas a foto da usina, nome, cidade e estado, e caso a foto ou o nome seja pressionado é aberta a página daquela usina.

Figura 3.15 – *Popup* do marcador



Fonte: GDash (2023)

3.4.3 Filtros de usina

Outra parte importante do mapa são seus filtros, que podem ser acessados através de um botão localizado no lado esquerdo superior. Esse botão abre um *popup*, como visto na Figura 3.16, onde podem ser aplicados filtros como por portal ou por status da usina, além também de ser possível a busca pelo nome de uma usina específica.

Figura 3.16 – *Popup* de filtros

Fonte: GDash (2023)

3.5 Desafios

O projeto do GDash foi desenvolvido inicialmente por uma empresa terceira, como descrito na Seção 3.1, e nesse desenvolvimento foi utilizado um *boilerplate* da empresa, com isso vieram diversas dificuldades e desafios depois que o desenvolvimento do projeto foi trazido de volta.

Um dos desafios enfrentados foi com as versões das dependências, visto que como esse *boilerplate* foi criado pela empresa a anos, muitos *plugins* do MeteorJS estavam desatualizados, e mais gravemente, alguns haviam sido abandonados e não tinham mais atualizações. Com isso havia algumas brechas de segurança no sistema devido ao uso de versões antigas, além também

da falta de algumas funções que vieram nas novas versões. Aos poucos as dependências que tinham novas versões foram sendo atualizadas e buscadas alternativas para as que não tinham novas versões.

Outro grande desafio enfrentado foi quando era necessária qualquer alteração na base desse *boilerplate*, pois todos módulos do sistema, desde a definição de modelos de documentos do banco de dados até a exibição de formulários na tela, passavam por alguma função dessa base. E nenhum membro da equipe tinha conhecimento sobre o funcionamento do *boilerplate*, então toda alteração era demorada e tinha que ser feita cuidadosamente para que não atrapalhasse outras partes do sistema.

Devido ao fato do contrato com a empresa terceirizada, a mesma não ofereceu mais suporte ou esclarecimento a respeito do sistema. Tal fato agravou ainda mais os problemas e dificuldades relacionados a este desafio.

4 CONCLUSÃO

Antes do fim do período de estágio, a plataforma GDash já havia saído do período de testes fechados para parceiros, sendo aberta para usuários pagantes, mesmo que de forma mais restrita e com um contato maior da equipe de suporte. Assim expandindo de acordo com o *feedback* recebido por esses primeiros assinantes.

Esse estágio foi de grande importância para o entendimento de como é um ambiente de trabalho, como se portar diante de dificuldades técnicas, como lidar com prazos de entrega e como gerenciar o tempo entre estudo e trabalho.

Foi muito importante também a relação com os colegas de trabalho, seja na troca de experiências ou nas relações interpessoais, desde de com colegas de equipe ou com o restante da empresa. Tendo um ambiente na empresa muito agradável, principalmente pela cultura da empresa, que permeia em todas suas áreas, de buscar jovens estagiários para se desenvolverem, sempre estando dispostos a ajudar.

Durante o período do estágio, teria sido interessante buscar um profissional mais experiente que atuasse como gerente de projetos ou arquiteto, visto que faltava alguém com experiência técnica e de gestão na tomada de decisões de arquitetura e design de *software*, que caiam muitas vezes sobre responsabilidade do estagiário.

Destaca-se o fato dos aprendizados adquiridos no curso de Sistemas de Informação. No decorrer dos períodos, haviam trabalhos progressivamente mais complexos passados pelos professores, que estimulavam a busca de soluções por conta própria, que acabou preparando muito para o que ocorreu durante o estágio.

Ressalta-se também os primeiros contatos com diversas tecnologias e conceitos ao longo do curso, não só nas disciplinas mas também em todo ambiente da universidade, em conversas com professores e outros alunos, que acabaram influenciando no desenvolvimento do estagiário e permitindo que chegasse preparado no estágio.

Algumas disciplinas foram muito importantes no desenvolvimento e preparação do estagiário, podendo destacar Introdução a Sistemas de Banco de Dados e principalmente Interação Humano-Computador, no meio do período de estágio o estagiário decidiu focar em tarefas *frontend*, tendo essa disciplina influenciado muito nas questões de usabilidade e acessibilidade durante o estágio.

Outro ponto interessante foi a abordagem do Scrum por algumas disciplinas, uma vez que grande parte das empresas de TI fazem a utilização do *framework*. Porém o estagiário

só foi cursar essas disciplinas depois do período de estágio, talvez sendo interessante que essas disciplinas fossem alocadas para os períodos mais iniciais, visto que esses conceitos não ajudam somente na preparação para o mercado de trabalho, mas também podem ajudar na organização de estudos e trabalhos da universidade.

REFERÊNCIAS

ABSOLAR. **Panorama da solar fotovoltaica no Brasil e no mundo**. 2022. Disponível em: <https://www.absolar.org.br/mercado/infografico/>. Acesso em: 20 dez. 2022.

COMSCORE. **Top 25 Smartphone Apps December 2022**. 2022. Disponível em: <https://www.comscore.com/Insights/Rankings?country=US>. Acesso em: 28 jan. 2023.

CRUZ, F. **Scrum e Agile em Projetos - 2ª Edição**. [S.l.]: Editora Brasport, 2018. ISBN 9788574528793.

DATE, C. J. **Introdução a Sistemas de Bancos de Dados - 8ª edição**. [S.l.]: Campus, 2003. ISBN 9788535212730.

DOCS, M. **Database**. 2023. Disponível em: <https://www.mongodb.com/atlas/database>. Acesso em: 24 jan. 2023.

DOCS, M. **Documents**. 2023. Disponível em: <https://www.mongodb.com/docs/manual/core/document/>. Acesso em: 12 jan. 2023.

G1. **Estiagem faz despencar uso das hidrelétricas para o menor nível já registrado no Brasil**. 2021. Disponível em: <https://g1.globo.com/jornal-nacional/noticia/2021/08/11/estiagem-faz-despencar-uso-das-hidreletricas-para-o-menor-nivel-ja-registrado-no-brasil.ghtml>. Acesso em: 28 jan. 2023.

GDASH. **GDash**. 2023. Disponível em: <https://www.gdash.com.br/>. Acesso em: 12 jan. 2023.

MDN. **Classes**. 2023. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Classes>. Acesso em: 12 jan. 2023.

MDN. **JavaScript**. 2023. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 12 jan. 2023.

METEOR. **DDP Specification**. 2021. Disponível em: <https://github.com/meteor/meteor/blob/devel/packages/ddp/DDP.md>. Acesso em: 09 jan. 2023.

MONGODB. **MongoDB**. 2023. Disponível em: <https://www.mongodb.com>. Acesso em: 12 jan. 2023.

NODE.JS. **How do I start with Node.js after I installed it?** 2023. Disponível em: <https://nodejs.org/en/docs/guides/getting-started-guide/>. Acesso em: 09 jan. 2023.

NODE.JS. **Não bloqueie o Event Loop (ou a Worker Pool)**. 2023. Disponível em: <https://nodejs.org/pt-br/docs/guides/dont-block-the-event-loop/>. Acesso em: 09 jan. 2023.

REACT. **React**. 2023. Disponível em: <https://pt-br.reactjs.org/>. Acesso em: 09 jan. 2023.

REACT. **Tutorial: Intro to React**. 2023. Disponível em: <https://reactjs.org/tutorial/tutorial.html>. Acesso em: 09 jan. 2023.

SCHWABER, K.; SUTHERLAND, J. **Um guia definitivo para o Scrum: As regras do jogo**. 2013. Disponível em: <https://scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf>.

SCRUM.ORG. **WHAT IS SCRUM?** 2023. Disponível em: <https://www.scrum.org/resources/what-is-scrum>.

SOL, A. do. **Potencial solar no Brasil**. 2022. Disponível em: <https://americadosol.org/potencial-solar-no-brasil/>. Acesso em: 28 jan. 2023.

SOLARVIEW. **Login**. 2023. Disponível em: <https://my.solarview.com.br/login>. Acesso em: 13 jan. 2023.

SOLARZ. **Configurações das usinas**. 2023. Disponível em: <https://solarz.octadesk.com/kb/article/3-configuracoes-das-usinas>. Acesso em: 13 jan. 2023.