Resultados

**Métricas Temperatura Café:**



O café não tolera temperaturas acima dos 34°C, podendo levar ao abortamento dos botões florais e também não tolera temperaturas abaixo dos 2°C, podendo levar a morte dos pés.

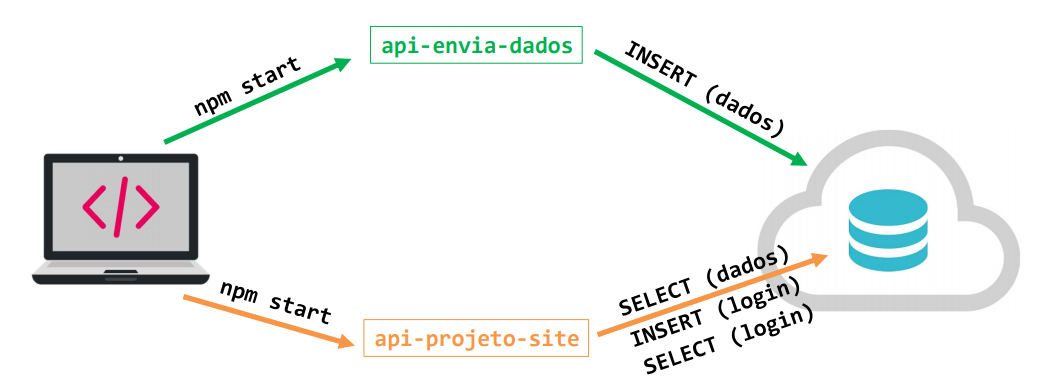
As regiões que apresentam temperatura média que varie entre 19°C e 22°C são as mais recomendadas para o cultivo do café. A variedade “robusta” tolera até 24°C. Já as regiões mais frias ou com temperaturas abaixo dos 18°C não são propícias para esse cultivo.

**Código inicial do projeto para análise e controle de temperatura:**

**API-ENVIADADOS**

Config.js será o arquivo utilizado para conexão com o banco de dados Azure, basta apenas inserir seus dados.

Na parte de “api-sensor.js” criamos o sql para inserir os dados na nossa tabela leitura que será usada na parte de “obterTemperatura()” onde ele ira pegar os dados e assim o ciclo será iniciado:



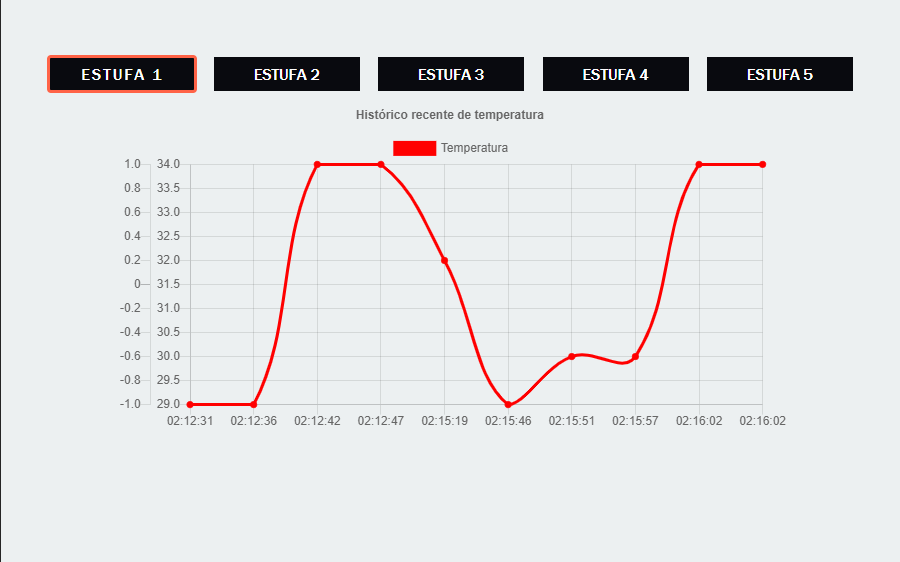
**API BOLTTECH**

Primeiro devemos configurar nossa Api para se conectar com nosso banco de dados do Azure na parte de arquivo “**config.js**

Ao entrarmos na área de Dashboard obtemos a temperatura logo de inicio pela função “obterdadosestufa()” onde utilizamos uma fecht para nós nos comunicarmos com o arquivo “**/leituras/tempo-real/${e capturamos o id da estufa desejada}”,** que se encontra no arquivo “leituras.js”.

Em “leituras.js” na router do tipo “get” temos o “/tempo-real/idSensor (o id da estufa já capturado)”, criamos a function com os parâmetros req de requisição e res de resposta, criamos a variável “idSensor” e passamos os parâmetros da requisição e assim na validação, por meio do Azure ou do env “production”, nós criamos a nossa instrução sql para capturarmos as informações do banco Azure por meio do sequelize e assim gerar um resultado.

Temos outros tipos de rotas onde capturamos as últimas temperaturas obtidas de nossos sensores e injetamos ela no gráfico do “chart.js” onde definimos o limite de linhas que irá aparecer no gráfico para analisar os dados. Sendo assim uma variável armazena a informação sql que irá até o banco de dados para recuperar as informações desejadas.

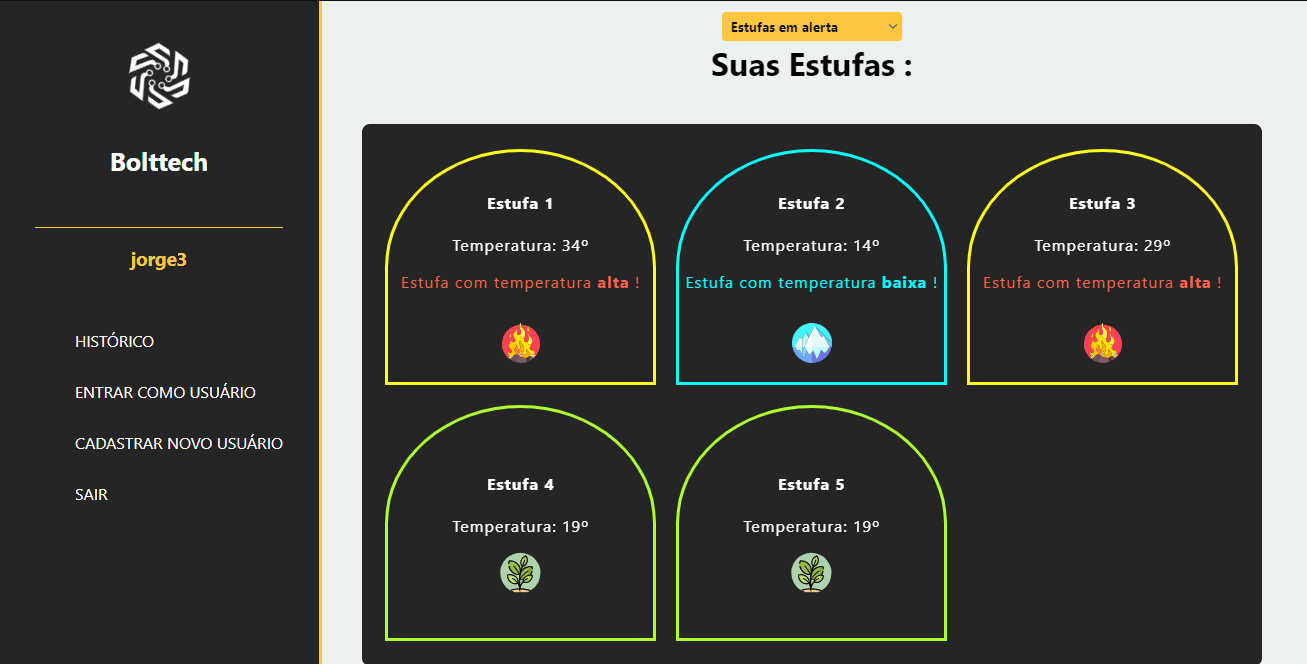


Temos nossas estatísticas de temperatura onde será analisada a média, o mínimo e a máxima temperatura da nossa tabela de “leitura”.

Voltando na function “obterdadosestufa()” podemos visualizar melhor a variável dados que será passada como parâmetro na função “alerta” e “atualizartela”, na Var dados será passada a resposta do fecht que contém as informações do nosso banco de dados, “resposta.temperatura”, que esta passando a temperatura por exemplo.

Na função “alertar” fazemos uma validação a partir dos dados obtidos e assim alteramos os dados do nosso “front-end” para uma melhor visualização e usabilidade do cliente.

No exemplo abaixo podemos observar que as bordas da figura da estufa que esta fria (estufa 2) ficaram azul e a que esta quente (estufa 1) ficaram amarelo piscante e ambas possuem um texto informativo de aviso.



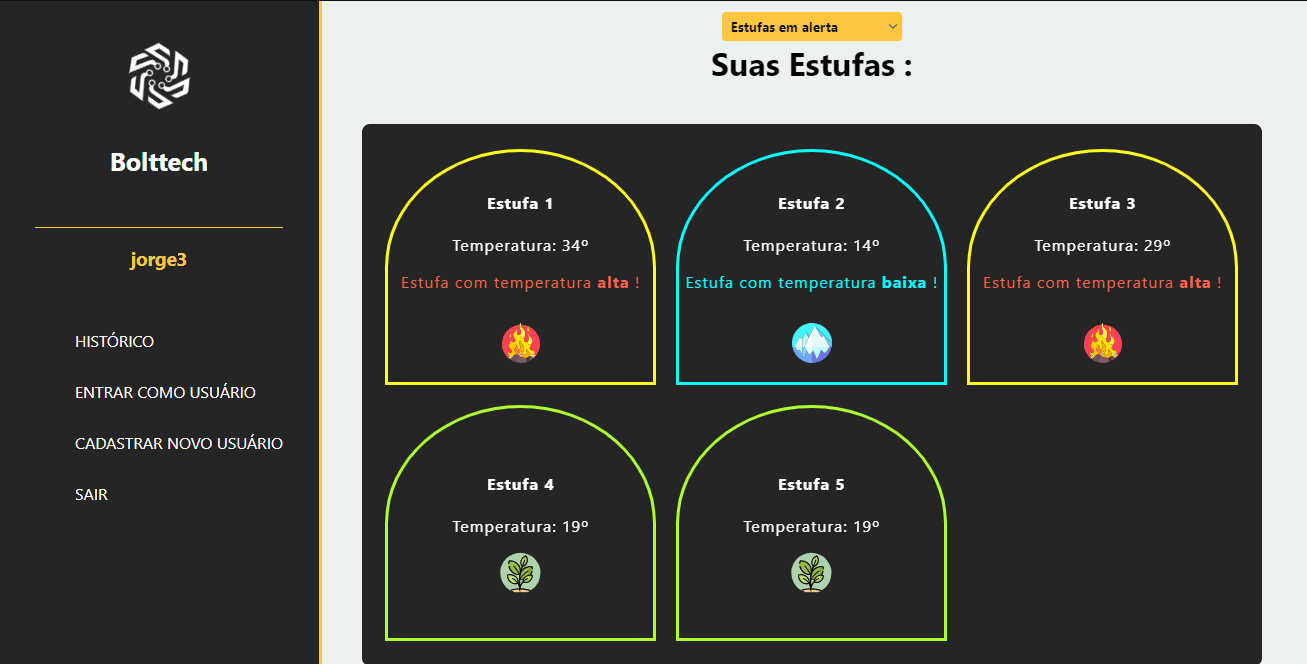
Na função atualizar tela ele captura o id e os dados obtidos para assim realizar a atualização que será realizada pela função “sendData”.

A função “sendData()” ela envia os dados a partir de uma requisição da página local 9001 utilizada na API-ENVIADADOS que simulando o envio de informação do Arduíno.

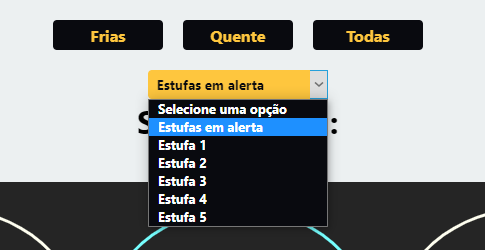
Com as duas APIs já em funcionamento e conectada ao banco de dados Azure temos o sistema Bolttech completo onde por meio dos resultados obtidos com a APi envia dados criamos as métricas de ALERTA PERIGO onde analisamos se a temperatura ambiente da estufa esta de acordo com os padrões internacionais das boas praticas de plantio de café, caso aja algum erro o usuário será informado imediatamente no momento em que o programa se inicia, criando assim um sistema inteligente de monitoramento de temperatura de cada estufa.

**Usabilidade**

Na parte de dashboard temos de maneira intuitiva todas as estufas registradas do cliente onde ele por meio de um botão ele pode analisar estufas que estejam quentes, frias ou até mesmo a partir de um select selecionar exatamente a estufa que deseja observar.



Na parte de Selecionar sua estufa por padrão as estufas que serão exibidas ao usuário seriam as estufas frias e quentes e essa opção seria a “bolttech”.



Temos a parte de histórico onde registramos os gráficos em tempo real de cada estufa do usuário e cada ponto é definido no código na rota “ULTIMA/IDSENSOR” no arquivo “leituras.js”.



Temos também a parte de Usuário onde temos a nossa “base de conhecimento” Bolttech onde usuários de todas as empresas Bolttech trocam informações que podem ser úteis no desenvolvimento geral da parceria.



O usuário também tem acesso às estufas de sua empresa com as mesmas funcionalidades.

**Aprendizado durante o desenvolvimento do projeto**

Detalhamento e visão do grupo em relação ao aprendizado durante o desenvolvimento do projeto.

Durante o processo de criação deste projeto entramos em contato com diversas tecnologias como, CSS, HTML, JS, NODE JS e implementação de gráfico da biblioteca chart.js, help desk com os serviços do Jira.

**HMTL**

Estruturação de elementos HTML com uma boa semântica, no exemplo acima, utilizando div como caixas de conteúdo e h4 como caixas de texto e span como uma tag coringa produzindo assim uma boa semântica de código HTML para uma boa leitura, utilização de **“class”, ”value”, ”id”** e **“name”**.

**CSS**

LINKAGEM CSS



Criação de caixas de classes e IDs para manipulação em HTML.

**Pseudo-classe**

Palavra-chave adicionada a seletores que especifica um estado especial do elemento selecionado. Por exemplo, :hover pode ser usado para alterar a cor de um botão quando o usuário passar o cursor sobre ele.

**Animações:**

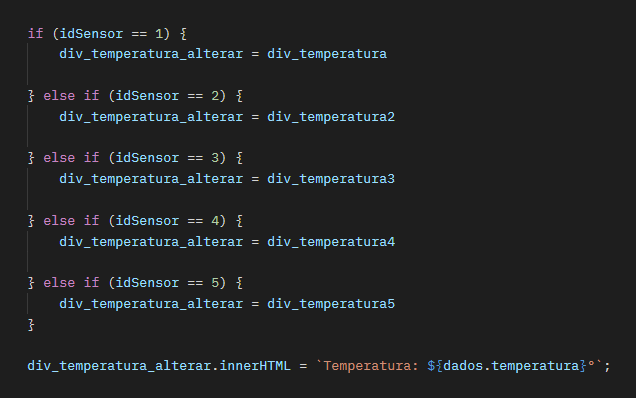
A animação é criada mudando gradualmente de um conjunto de estilos CSS para outro. Durante a animação, você pode alterar o conjunto de estilos CSS várias vezes.

Logo animado:

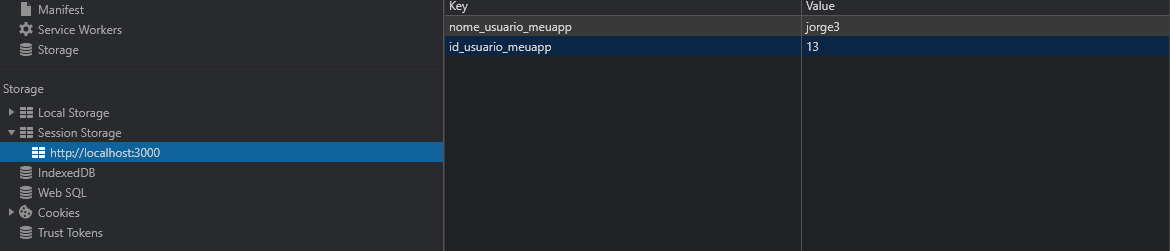


**JAVASCRIPT**

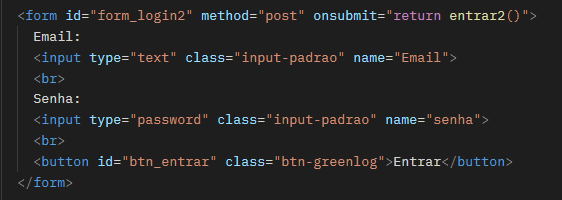
Validações com If elf if e else



Criação de funções para manipulação do Html por meio da utilização das funcionalidades do ID. Utilização de sessionStorage como método de salvar informações desejáveis.



Utilização de Eventos em Javascript como Onsubmit, Onclick, Onchange entre outros.



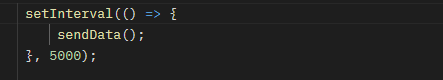
Criar objetos HTML a partir de JS por meio da função “**creatElement()”,** injetando as divs criadas por meio do **appendChild** alterando html por meio do método innerHTML, utilização de **JSON** na variável publicação sendo usada no exemplo abaixo na alteração innerHTML como “**publicao.nomeUsuario”.**

Assíncronicidade por meio da Utilização de “FETCH()” para requisições de outras páginas.

Quando uma função assíncrona é chamada, ela retorna uma Promise . Quando a função assíncrona retorna um valor, a Promise será resolvida com o valor retornado. ... A proposta das funções async/await é de simplificar o uso de forma síncrona das Promises e executar alguns procedimentos em um grupo de Promises .

Implementação de gráfico pela biblioteca “chart.js”.

Método SETINTERVAL, uma função que opera com intervalos especificados (em milissegundos) sendo assim no exemplo abaixo a funcção se repetira a cada 5 segundos e chamara ela mesma sendo uma função declarada dentro de outra.



Manipulação de classe CSS via JS para os ALERTAS das estufas

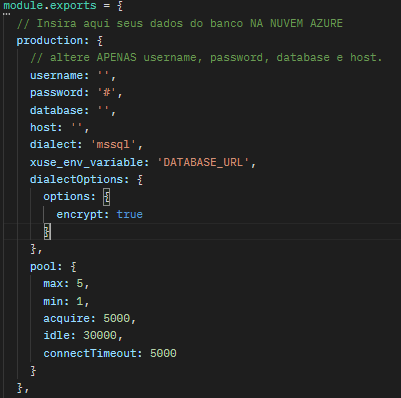
Foi utilizada os elementos classList.add() e classList.remove() para fazer as alterações de classe utilizadas por cada estufa juntamente com a validadeção if e else para precisão métrica

**Azure SQL SERVER**

Criação do banco de dados e criação das tabelas por meio do Azure, utilizada a linguagem SQL.

**NODE.js**

Conexão com banco de dados via Javascript.

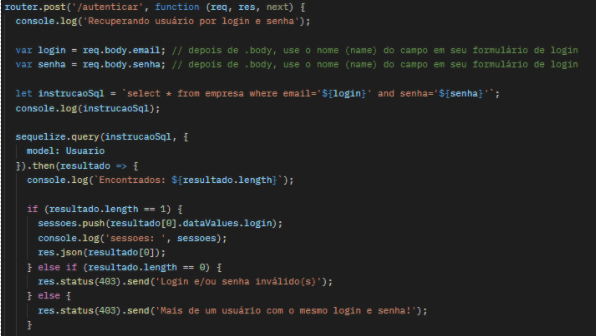


Mapeamento do banco em node utilizando a biblioteca Sequileze para consulta no banco de dados Workbench.

UTILIZAÇÃO SIMPLES DO REQUIRE PARA CAPTURAR INFORMAÇÕES DA VARIÁVEL USUÁRIO

ROTA para cadastro do usuário utilizando sequelize e a função “create” para a ponte para a tabela do Azure.

Criação de ROTAS para validar e autentificar usuário por meio da consulta de banco de dados utilizando o Sequelize.



Realização de INSERT por meio da API-ENVIA –DADOS

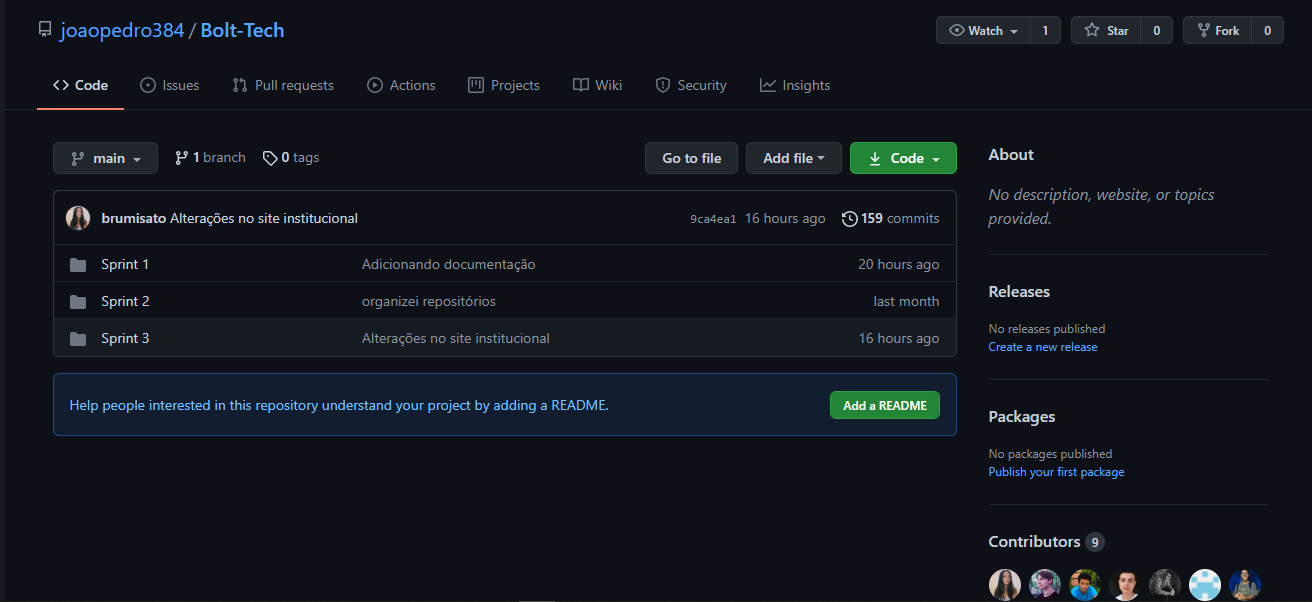
GIT, BASH E NPM

npm-i. Instale módulos npm e dependências em locais arbitrários do código

Isso executa um comando predefinido de um pacote.

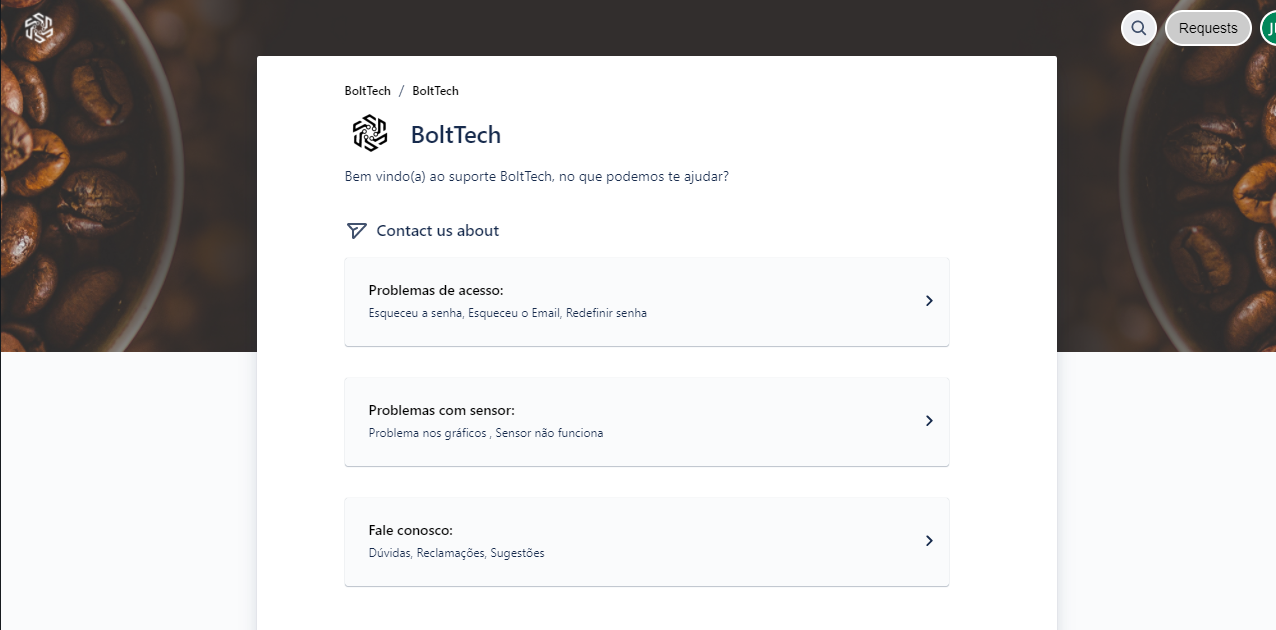
GitHub

Controle de projeto e atualização no repositório do GITHUB



**Help desk**

Configuração por meio do Jira, um ambiente de controle de trabalho onde podemos analisar todos nossos, problemas, incidentes e chamados para um melhor serviço de apoio a usuários para suporte e resolução de problemas técnicos.



## **Considerações finais**

Nós da Bolttech estamos animados com a evolução deste projeto com uma visão de forte otimismo, com os acontecimentos atuais envolvendo a área agropecuária e tecnologia, onde cada vez mais temos visto o desenvolvimento de estufas para uma competição larga e global de maneira limpa e inteligentemente econômica, de um grau de cuidado por métricas exatas o que permite a segurança do produto. Caso haja mais tempo e dedicação no projeto em versões futuras acreditamos que poderemos até mesmo dobrar a economia orçamentária de cada estufa em relação a uma plantação ao ar livre por meio da eliminação de agrotóxicos e perdas pelo mau uso da água entre outras adversidades. Nossa plataforma seria apresentada de maneira simples e intuitiva para nosso fornecedor ajudando assim com o auxilio de sua plantação e um aumento na comunicabilidade de seus funcionários por meio da troca de base de conhecimento dos parceiros da rede Bolttech.