DEVinHouse

Módulo 1 - Projeto Avaliativo 2

SUMÁRIO

1 MOTIVAÇÃO

A **DEVinThings LTDA**, empresa líder no segmento tecnológico de gestão de equipamentos para uso em *home-office*, está com um projeto novo intitulado **DEVinventary**, um software audacioso para gestão de inventário de empresas.
O seu perfil chamou a atenção dos gestores, para criar a aplicação FrontEnd do software, que deverá ser construída utilizando o *framework* **VueJs**.

2 REQUISITOS DA APLICAÇÃO

A aplicação que deverá ser realizada **individualmente**, deve contemplar os seguintes requisitos:

1. Ser construída utilizando o *framework* VueJs:
    a. Utilizar o Vue Router para gerenciamento das rotas.
    b. Utilizar o Vuex para gerenciamento de estado.
    c. Utilizar o Vue Gravatar para gerenciar as fotos de perfil.
    d. Utilizar animações de loading e transições entre páginas.
    e. Utilizar consumo da API ViaCEP para cadastro de endereço.
2. Utilizar favicon, título de página e demais assets.
3. Utilizar o localStorage para guardar as informações cadastradas.
4. Utilizar o GitHub como versionador de código:

a. Utilização do padrão baseado em GitFlow com branches master/main, features e releases.
b. Utilização de commits curtos e concisos.

5. Possuir as seguintes páginas e funcionalidades:
    a. **Página de Login**: Contendo um formulário com campos de email e senha, botões para criar uma conta, entrar na aplicação, entrar com Google, e um link para resetar a senha.
        i. O botão de criar conta deve abrir uma página ou modal com formulário para criar um usuário, sendo exigido e-mail válido e senha com mais de 8 caracteres. Para a senha utilize dois campos e valide se a mesma é equivalente. Salve o usuário no localStorage.
        ii. As funcionalidades de login com Google e esqueceu a senha não precisam ser implementadas. Caso não sejam implementadas, utilize um *pop-up*, *toast* ou *alert* para avisar que esta funcionalidade está em construção.
        iii. O botão de entrar deve direcionar o usuário para a tela de Inventário, caso o e-mail seja válido, a senha maior que 8 caracteres, e o usuário esteja cadastrado.
    b. **Menu Lateral:** Com exceção da tela de login, deverá estar presente em todas as outras páginas, apresentando todos os botões de navegação. O botão de sair deve 'deslogar' e direcionar o usuário para a página de login. Deverá possuir um botão para esconder e mostrar o menu lateral.
    c. **Toolbar:** Com exceção da tela de login, deverá estar presente em todas as outras páginas, apresentando o título da página, nome do usuário e foto do usuário. Para a foto do usuário utilize o Vue Gravatar. Opcionalmente poderá ser adicionado outras funcionalidades como dropdown e botão de sair, por exemplo.
    d. **Página de Inventário**: Conter as estatísticas do sistema e busca de produtos/itens.
        i. As estatísticas devem ser ao menos a quantidade de colaboradores cadastrados, quantidade de itens cadastrados, valor total dos itens e quantidade de empréstimos.
        ii. A busca de itens deve mostrar um card para cada item com ao menos uma imagem (usar cadastro de url), breve descrição, marca, modelo e se está emprestado.
        iii. Deverá existir um campo para buscar itens pela descrição, marca, modelo ou modelo.
        iv. Ao clicar no card/item, uma nova página ou modal deve ser aberta mostrando todas as informações cadastradas, e a opção de editar. Dica:

Você pode utilizar a mesma página de cadastro, carregando as informações para os inputs, assim será possível mostrar os detalhes e editar.

e. **Página de Cadastro de Colaborador:** Deve conter um formulário para cadastro de colaborador com botões para salvar e limpar.
    i. Campos do formulário: nome completo, gênero (usar dropdown), data de nascimento, telefone, e-mail, cargo (usar dropdown), cep, cidade, estado, logradouro, número, complemento, bairro e ponto de referência.
    ii. Deverá verificar os dados informados antes de cadastrar.
    iii. Deverá utilizar a API do ViaCEP para buscar os dados de endereço.
    iv. Deverá apresentar animação ao salvar.
    v. Deverá limpar o formulário ao clicar no botão limpar.
f. **Página de Listagem de Colaboradores:** Deverá possuir uma barra de pesquisa, listagem de colaboradores cadastrados e detalhes ao clicar no colaborador.
    i. A barra de busca deverá buscar o colaborador pelo nome cadastrado.
    ii. Ao clicar no item/card do colaborador, uma nova página ou modal deve ser aberta mostrando todas as informações cadastradas, e a opção de editar. Dica: Você pode utilizar a mesma página de cadastro, carregando as informações para os inputs, assim será possível mostrar os detalhes e editar.
    iii. A imagem do colaborador deverá ser um avatar do vue-gravatar.
g. **Página de Cadastro de Item:** Deve conter um formulário para cadastro de item com botões para salvar e limpar.
    i. Campos do formulário: código de patrimônio, título do item, categoria do item (usar dropdown), valor do item, url da foto, marca, modelo e descrição.
    ii. Deverá verificar os dados informados antes de cadastrar.
    iii. Deverá apresentar animação ao salvar.
    iv. Deverá limpar o formulário ao clicar no botão limpar.
h. **Página de Empréstimo de Itens:** Deverá possuir uma barra de pesquisa, listagem de todos os itens cadastrados e detalhes ao clicar no item.
    i. A barra de busca deverá buscar o item pelo número de patrimônio.
    ii. Os itens a serem mostrados devem possuir os campos patrimônio, título, categoria e para quem foi emprestado.
    iii. Ao clicar no item/card, uma nova página ou modal deve ser aberta mostrando todas as informações cadastradas, e a opção de editar. Dica: Você pode utilizar a mesma página de cadastro, carregando as

informações para os inputs, assim será possível mostrar os detalhes e editar.

iv. Cada card/item deve possuir um *dropdown* ou outro tipo de elemento que permita mostrar todos os colaboradores cadastrados, e ao selecionar o mesmo, deverá vincular o empréstimo. Lembrando que um único colaborador pode pegar mais de um item emprestado.

3 EXEMPLO DE APLICAÇÃO

A aplicação deverá conter os requisitos apresentados anteriormente, sendo codificada em html, css e javascript através do *framwork* VueJs, também deverá ser utilizado markdown para o readme.md.
As imagens a seguir demonstram exemplos da aplicação que deverá ser desenvolvida. Você poderá desenvolver igual ao modelo, ou criar um layout diferente com estilo próprio.

Visualização do protótipo: DEVinventary Project

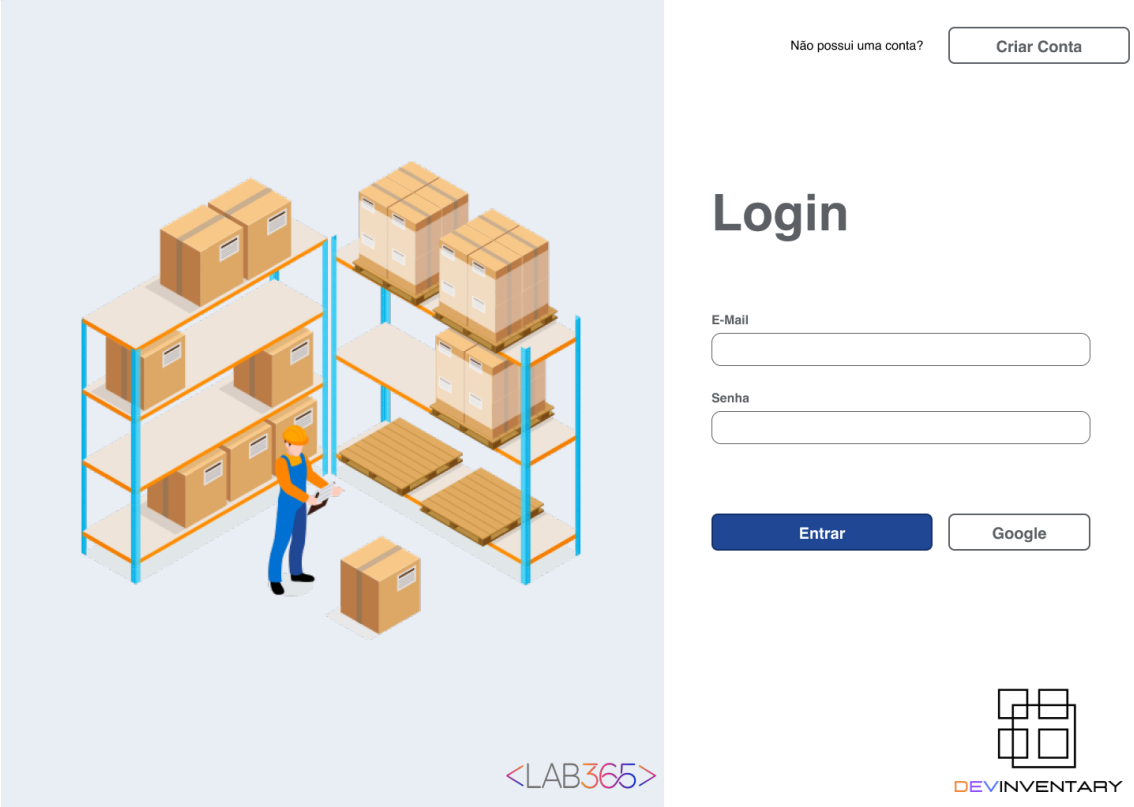

Imagem 1: Página de Login

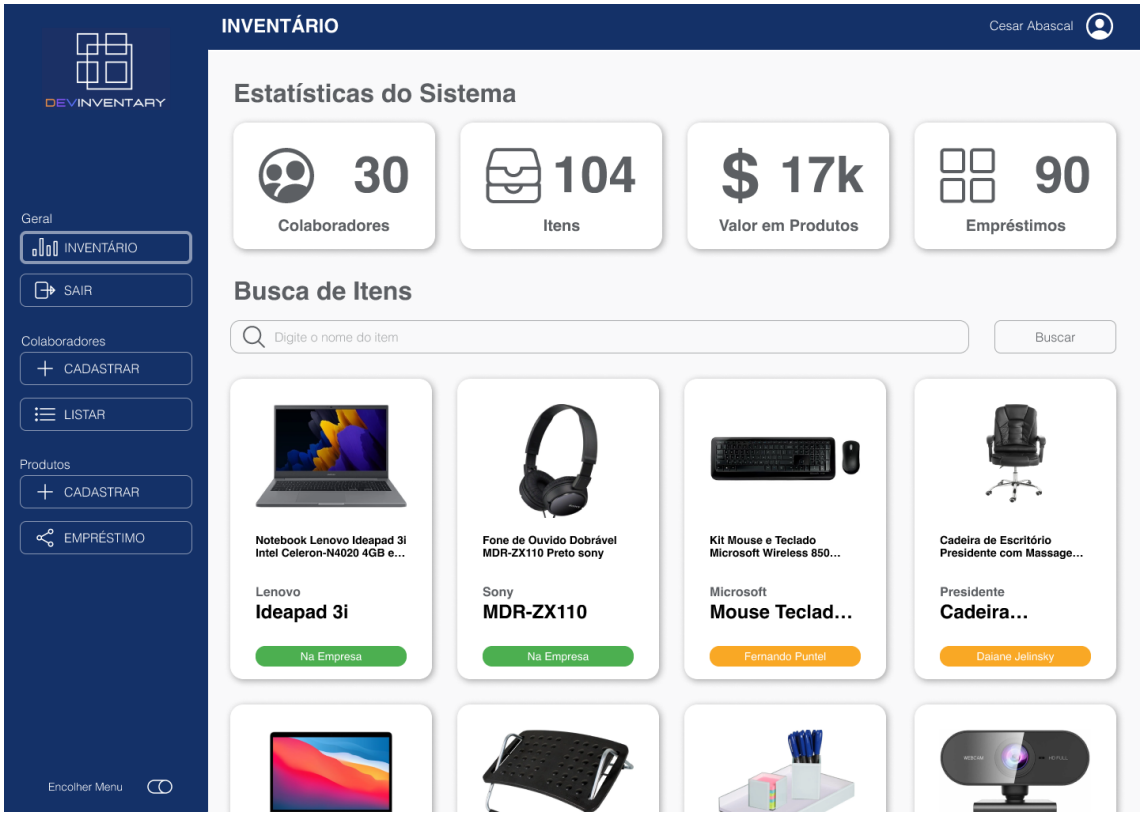

Imagem 2: Página de Inventário

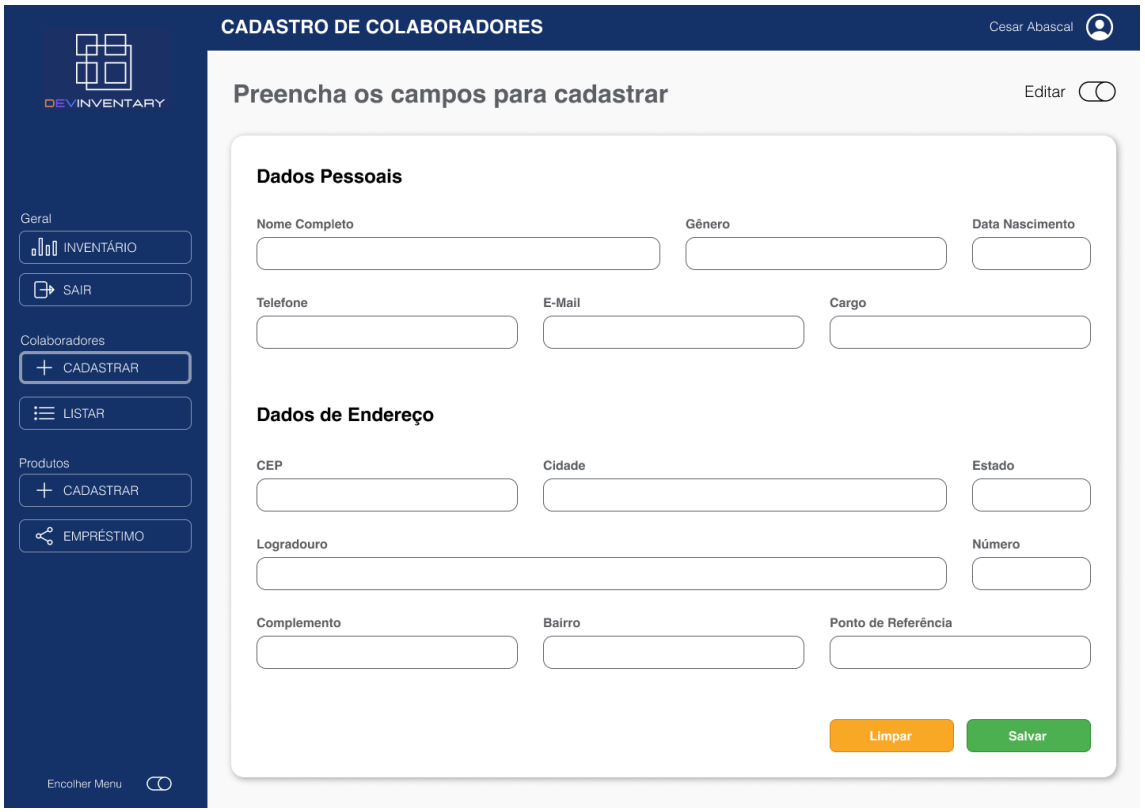

Imagem 3: Página de Cadastro de Colaborador

Imagem 4: Página de Listagem de Colaboradores

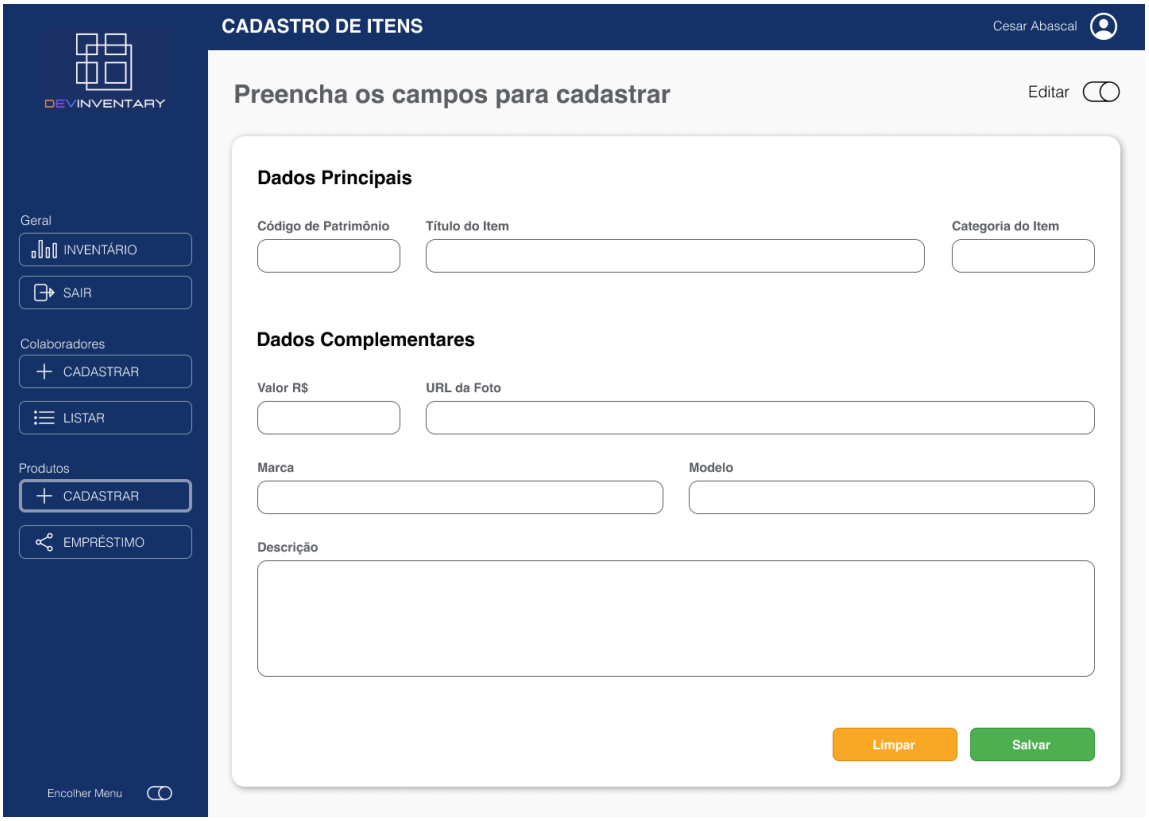

Imagem 5: Página de Cadastro de Item

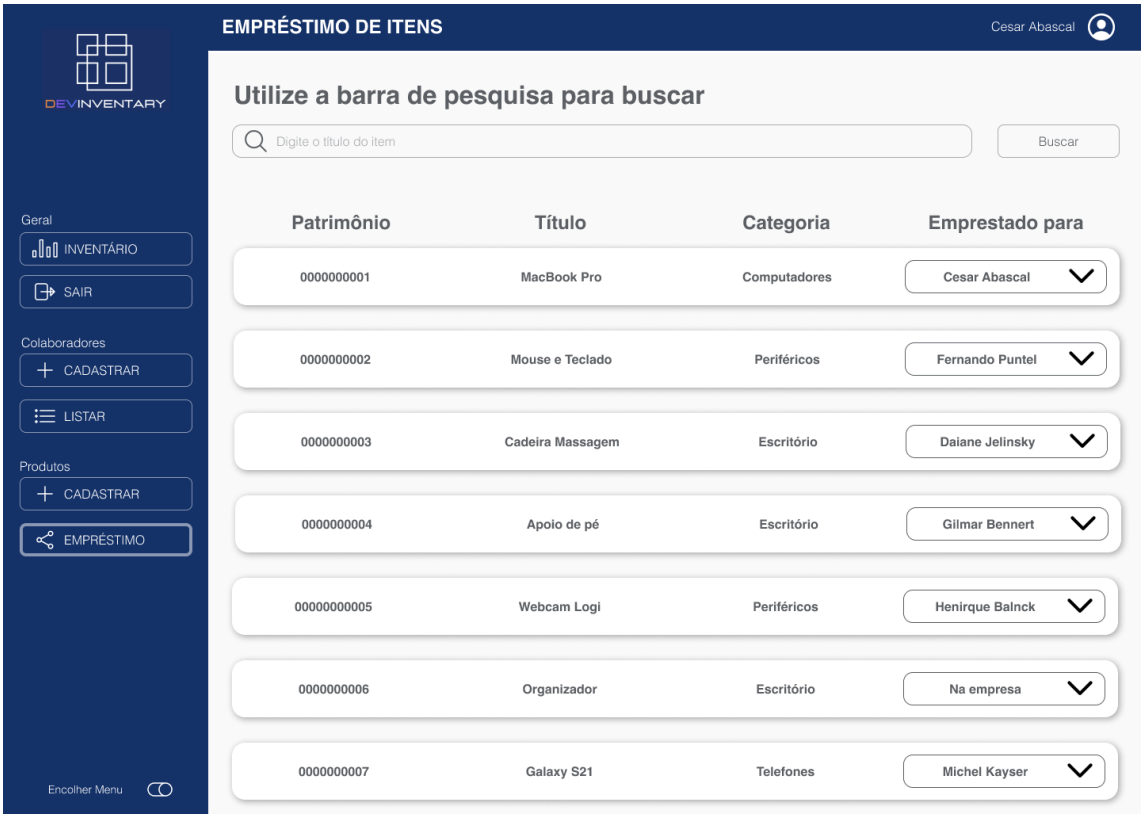

Imagem 6: Página de Empréstimo de Itens

4 ENTREGA

O código desenvolvido deverá ser submetido no **GitHub**, e o link deverá ser disponibilizado na tarefa **Módulo 1 - Projeto Avaliativo 2**, presente na semana 12 do AVA até o dia **29/05/2022** às **23h55**.

O repositório deverá ser **privado**, com as seguintes pessoas adicionadas:

- Gilmar Bennert - **GilmarBennertJr**
- Operação DEVinHouse - **devinhouse-operacao**

Não serão aceitos projetos submetidos após a data limite da atividade, e, ou alterados depois de entregues.

Importante:

1. Será considerado como data final de entrega a **última atualização** no repositório do projeto no GitHub. Lembre-se de não modificar o código até receber sua nota.
2. Não esqueça de submeter submeter o link no AVA.

5 CRITÉRIOS DE AVALIAÇÃO

A tabela abaixo apresenta os critérios que serão avaliados durante a correção do projeto. O mesmo possui variação de nota de 0 (zero) a 10 (dez) como nota mínima e máxima, e possui peso de **45% sobre a avaliação do módulo**.

Serão **desconsiderados e atribuída a nota 0 (zero)** os projetos que apresentarem plágio de soluções encontradas na internet ou de outros colegas. Lembre-se: Você está livre para utilizar outras soluções como base, mas **não é permitida** a cópia.

Uso adequado do GitHub				
Nº	**Critério de Avaliação**	**0**	**0,25**	**0,5**
1	As mensagens do commit estão claras e adequadas?	Os commits do aluno não tiveram nenhuma mensagem	Os alunos fizeram commits com comentários, mas esses estavam inadequados ou incompletos com relação ao conteúdo das alterações	As mensagens do commit estão claras e adequadas em relação ao conteúdo das alterações
2	Foi aberto um branch separado para cada etapa, funcionalidade?	Os commits foram realizados diretamente na master/main.	Foram abertas branchs para cada funcionalidade, porém foram realizados commits de mesma funcionalidade em branchs diferentes.	Foram abertas branchs para cada funcionalidades, e os commits foram realizados de forma concisa nas mesmas.

Desenvolvimento adequado da aplicação				
Nº	**Critério**	**0**	**0,25 a 0,50**	**0,75 a 1**
3	O aluno inovou construindo um layout agradável, responsivo e com acessibilidade?	O aluno não inovou e construiu um layout sem responsividade e acessibilidade.	O aluno inovou, construindo um layout agradável, personalizado e responsivo.	O aluno inovou, construindo um layout agradável, personalizado, responsivo e com acessibilidade.
4	O aluno estruturou o projeto, separando os recursos de acordo com o padrão de projetos?	Não foi efetuado a separação das telas e componentes e a criação do router e store estão sendo feitas direto no main.	Foi efetuado a separação parcial de telas e componentes ou inicialização do router e store em arquivos separados, com poucos erros na padronização.	Foi efetuado a separação de views e componentes, criação da store e router em seus devidos diretórios, separando regras de negócios e criação, sempre que necessário, em arquivos específicos.

5	Desenvolver a tela de login com a funcionalidade de autenticação funcional e com devidos tratamentos no formulário. Além de conter todos os elementos descritos no requisitos como acesso à tela de cadastro, login com Google e esqueci a senha.	O aluno não conseguiu desenvolver a tela ou entregou sem estar funcional.	O aluno desenvolveu a tela com a autenticação funcional porém com falhas na validação do formulário, alertas ou falta de redirecionamento.	Foi entregue a tela de login com todos os elementos do requisito, autenticação funcional com redirecionamento, alertas de avisos e falhas, além criar o guard com redirecionamento para tela inicial se usuário já estiver autenticado.
6	O aluno desenvolveu a tela ou modal de cadastro de usuário, guardando as informações no localStorage efetuando as devidas validações de formulário.	O aluno não conseguiu desenvolver a tela ou não está funcional.	Foi desenvolvida a tela de cadastro, armazenado o usuário no localStorage, porém com falhas na validação ou não utilizando o Vuex para gerenciar as regras de negócio.	Foi desenvolvida a tela de cadastro com todas as validações funcionais, armazenando o usuário no localStorage utilizando o VueX na regra de negócio, além de retornar para o usuário mensagem de conclusão.
7	O aluno criou o template do dashboard contendo o Sidebar e Navbar funcional de acordo com os requisitos.	O aluno não desenvolveu o template para o dashboard.	O aluno desenvolveu o template, porém sem o Sidebar ou Navbar ou com algum dos elementos desses componentes não funcionais.	Foi desenvolvido o template utilizando os componentes de Sidebar e Navbar, com todas as funcionalidades de acordo com os requisitos e com o guard permitindo o acesso somente com autenticação.
8	O aluno desenvolveu a tela de inventário contendo as estatísticas, listagem de produto, pesquisa e acesso a um modal ou tela de edição do produto.	O aluno não desenvolveu a tela ou apenas efetuou a criação da página com dados estáticos.	Foi desenvolvido a tela parcialmente com erros ou não utilizando o VueX.	Foi desenvolvido a tela de inventário, com todas as estatísticas, listagem de produtos, pesquisa e acesso à edição de produto funcional.
Nº	**Critério**	**0**	**0,25**	**0,50**

9	O aluno desenvolveu a tela de cadastro de colaborador de acordo com os requisitos, efetuando validação no formulário e armazenando os dados no localStorage.	O aluno não desenvolveu a tela ou não está funcional.	Foi desenvolvido parcialmente a tela ou com erros de validação, não salvando corretamente no localStorage ou não efetuando a busca de CEP com a API ViaCep.	Foi desenvolvido a tela de cadastro de colaborador funcional, com todas as validações, efetuando consulta na API, guardando os registros no localStorage e utilizando o VueX para gerenciar a regra de negócio.
11	Desenvolver a tela de listagem de colaborador com filtro.	O aluno não desenvolveu a tela de listagem ou entregou com dados estáticos.	Foi desenvolvido a tela de listagem, porém não buscou os dados corretamente, sem filtro ou não utilizando o VueX.	Foi desenvolvido a tela de listagem com todos os requisitos funcionais, além do uso do vue-gravatar.
Nº	**Critério**	**0**	**0,25 a 0,50**	**0,75 a 1**
12	O aluno desenvolveu a tela de cadastro de itens de acordo com os requisitos apresentados.	O aluno não desenvolveu a tela de cadastro de itens ou não está funcional.	Foi desenvolvido o cadastro de itens porém com falhas na validação ou não utilizando o VueX para gerenciar a regra de negócio.	Foi desenvolvido o cadastro de itens com todas as validações funcionais, salvando no localStorage, com as regras de negócio centralizadas, além de conter mensagem de sucesso e tratamento após salvar.
13	O aluno desenvolveu a tela de empréstimo de itens.	O aluno não desenvolveu a tela de empréstimo de itens ou não está funcional.	Foi desenvolvido a tela empréstimo de itens porém com falhas no funcionamento da pesquisa ou não apresentando detalhes dos itens corretamente.	Foi desenvolvido a tela de empréstimo de itens com o funcionamento de pesquisa, demonstrativo de detalhes, vínculo ao colaborador e regras de negócio centralizadas.

6 PLANO DE PROJETO

Ao construir a aplicação proposta, o aluno estará colocando em prática os aprendizados em:

- **HTML e CSS**: HTML5 (elementos semânticos) e CSS (seletores, principais estilos, layouts e flexbox)
- **Javascript**: Variáveis e tipos de dados, Operadores (aritméticos, lógicos e relacionais), Manipulação do DOM (utilização de seletores), Estrutura de Controle de Fluxo (condicional e repetição), Funções, Eventos, JSON, LocalStorage, Interval, Timeout, ES6+, Escopo (let, var e const), Classes, Atributos, Encapsulamento (closure), Herança, Polimorfismo, Arrow Functions, Módulos (export e import), Funções Assíncronas (Promises, Async e Await), Operadores Rest e Spread, Funções de Arrays (forEach, map, filter, find, reduce e every), fetch
- **VueJs**: Introdução (o que é e utilização) instalação e configuração (node, npm e vue-cli), hands-on, renderização de componentes, bootstrap, Eventos, componentes (props: comunicação entre componentes), formulários, validações, Diretivas (v-if/v-else, v-else-if, v-for, v-show, ...), renderização de listas, filtros, maxins, Animações e transições, roteamento, gerenciamento de estado (vuex), Consumo de API com Axios e Deploy
- **Versionamento**: utilização do GitHub como ferramenta de versionamento, utilizando os conceitos de GitFlow.