FACULDADE DE TECNOLOGIA SENAI JARAGUÁ DO SUL - SC GRADUAÇÃO TECNOLÓGICA EM SISTEMAS PARA INTERNET

SMART CARE
SOFTWARE DE GESTÃO EMPRESARIAL COM FOCO EM ORDENS DE MANUTENÇÃO

DOUGLAS PENNA BASTOS BLANK
FERNANDO MARQUES CANDIDO
GUSTAVO GIESE
RONEI JOSÉ ROTESKI

JARAGUÁ DO SUL, SC
2020

FACULDADE DE TECNOLOGIA SENAI JARAGUÁ DO SUL - SC GRADUAÇÃO TECNOLÓGICA EM SISTEMAS PARA INTERNET

DOUGLAS PENNA BASTOS BLANK
FERNANDO MARQUES CANDIDO
GUSTAVO GIESE
RONEI JOSÉ ROTESKI

SMART CARE
SOFTWARE DE GESTÃO EMPRESARIAL COM FOCO EM ORDENS DE MANUTENÇÃO

JARAGUÁ DO SUL, SC
2020

SUMÁRIO

Aplicação Web

Vue.js:

Vue.js é um framework JavaScript de código-aberto, focado no desenvolvimento de interfaces de usuário e aplicativos de página única.

Este framework é voltado a componentização, em outras palavras, o desenvolvimento se volta à criação de componentes, para formar compor o visual final. Com este Framework é possível a criação de toda parte visual e de interação com usuário do sistema.

Componentização:

Levando em conta a utilização de um framework voltado a componentização, é natural a extensa utilização de componentes no projeto, estes têm como objetivo fragmentar uma parte do sistema, com o intuito de reutilizar e dividir o código do sistema.

Os benefícios da componentização vão além da capacidade de reutilizar o código, uma vez que o mesmo quando bem utilizado pode reduzir o custo da manutenção futura no projeto, evitando que o mesmo se torne demasiadamente custoso.

Estrutura das pastas:

```
src/..................... Ponto de entrada dos códigos
 |assets/................. Arquivos estáticos
 |components/............. Componentes Vue globais
  |mobile/................ Componentes Vue globais para a versão mobile
  |web/................... Componentes Vue globais para a versão WEB
 |plugins/................ Plugins usados pela aplicação
 |routes/................. Sistema de roteamento
 |services/............... Gerenciadores de requisições
 |store/.................. Gerenciador de estado global Vuex
 |utils/.................. Utilitários do sistema
 |view/................... Páginas principais e seus componentes
  |Analysis/............... Módulo de análise
   |components/............ Componentes do módulo de análise
  |Core/................... Telas principais do sistema (Configuração, Login, Dashboard)
   |mobile/................ Componentes mobile para os arquivos do core
   |web/................... Componentes WEB para os arquivos do core
  |Movimentations/......... Telas responsáveis pelas movimentações e execuções das ordens
   |consult/............... Contém os arquivos e componentes para a tela de consulta
    |components/........... Componentes individuais da tela de consulta
     |mobile/.............. Componentes da versão Mobile da tela de consulta
     |web/................. Componentes da versão WEB da tela de consulta
    |util/................. Utilitários para a tela de consulta
   |detail/................ Contém os arquivos e componentes para a tela de detalhamento
    |components/........... Componentes individuais da tela de detalhamento
     |mobile/.............. Componentes da versão Mobile da tela de detalhamento
     |modal/............... Janelas de diálogo (Modais)
     |web/................. Componentes da versão WEB da tela de detalhamento
    |util/................. Utilitários para a tela de detalhamento
  |Register/............... Contém todos os arquivos pertinentes aos cadastros do sistema
   |Cruds/................. Cadastros diversos (Usuário, operações, equipamentos)
   |MaintananceOrder/...... Cadastros das ordens de manutenção
 |app.scss
 |App.vue
 |main.js
```

Fluxo:

O sistema inicia no arquivo *main.js* onde o vue monta toda a aplicação:

```
new Vue({
  async beforeCreate() {
    moment.locale('pt-BR');

    try {
      if (router.currentRoute.name === 'login') return;

      await validateSession(router.options.apiUrl);
    } catch (err) {
      console.log('session err => ', err);

      const { response } = err;      You, 8 months ago •

      localStorage.removeItem('token');

      if (!response.data)
        return throwError();

      if (response.data.name === 'TokenExpiredError')
        return router.replace('/');

      return throwError();
    }
  },
  router,
  store,
  render: h => h(App),
}).$mount('#app');
```

Logo em seguida o arquivo *app.vue* é responsável pela renderização das rotas e componentes de menu lateral e menu no topo:

```
<template>
  <div id="app">
    <div
      class="content"
      :class="isLoginScreen ? 'justify-content-center align-items-center' : ''"
    >
      <transition
        name="sidebar-slide"
        @after-leave="afterLeaveSidebar()"
        @before-enter="beforeEnterSidebar()"
      >
        <div
          v-if="!state.isMobile && !isLoginScreen"
          class="sidebar-content shadow"
          :class="isSidebarHided ? 'hided-sidebar' : ''"
        >
          <sidebar />      You, 3 months ago • improved system transition
        </div>
      </transition>
      <div class="wrapper">
        <transition name="header-slide">
          <div v-if="!state.isMobile && !isLoginScreen" key="header" class="topbar-content">
            <Header />
          </div>
        </transition>

        <transition name="header-slide">
          <div v-if="state.isMobileAfterSidebarLeave && !isLoginScreen" key="mobileHeader" c
            <mobile-header />
          </div>
        </transition>
        <div
          key="router"
          :class="state.isMobile && isLoginScreen ? 'content justify-content-center align-it
        >
          <transition
            name="slide-fade"
            mode="out-in"
            v-on:after-enter="afterEnterRouter()"
          >
            <router-view />
          </transition>
        </div>

        <transition name="fade">
```

A primeira tela renderizada é a de *Login.vue* que exibe toda a interface para realizar a autenticação:

```
methods: {
  async loginValidation() {
    if (this.isLoading === true) return;

    try {
      this.isLoading = true;

      const response = await this.$http.post('users', this.inputValues);

      this.$store.commit('addUser', {
        email: response.email,
        nome: response.nome,
        nivelAcesso: response.nivel_acesso,
        funcao: response.funcao,
        cracha: response.numeroCracha,
        userId: response.idUsuario,
      });

      this.setTokenLocalStorage(response.token);

      this.$swal({
        position: 'top',
        type: 'success',
        toast: 'true',
        title: 'Autenticado com sucesso!',
        showConfirmButton: false,
        timer: 1500,
      }).then(() => {
        this.isLoading = false;
        this.$http.setActivity(this.$activities.LOGIN);

        this.$router.push('dashboard');
      });
    } catch (err) {
      console.log('err login => :', err.response || err);

      this.isLoading = false;

      return this.$swal({
        type: 'error',
        title: 'Não foi possível realizar o login',
        html: getErrors(err),
        confirmButtonColor: '#F34336',
      });
```

Após a autenticação, será redirecionado pelo **vue-router** para a tela de dashboard:

```vue
<template>
  <div class="root-dashboard-view">
    <template v-if="isMobile">
      <dashboard-mobile
        :orders="orders"
        :is-loading="isLoading"
        :has-errors="hasErrors"
      />
    </template>

    <template v-else>
      <dashboard-web
        :orders="orders"
        :custom-styles="customStyles"
        :quantity="quantity"
        :chart-style="chartStyle"
        :labels="labels"
        :is-loading="isLoading"
        :has-errors="hasErrors"      Fernandocm1, 4 months ago • feat: custo
        :loaded="loaded"
      />
    </template>
  </div>
</template>

<script>
import { getToken } from '../../utils/utils';
import { mapGetters } from 'vuex';

export default {
  components: {
    DashboardMobile: () => import('./mobile/Dashboard-mobile.vue'),
    DashboardWeb: () => import('./web/Dashboard-web.vue'),
  },
  data() {
    return {
```

API

Arquitetura Monolítica:

Esta arquitetura é desenvolvida para ser instalada num só lugar, geralmente possuem camadas sobrepostas com responsabilidades distintas e quando necessita realizar algum ajuste, todo o sistema deve ser atualizado no servidor de produção. Em outras palavras a arquitetura monolítica todos os módulos estão em uma mesma estrutura, sendo qualquer alteração seja uma simples ou complexa, a necessidade de refazer todo processo de implantação, desde testes, compilação caso a linguagem seja deste tipo até o build em produção.

Estrutura das Pastas:

```
src/                        |
  services/                 | Todos os serviços e funcionalidades exclusivas da api
    dao/                    | Comunicação e manipulação do banco de dados
      cruds/                | Todos os cruds que o sistema possui
      movimentations/       | Todas as movimentações no sistema
    session/                | Classes, validações e regras de negócio do sistema
      cruds/                | Todos os cruds que o sistema possui
      movimentations/       | Todas as movimentações no sistema
    routes/                 | Rotas de endpoints express
      cruds/                | Todos os cruds que o sistema possui
      movimentations/       | Todas as movimentações no sistema
  shared/                   | Tudo o que é compartilhado entre os services
    auth/                   | Autenticação do token
    constants/              | Constantes/Enums globais
    database/               | Middleware de conexão com o banco de dados
    guard/                  | Criptografia dos usuários
    utils/                  | Tem funções uteis para diversas partes do sistema
  index.js                  | Ponto inicial do sistema
```

Fluxo:

Todas as requisições que chegam para a API passam pelo arquivo index.js, que faz todo o roteamento da mesma:

```
const express = require('express');
const cors = require('cors');
const bodyParser = require('body-parser');

const ConnectionFactory = require('./shared/database/ConnectionFactory');
const Auth = require('./shared/auth/auth');

require('dotenv').config({ path: '.env' });

const app = express();
const connectionFactory = new ConnectionFactory();
const auth = new Auth();

app.use(cors());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());

app.use(connectionFactory.createConnection.bind(connectionFactory));
app.use(auth.run.bind(auth));

// CRUDS
const User = require('./services/routes/cruds/User');
const Status = require('./services/routes/cruds/Status');
const Symptom = require('./services/routes/cruds/Symptom');
const AccessLevel = require('./services/routes/cruds/AccessLevel');
const MaintenanceOrder = require('./services/routes/cruds/MaintenanceOrder');
const Equipment = require('./services/routes/cruds/Equipment');
const InstallationLocation = require('./services/routes/cruds/InstallationLocation');
const Cause = require('./services/routes/cruds/Cause');
const Epi = require('./services/routes/cruds/Epi');
const OrderType = require('./services/routes/cruds/OrderType');
const Priority = require('./services/routes/cruds/Priority');
const WorkCenter = require('./services/routes/cruds/WorkCenter');
const Operation = require('./services/routes/cruds/Operation');
const Component = require('./services/routes/cruds/Component');

app.use('/users', User);
app.use('/status', Status);
app.use('/sintoma', Symptom);
app.use('/nivel-acesso', AccessLevel);
app.use('/ordem-manutencao', MaintenanceOrder);
app.use('/equipamento', Equipment);
app.use('/componente', Component);
app.use('/local-instalacao', InstallationLocation);
app.use('/causa', Cause);
```

O roteamento para os endpoints específicos acontece dentre dos arquivos que ficam na pasta **routes:**

```
/**
 *  ROTA DE VALIDAÇÃO DE LOGIN
 */
router.post('/', createAccountLimiter, async (req, res, next) => {
  try {
    const response = await new LoginValidate().run(req);

    next();
    res.status(200).send(response);
  } catch (err) {
    const responseError = errorResponseTreatment(err);

    res.status(responseError.status).send(responseError);
  }
});

/**
 *  ROTA DE REGISTRO DE USUÁRIO
 */
router.post('/register', async (req, res, next) => {
  try {
    const response = await new RegisterUpdateUser().run(req);

    next();
    res.status(200).send(response);
  } catch (err) {
    const responseError = errorResponseTreatment(err);

    res.status(responseError.status).send(responseError);
  }
});
```

O próximo ponto que o sistema leva são os arquivos na pasta session que são responsáveis pela lógica, regra de negócio e processamento dos dados:

```
async run(req, type = '') {
  try {
    const parameters = this.getParameters(req);

    const errors = this.checkParameters(parameters, type);
    if (Object.values(errors).length > 0) throw errors;

    await this.validateGroups(parameters);

    const user = await this.getPasswordHash(parameters);
    await this.registerUpdateUser(user, type);

    if (!this._queryResult.affectedRows)
      throw type ? 'Nenhum registro foi alterado' : 'Nenhum registro foi inserido';

    return this._queryResult;
  } catch (err) {
    console.log('err registerUser :>> ', err);

    throw err;
  }
}

async registerUpdateUser(parameters, type = '') {
  if (type === 'update')
    this._queryResult = await new userDao(parameters).updateUser();

  else this._queryResult = await new userDao(parameters).registerUser();
}

async getPasswordHash(parameters) {
  const _hash = await generateHash(parameters.senha);

  return {
    numeroCracha: parameters.numeroCracha,
    senha: _hash,
    nome: parameters.nome,
    funcao: parameters.funcao,
    email: parameters.email,
    nivelAcesso: parameters.nivelAcesso,
    updateId: parameters.updateId,
    mysql: parameters.mysql,
```

Por fim, os dados são inseridos no banco de dados pelos arquivos na pasta **dao**:

```
/**
 * registerUser
 * Utiliza as várias globais do construtor para registrar um usuário no banco
 * @returns {Object} objecto parsed contendo informações da execução
 */
async registerUser() {      You, 6 months ago • implemented privite methods for
  const values = {
    numeroCracha: this._numeroCracha,
    senha: this._senha,
    nome: this._nome,
    funcao: this._funcao,
    email: this._email,
    nivel_acesso: this._nivelAcesso,
  };
  const [rows] = await this._mysql.query(/* SQL */ `
    INSERT INTO ${TABLE_USUARIO} SET ?
  `, [values]);

  console.log('user registered => ', this._nome);

  return this.parseInsertResponse(rows);
}

/**
 * updateUser
 * Atualiza os dados do usuário no banco
 * @returns {Object} objecto parsed contendo informações da execução
 */
async updateUser() {
  const values = {
    numeroCracha: this._numeroCracha,
    senha: this._senha,
    nome: this._nome,
    funcao: this._funcao,
    email: this._email,
    nivel_acesso: this._nivelAcesso,
  };

  const [rows] = await this._mysql.query(/* SQL */ `
    UPDATE ${TABLE_USUARIO} SET ? WHERE idUsuario = ?;
  `, [values, this._updateId]);
```

Microsserviços

Arquitetura Microsserviços:

A arquitetura de microsserviços, é utilizada para desenvolver uma aplicação como um conjunto de pequenos serviços, cada um funciona em seus próprios processos sem interferir em outros micro serviços. Cada microsserviço é desenvolvido em cima de um conjunto de regras de negócio específicas,e sua implementação é de forma independente.

Estrutura das Pastas:

```
src/                        |
  lib/                      | Contém todas as regras de negócios
    datasource/             | Comunicação e manipulação do banco de dados
    session/                | Classes, validações e regras de negócio do sistema
    integration/            | Integrações entre microsserviços
  v1/                       | Onde todas as requisições chegam e os dados tratados
  index.js                  | Ponto inicial do sistema
```

Fluxo:

Todas as requisições que chegam nos microsserviços passam pelo arquivo index.js que faz o roteamento para os devidos arquivos:

```
const express = require('express');
const cors = require('cors');
const bodyParser = require('body-parser');

const summary = require('./summary/get');
const lastMonth = require('./orderByMonth/get');
const verificationsOrder = require('./verificationsOrder/get');
const verificationsOrderReport = require('./verificationsOrderReport/get');
const verificationsOrderRequester = require('./verificationsOrderRequester/get');

const app = express();

app.use(cors());

app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());

app.get('/analysis/order-summary', async (req, res) => {
  try {
    const response = await summary.run(req);

    res.status(200).send(response);
  } catch (err) {
    res.status(404).send(err);
  }
});
```

Depois do roteamento, as requisições vão para os arquivos que ficam dentro da pasta v1 para que os dados sejam retirados e tratados:

```
const checkParameters = ({
  auth,
} = {}) => ({
  ...(!auth ? { auth: 'undefined' } : ''),
});

const run = async req => {
  try {
    const parameters = getParameters(req);

    const errs = checkParameters(parameters);
    if (Object.keys(errs).length > 0) throw errs;

    const response = await new GetOrderSummary(parameters).run();

    return response;
  } catch (err) {
    console.log('err get => ', err);
    throw err;
  }
};
```

Após os tratamentos dos dados, a requisições segue para a classe que cuida da regra de negócio:

```
module.exports = class GetOrderSummary {
  constructor({
    auth,
  } = {}) {
    this._auth = auth;

    this._integrationAuthJwt = new GetUserAutentication();
    this._getSummaryData = new GetOrderSummaryData();

    this._checkParameters();
  }


  async run() {      You, 8 months ago • Develop dashboard visualiz
    try {
      await this._validateSession();

      const summary = await this._getOrderSummary();

      return summary;
    } catch (err) {
      throw err;
    } finally {
      this._getSummaryData.closeConnection();
    }
  }

  async _getOrderSummary() {
    try {
      const orders = await this._getSummaryData.getSummary();

      if (!orders) throw 'could not find orders';

      return orders;
    } catch (err) {
      throw err;
    }
  }
```

Ao final, é enviado para a classe que cuida da comunicação com o Banco de Dados:

```
You, 21 days ago | 1 author (You)
module.exports = class GetOrderSummaryData {
  constructor() {
    this.mysql = '';

    this.createConnection();
  }

  async getSummary() {      You, 8 months ago • Develop dashboard visualization and new analysis microservice
    return new Promise((resolve, reject) => {
      this.mysql.query(/* sql */ `
        SELECT
          (SELECT count(*) FROM ${ORDERS_TABLE} WHERE Status_idStatus = 1 AND ${ORDERS_TABLE}.excluded = 0) as openOrders,
          (SELECT count(*) FROM ${ORDERS_TABLE} WHERE Status_idStatus = 2 AND ${ORDERS_TABLE}.excluded = 0) as currentOrders,
          (SELECT count(*) FROM ${ORDERS_TABLE} WHERE Status_idStatus = 3 AND ${ORDERS_TABLE}.excluded = 0) as finishOrders,
          (SELECT count(*) FROM ${ORDERS_TABLE} WHERE Status_idStatus = 4 AND ${ORDERS_TABLE}.excluded = 0) as canceledOrders
        FROM ${ORDERS_TABLE}
        GROUP BY openOrders
      `, (err, res) => {
        if (err) return reject(err);
        return resolve(res[0]);
      });
    });
  }

  async createConnection() {
    try {
      const connection = mysql.createConnection({
        host: 'duasrodasdb.cjh4gc3id4wo.sa-east-1.rds.amazonaws.com',
        user: 'adminDuasRodas',
        password: 'twowheels2020',
        database: 'duasrodas',
      });

      connection.connect(err => {
        if (err) throw err;
      });
```

Endpoints

Ordem Manutenção

POST Register Ordem Corretiva

```
http://localhost:3000/ordem-manutencao
```

HEADER

Authorization	Bearer [Code]
type	summary

BODY:

```
{
	"title": "BRT54234",
	"summary": "Máquina estragou",
	"description": "",
	"plannedStart": "2020-08-20",
	"plannedEnd": "2020-08-30",
	"beginData": "2020-08-02",
	"requireStop": "true",
	"equipment": "4",
	"requester": "3",
	"report": "1",
	"typeMaintenance": 1,
	"sector": "2",
	"priority": "2",
	"stats": 1,
	"plannedTime": "",
	"operations": [{
		"Operacao": 1,
		"sequencia_operacao": "0010"
	}, {
		"Operacao": 2,
		"sequencia_operacao": "0020"
	}, {
		"Operacao": 3,
		"sequencia_operacao": "0030"
	}],
	"epis": [{
		"Epi_idEpi": 1
	}, {
```

```
            "Epi_idEpi": 2
        }, {
            "Epi_idEpi": 3
        }, {
            "Epi_idEpi": 4
        }]
}
```

GET Get Ordens

```
http://localhost:3000/ordem-manutencao
```

HEADER

Authorization	Bearer [Code]
order	idOrder

DELETE Delete Ordem de Manutenção

```
http://localhost:3000/ordem-manutencao/:id
```

HEADER

Authorization	Bearer [Code]

GET Get Operações da Ordem

```
http://localhost:3000/ordem-manutencao/equipments-operations
```

HEADER

Authorization	Bearer [Code]
order	idOrder
order_type	type

POST Post Check Operação

```
http://localhost:3000/operacoes/check
```

HEADER

Authorization	Bearer [Code]
order	idOrder

BODY:

```
{
    "order": "",
    "equipment": "",
    "operation": ""
}
```

Causas

GET Get Causas

```
http://localhost:3000/causa
```

HEADER

Authorization	Bearer [Code]

POST Register Causa

```
http://localhost:3000/causa
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "descricaoCausa": ""
}
```

PUT Register Causa

```
http://localhost:3000/causa/:id
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "descricaoCausa": ""
}
```

DELETE Delete Causa

```
http://localhost:3000/causa/:id
```

HEADER

Authorization	Bearer [Code]

Usuario

POST Login

```
http://localhost:3000/users
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "numeroCracha": "",
    "senha": ""
}
```

GET Get Users

```
http://localhost:3000/users
```

HEADER

Authorization	Bearer [Code]
type	type
order_id	[orderId]

POST Register User

```
http://localhost:3000/users/register
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "numeroCracha": "",
    "senha": "",
    "nome": "",
    "email": "",
    "funcao": "",
    "nivel_acesso": 1
}
```

PUT Update User

```
http://localhost:3000/users/:id
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "numeroCracha": "",
    "senha": "",
    "nome": "",
    "email": "",
    "funcao": "",
    "nivel_acesso": 1
}
```

DELETE Delete User

```
http://localhost:3000/users/:id
```

HEADER

Authorization	Bearer [Code]

EPI

GET Get EPI

```
http://localhost:3000/epi
```

HEADER

Authorization	Bearer [Code]

POST Register EPI

```
http://localhost:3000/epi
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "descricaoEpi": ""
}
```

PUT Update EPI

```
http://localhost:3000/epi/:id
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "descricaoEpi": ""
}
```

DELETE Update EPI

```
http://localhost:3000/epi/:id
```

HEADER

Authorization	Bearer [Code]

Tipo de Ordem

GET Get Tipo de Ordens

```
http://localhost:3000/tipo-ordem
```

HEADER

Authorization	Bearer [Code]

Prioridade

GET Get Prioridades

```
http://localhost:3000/prioridade
```

HEADER

Authorization	Bearer [Code]

Centro de Trabalho

GET Get Centro de Trabalho

```
http://localhost:3000/centro-trabalho
```

HEADER

Authorization	Bearer [Code]

POST Register Centro de Trabalho

```
http://localhost:3000/centro-trabalho
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "descricao": ""
}
```

PUT Update Centro de Trabalho

```
http://localhost:3000/centro-trabalho/:id
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "descricao": ""
}
```

DELETE Delete Centro de Trabalho

```
http://localhost:3000/centro-trabalho/:id
```

HEADER

Authorization	Bearer [Code]

Operações

POST Register Operação

```
http://localhost:3000/operacoes
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "descricao_operacao": "",
    "material": "",
    "quantidade_material": 10,
    "unidade_material": "",
    "tempo_planejado": ""
}
```

PUT Update Operação

```
http://localhost:3000/operacoes/:id
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "descricao_operacao": "",
    "material": "",
    "quantidade_material": 10,
    "unidade_material": "",
    "tempo_planejado": ""
}
```

DELETE Delete Operação

```
http://localhost:3000/operacoes/:id
```

HEADER

Authorization	Bearer [Code]

GET Get Operação

```
http://localhost:3000/operacoes
```

HEADER

Authorization	Bearer [Code]

Componente

GET Get Componente

```
http://localhost:3000/componente
```

HEADER

Authorization	Bearer [Code]

POST Post Componente

```
http://localhost:3000/componente
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "DescricaoComponente": "",
    "Equipamento_idEquipamento": 1
}
```

PUT Put Componente

```
http://localhost:3000/componente/:id
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "DescricaoComponente": "",
    "Equipamento_idEquipamento": 1
}
```

DELETE Delete Componente

```
http://localhost:3000/componente/:id
```

HEADER

Authorization	Bearer [Code]

Movimentações de Etapas

POST Assumir

```
http://localhost:3000/movimentacao-etapa/assumir
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "email": "",
    "nome": "",
    "nivelAcesso": 1,
    "funcao": "",
    "cracha": "",
    "userId": 2,
    "order": 3037
}
```

Operações dos Equipamentos

POST Check de Operação

```
http://localhost:3000/operacoes/check
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "operation": "",
    "order": 3037
}
```

Equipamentos

GET Get Equipamentos

```
http://localhost:3000/equipamento
```

HEADER

Authorization	Bearer [Code]

POST Post Equipamento

```
http://localhost:3000/equipamento
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "Setor_idSetor": 1,
    "equipamento": "",
    "equipamentoSuperior": 1,
    "descricao": ""
}
```

PUT Put Equipamento

```
http://localhost:3000/equipamento/:id
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "DescricaoComponente": "",
    "Equipamento_idEquipamento": 1
}
```

DELETE Delete Equipamento

```
http://localhost:3000/equipamento/:id
```

HEADER

Authorization	Bearer [Code]

Local Instalação

GET Get Local Instalação

```
http://localhost:3000/local-instalacao
```

HEADER

Authorization	Bearer [Code]

POST Post Local Instalação

```
http://localhost:3000/local-instalacao
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "nome": ""
}
```

PUT Put Local Instalação

```
http://localhost:3000/local-instalacao/:id
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "nome": ""
}
```

DELETE Delete Local Instalação

```
http://localhost:3000/local-instalacao/:id
```

HEADER

Authorization	Bearer [Code]

Nível Acesso

GET Get Nivel Acesso

```
http://localhost:3000/local-instalacao
```

HEADER

Authorization	Bearer [Code]

Sintomas

GET Get Sintomas

```
http://localhost:3000/sintoma
```

HEADER

Authorization	Bearer [Code]

POST Post Sintomas

```
http://localhost:3000/sintoma
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "descricaoSintomas": "TEsteeee"
}
```

PUT Put Sintomas

```
http://localhost:3000/sintoma/:id
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "descricaoSintomas": "TEsteeee"
}
```

DELETE Delete Sintomas

```
http://localhost:3000/sintoma/:id
```

HEADER

Authorization	Bearer [Code]

Status

GET Get Status

```
http://localhost:3000/status
```

HEADER

Authorization	Bearer [Code]

Verificação

POST Post Verificar Ordem

```
http://localhost:3000/verificacao
```

HEADER

Authorization	Bearer [Code]
order	[orderId]

BODY:

```
{
    "solutionDescription": "tessssteeee",
    "resolved": "1",
    "dateVerification": "2020-04-22 18:30:53.0",
    "order": 3048,
    "typeVerification": 1,
    "cracha": "12035"
}
```

Delegar

POST Post Delegar

```
http://localhost:3000/delegar-manutentor
```

HEADER

Authorization	Bearer [Code]
order	[orderId]

BODY:

```
{
    "orderId": "3048",
    "numeroCracha": "10192",
    "nome": "Douglas"
}
```

Apontamentos

POST Post Apontamentos

```
http://localhost:3000/order-note
```

HEADER

Authorization	Bearer [Code]

BODY:

```
{
    "user": "",
    "order": "",
    "date": "",
    "time": "",
    "description": ""
}
```

Tecnologias Utilizadas

- JWT

O JWT(JSON Web Token) é um módulo RCT 7519 padrão da indústria para realizar autenticação entre duas partes por meio de um token assinado que autentica uma requisição web. Este token é um código gerado em Base64 que armazena objetos JSON com os dados que permite a autenticação de requisições HTTP/HTTPS.Após o token ser gerado o usuário tem seu acesso liberado enquanto aquele token estiver ativo na API servidor.

Link para a documentação:
https://jwt.io/

- Axios

Axios é um cliente HTTP baseado em Promises para fazer requisições. Pode ser utilizado tanto no navegador quanto no Node.Ele é utilizado para realizar a comunicação entre o front end e o backend.

Link para a documentação:
https://www.npmjs.com/package/axios

- Vuex

O Vuex é uma biblioteca que serve como um armazenamento centralizado para todos os componentes em um aplicativo, com regras garantindo que o estado só possa ser modificado de maneira previsível. Esta tecnologia apresenta a capacidade de fornecer um meio comum para que diversas partes do sistema, possam acessar o mesmo estado.

Link para a documentação:
https://vuex.vuejs.org/

- Bootstrap-vue

O bootstrap-vue é uma versão customizada de bootstrap que foi adaptado para framework vue.js. Bootstrap Vue contém mais de 85 componentes, mais de 45 plug-ins disponíveis, diretivas, temas e mais de 1100 ícones. É uma biblioteca que permite construir projetos responsivos, utilizando as melhores partes do maior framework CSS disponível no mercado, bootstrap.

Link para a documentação:
https://bootstrap-vue.org/

- Vue-chartjs

Vue-chartjs é um pacote para Chart. js no para Vue.js. Com ele é possível criar facilmente componentes de gráficos reutilizáveis, podendo assim ser montado diversos relatórios dentro do sistema com esta tecnologia.

Link para a documentação:
https://vue-chartjs.org/

- Vue-form-wizard

É uma tecnologia voltada para Vue.js, que tem como intuito, organizar um modelo de step by step (passo a passo), sendo assim possível separar em diversas etapas um fluxo a ser completo no sistema.

Link para a documentação:
https://binarcode.github.io/vue-form-wizard/#/

- Momentjs

É uma biblioteca Javascript gratuita que permite converter, manipular, validar e apresentar datas e horas em javascript.

Link para a documentação:
https://momentjs.com/

- Fontawesome

O fontawesome se trata de um conjunto de ícones baseado em webfont e CSS. Além disso seus ícones vetoriais escaláveis que podem ser personalizados instantaneamente como em, tamanho, cor, sombra e outras características compatíveis com CSS.

Link para a documentação:
https://fontawesome.com/

- Vue-tables-2

O Vue table 2 é um componente de tabela de dados reutilizáveis, que converte os dados brutos(JSON) em formato de tabela HTML com facilidade e flexibilidade. Permite definir quais campos serão usados e manipulados antes de exibir, além de configurar progressivamente a forma de exibir os dados de acordo com seu projeto.

Link para a documentação:
https://www.vuetable.com/

- Vue-sweetalert2

SweetAlert 2 é uma biblioteca de notificações e alertas que é customizável, responsivo, e acessível . O Vue-sweet alert 2 é um wrapper para SweetAlert 2 que facilita a integração dos recursos com suas aplicações vue.js.

Link para a documentação:
https://sweetalert2.github.io/

- Dotenv

O dotenv é uma ferramenta utilizada para orquestrar as variáveis ambientes de um projeto. O nome dela sugere o arquivo em que as informações ficarão, *dot* que é ponto em inglês acrescido de *env*, então temos o arquivo *.env* que é composto de chaves e valores.

Link para a documentação:
https://www.npmjs.com/package/dotenv

- Bcryptjs

O BCrypt foi desenvolvido com a finalidade de esconder senhas criadas pelos usuários em forma de texto em dados indecifráveis, utilizando um algoritmo hash.

Link para a documentação:
https://www.npmjs.com/package/bcrypt

- Express

O Express é um framework de web para Node.js, ele fornece um conjunto robusto de recursos para aplicativos web e mobile. O Express fornece uma camada fina de recursos fundamentais para aplicativos web, sem obscurecer os recursos do Node.js.

Link para a documentação:
https://expressjs.com/pt-br/

- Eslint

O ESLint analisa estaticamente o código para encontrar problemas rapidamente. ESLint é integrado à maioria dos editores de texto. Eslint também permite configurar as opções das regras e padrões de desenvolvimento para que a equipe se adeque a forma da empresa trabalhar.

Link para a documentação:
https://eslint.org/

- ## Mysql2

  O MySQL2 é uma biblioteca que tem a capacidade de trabalhar com o mysql, sendo compatível com a API do mysql. Esta tecnologia é responsável por realizar a conexão com o banco de dados MySQL.

  Link para a documentação:
  https://www.npmjs.com/package/mysql2

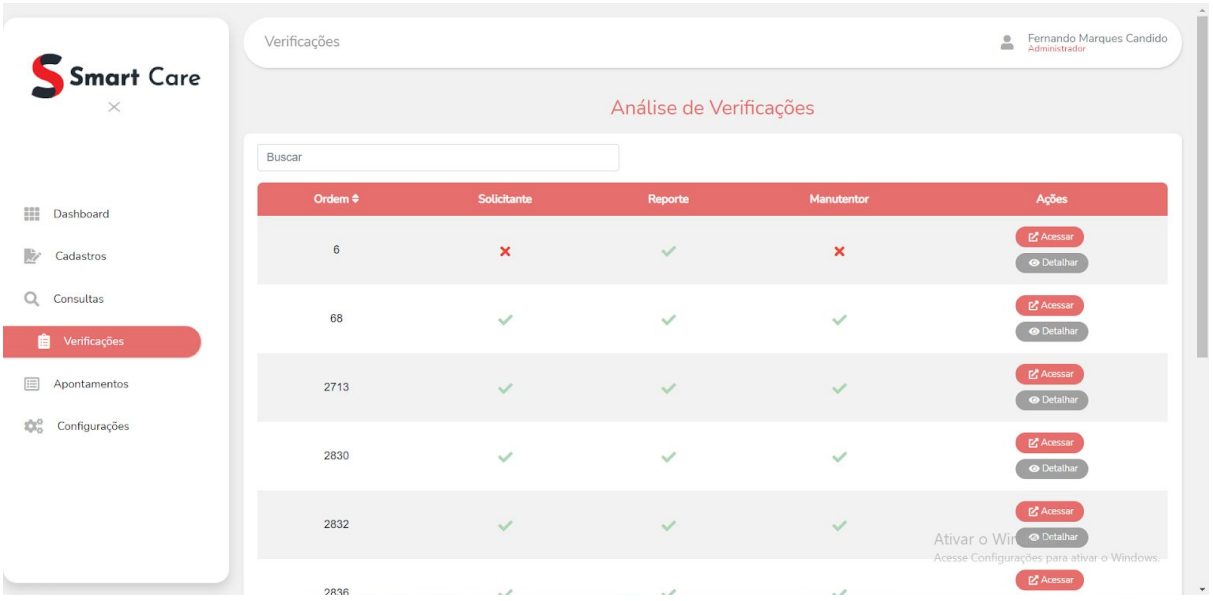
- ## Express-rate-limit

  O Express Rate Limit permite limitar solicitações repetidas a APIs e / ou endpoints públicos, como redefinição de senha.

  Link para a documentação:
  https://www.npmjs.com/package/express-rate-limit

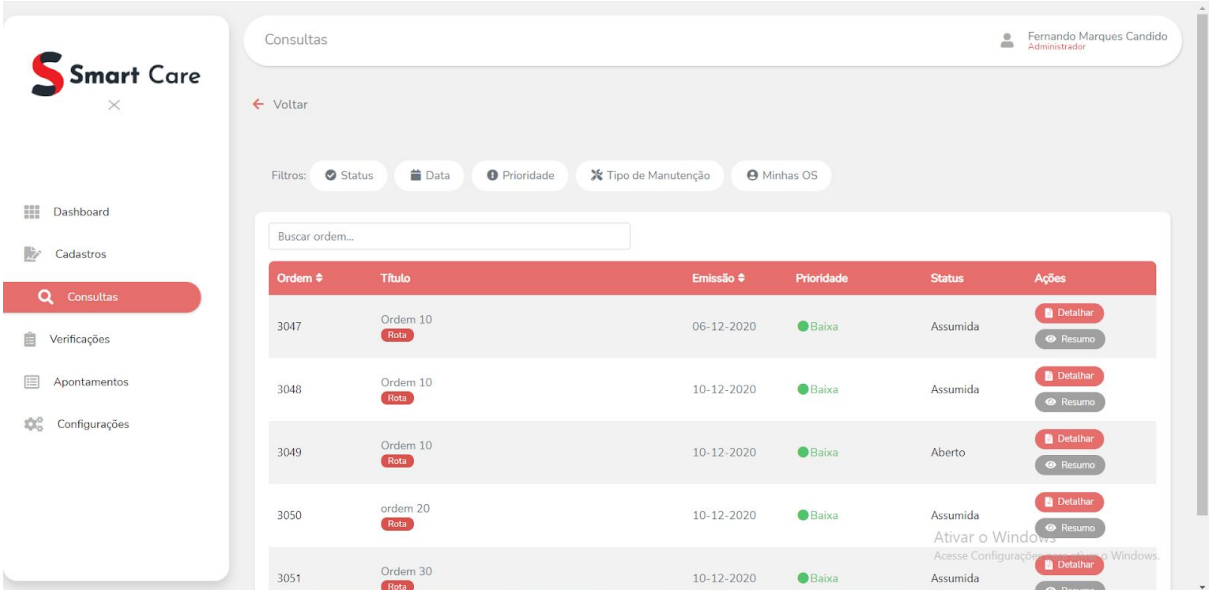
Sistema Implementado

Tela de apontamento das verificações:



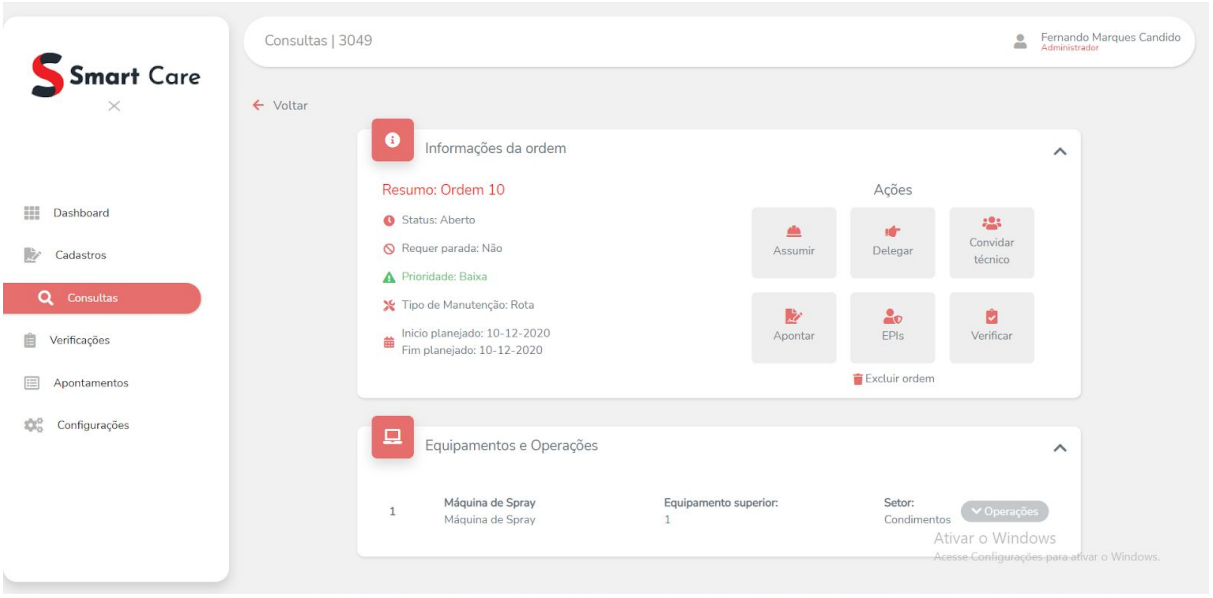
A tela de apontamento das verificações possibilita verificar todas as verificações realizadas em uma ordem de serviço. Ao clicar na opção detalhar o usuário terá na íntegra os dados de quem assinou, se o problema foi resolvido, data da assinatura e as observações, mas também se não estiver com todos as verificações seja pela ordem tar em aberto ou em andamento , será exibido a mensagem “Não há verificações registradas”. Outra funcionalidade tanto em uma opção dentro do modal de detalhar quanto na opção abaixo dela é a opção acessar aquela ordem para ter uma visualização mais completa sobre a ordem.

Tela de consultas de ordens de manutenção:



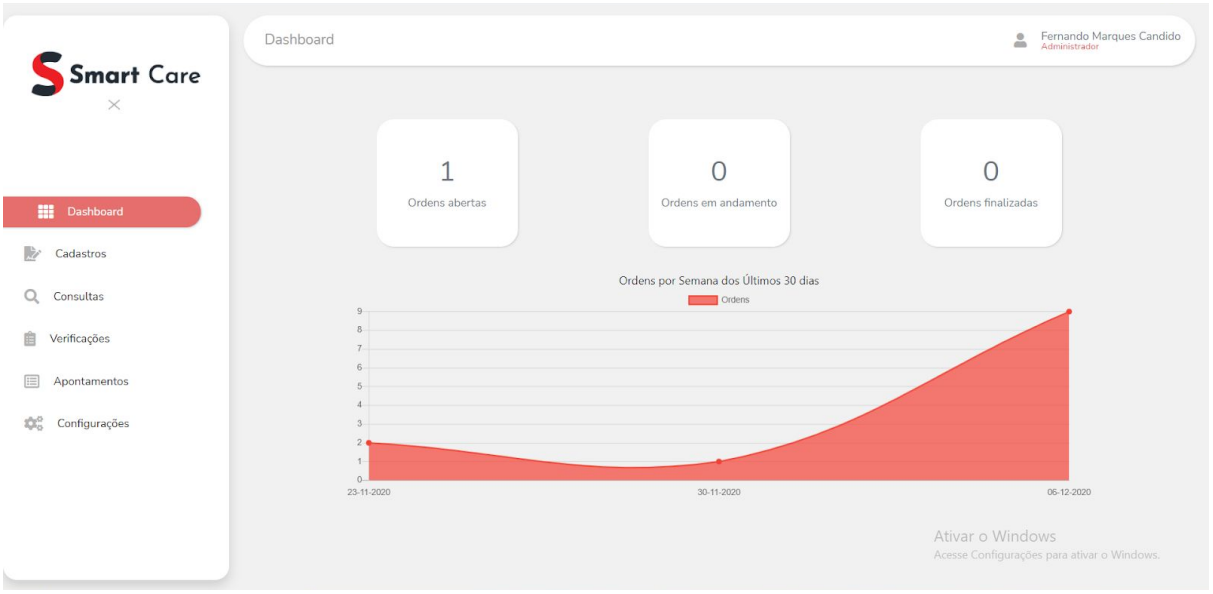
Nesta tela de consulta é possível consultar as ordens de manutenção que estão abertas ou assumidas. Na tela de consultas também tem disponíveis filtros de pesquisas, a fim de facilitar ao usuário encontrar a ordem que deseja , seja a que ele foi delegado, ou assume ou mesmo procurar uma para assumir. Outra opção desta tela é a possibilidade de visualizar um resumo rápido das informações daquela ordem, este resumo por sua vez contém há quanto tempo que ela está aberta, número da ordem, tipo, descrição, prioridade e status. Na opção acima de resumo temos o link de acesso para as informações completas da ordem, bem como as ações que se pode efetuar na ordem como mostrado na imagem a seguir.

Tela com as opções de uma ordem



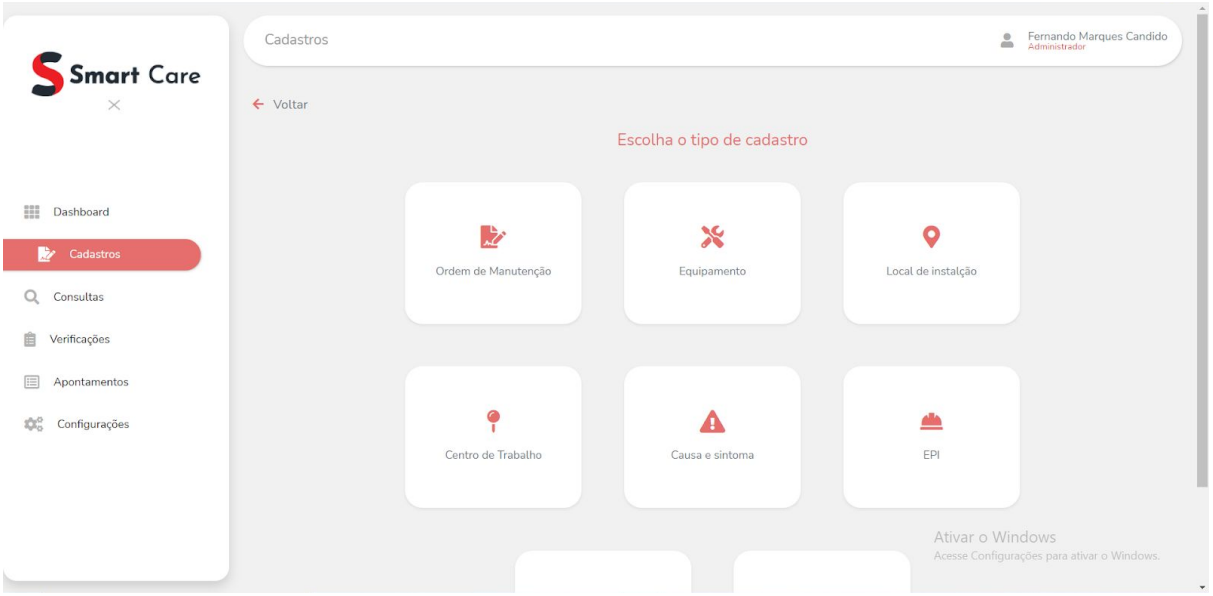
Ao acessar a opção de detalhar na tela anterior, você poderá visualizar as ações disponíveis na ordem de serviço podendo assumir/iniciar, delegar, convidar técnico, apontar, epi's, verificar e excluir a ordem caso tenha nível de administrador no sistema. Além das ações você poderá visualizar as informações na íntegra daquela ordem, bem como os equipamentos que necessitarão de alguma intervenção.

Tela de dashboard



A tela de dashboard é visualizada após ser efetuado o login no sistema. Esta contém detalhes básicos como ordens abertas, finalizadas, em andamento bem como um gráfico com a quantidade de ordens nos últimos 30 dias divididas em 3 datas logo abaixo do gráfico.

Tela com todas as opções de cadastros



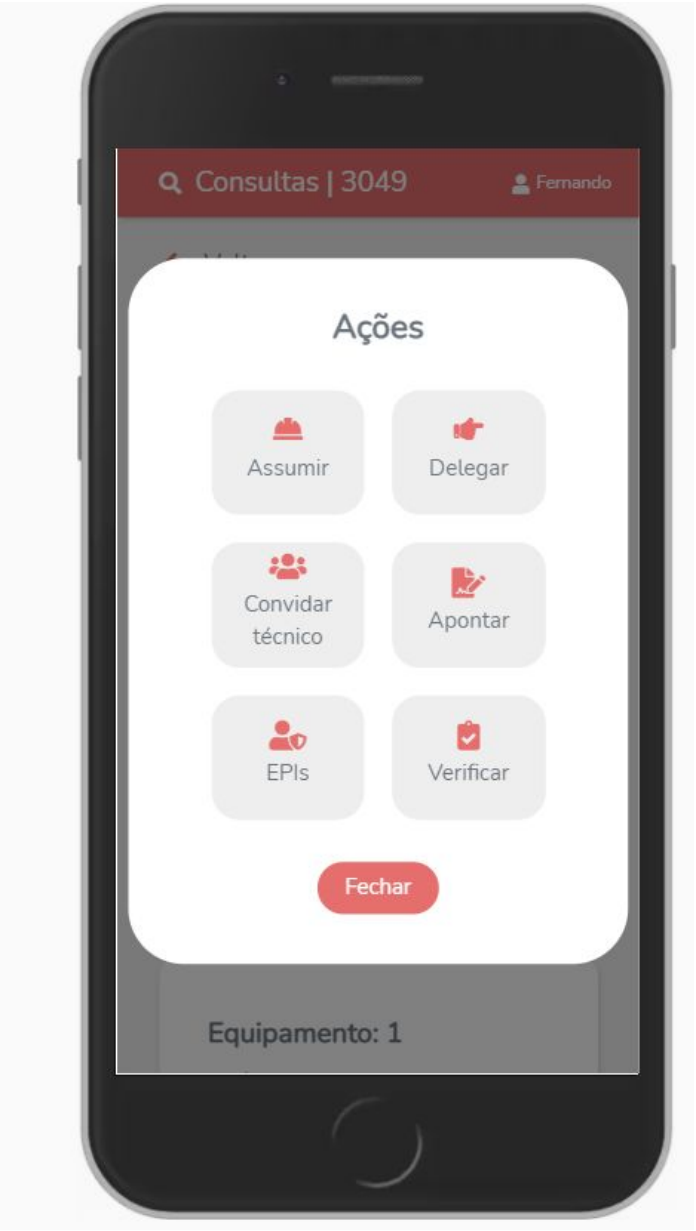
A Opção de cadastros em nosso menu abrange quase todos os tipos de cadastros disponíveis no sistema. Estão disponíveis as opções para cadastrar: os quatro tipos de ordem: serviço, equipamento, local de instalação, centro de trabalho,

causa/sintoma, EPI, componentes e operações. Cada uma das opções de cadastro contam com listar, cadastrar, editar e excluir.

Tela mobile com as opções da ordem

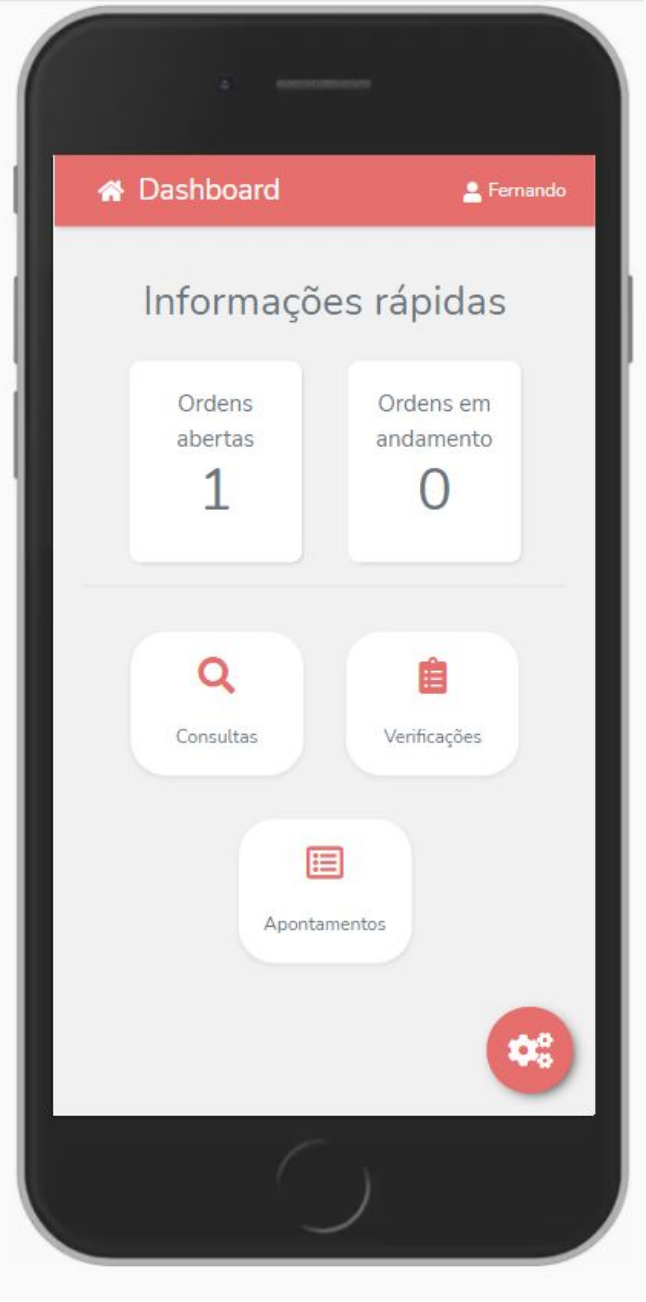
A tela de consulta da ordem no mobile se divide em duas partes sendo esta primeira correspondente a toda as informações pertinentes aquela ordem escolhida, desde os dados básicos de uma manutenção até os equipamentos e procedimentos que deve-se aplicar naquela máquina específica.

Tela das ações versão mobile



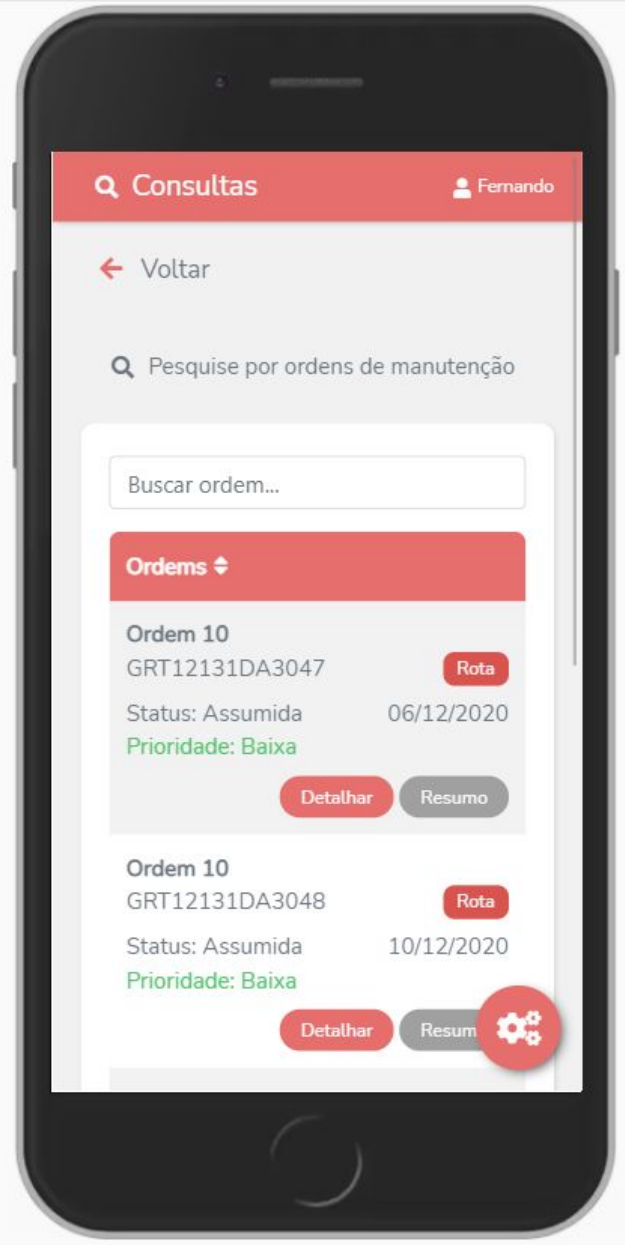
Esta é a segunda parte da tela anterior que ao clicar no sinal de **"+"**o usuário terá as opções disponíveis para cada ordem no sistema, desde a parte de assumir uma ordem até a parte de efetuar os apontamentos da mesma.

Tela de dashboard da versão mobile



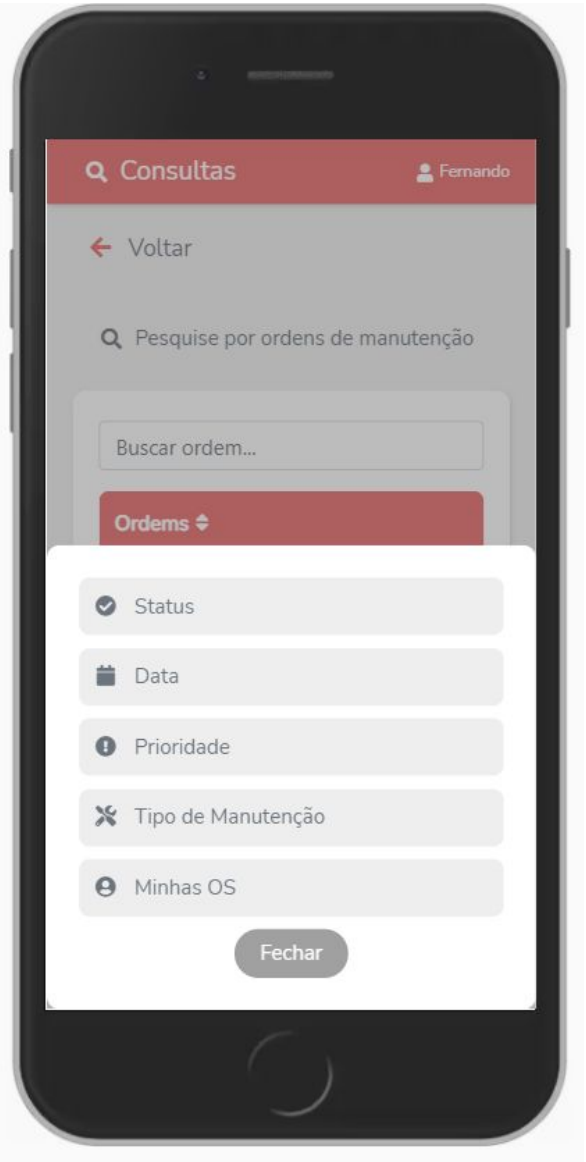
A versão mobile da dashboard disponibiliza informações rápidas como ordens abertas, ordens em andamento, mas também tem botões logo abaixo que permitem ao usuário ter acesso às opções de navegação no sistema, seja apontamentos, verificações e consultas de ordens.

Tela de consultas versão mobile



Tela de consulta da versão mobile é semelhante a versão web pois a aplicação foi construída para ser um web app, ou seja, ela se adapta a tela que está sendo utilizada no momento que é executada. Mas os filtros foram remanejados para o ícone no canto inferior direito com o símbolo de 3 engrenagens dentro de um botão circular vermelho.

Filtros da tela de consulta mobile



Esta tela disponibiliza todas as opções de filtros disponíveis para o usuário do sistema para encontrar o tipo, prioridade, data, status e suas próprias ordens de serviço. Desta forma se torna mais prático procurar no sistema uma ordem específica ou mesmo conferir detalhes. Esta tela é acessível após ser clicado no botão localizado na imagem anterior.