

DEVinHouse

Módulo 1 - Projeto Avaliativo

SUMÁRIO

1 INTRODUÇÃO	1
2 REQUISITOS DA APLICAÇÃO	1
2.1 GRAVAÇÃO DE VÍDEO	4
3 ROTEIRO DA APLICAÇÃO	4
4 CRITÉRIOS DE AVALIAÇÃO	10
5 ENTREGA	13
6 PLANO DE PROJETO	13
7 TUTORIAL JSON-SERVER	14

1 INTRODUÇÃO

A **LABFashion LTDA**, empresa líder no segmento tecnológico para gestão de moda, está com um projeto novo intitulado **LAB Clothing Collection**, um software **audacioso** para gestão de coleções de moda no setor de vestuário. O seu perfil chamou a atenção dos gestores, para criar a aplicação Front-End do software, que deverá ser construída utilizando o *framework* **Angular**.

2 REQUISITOS DA APLICAÇÃO

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

1. Uma **Tela de Login** contendo um formulário com campos de **email e senha**. Os campos de email e senha são obrigatórios (Validar ao disparar o evento de onSubmit). Ao clicar no botão de Entrar e passar pela validação, **verificar se é email e se a senha possui 8 caracteres**, redirecionar para tela de Dashboard, a tela contém ainda um redirecionamento para "Esqueci minha senha". *Seguir layout conforme o mockup disponibilizado.*
2. Uma **Tela de Esqueci a Senha** contendo um simples formulário com o campo de e-mail, adicionar validator na construção do formulário, esta tela possui um botão de

voltar que redireciona o usuário para tela de login.

3. Uma **Tela de Cadastro de Usuário** contendo um formulário com os campos de nome, nome da empresa, cnpj, e-mail, adicionar validator na construção do formulário, esta tela possui um botão de voltar que redireciona o usuário para tela de login.
4. Um **Menu Lateral**, contendo as opções **Dashboard, Coleções, Modelos, Obter Ajuda, Enviar Comentários e Logout**. Os botões de “Obter Ajuda” e “Enviar Comentários” não possuem ação, o botão de Logout redireciona o usuário para tela de login. *O menu deve ser desenvolvido como um componente à parte, para ser referenciado pela tela da aplicação. Seguir layout conforme o mockup disponibilizado.*
5. Uma **Tela de Dashboard** contendo 3 cards: Total de coleções (Exibir o total de coleções cadastradas no json-server) , modelos (Exibir total de modelos) e média de orçamento por coleção (Soma de todos os orçamentos das coleções / total de coleções).
 - a. A tela de Dashboard deve conter uma tabela contendo os 5 maiores orçamentos dentre as coleções, a tabela deve ser formada pelas colunas de Código da Coleção (ColeçãoID), Nome da Coleção, Orçamento e Responsável, ela deve estar ordenada do maior para o menor valor em orçamento.
6. Uma **Tela de Listagem de Coleções** (*consumir rota /colecoes do json-server*) contendo uma tabela com as colunas ColecaoID, nome da coleção, estação + lançamento (deve ser um dado combinado entre as propriedades) e responsável, além do botão **Criar Coleção** que redireciona o usuário para tela de cadastro de coleção. Além disso, as linhas da tabela deverão ser clicáveis e o clique na linha deve redirecionar o usuário para **Editar Coleção**. *Seguir layout conforme o mockup disponibilizado.*
7. Uma **Tela de Cadastro de Coleção**, contendo um formulário com os campos nome, responsável, estação, marca, orçamento e ano de lançamento . Ao clicar no botão salvar, chamar evento de onSubmit e cadastrar unidade via POST na rota /colecoes do json-server, o botão “Cancelar” fará com o que o usuário seja enviado para Listagem de Coleções. Todos os campos do formulário são obrigatórios. *Seguir layout conforme o mockup disponibilizado.*
8. Uma **Tela de Edição de Coleção**, contendo um formulário com os campos nome, responsável, estação, marca, orçamento e ano de lançamento. Essa tela precisa receber um parâmetro na rota para identificar que coleção o usuário deseja editar, mostrar no formulário os dados da coleção ao carregar as informações sobre ela. Ao

clicar no botão salvar, chamar evento de onSubmit e atualizar a unidade via PUT na rota /colecoes/:id do json-server, o botão “Cancelar” fará com o que o usuário seja enviado para Listagem de Coleções. Todos os campos do formulário são obrigatórios. *Seguir layout conforme o mockup disponibilizado.*

9. Uma **Tela de Listagem de Modelos** (consumir rota /modelos do json-server) contendo uma tabela com as colunas ModeloID, nome do modelo, coleção relacionada e responsável, além do botão **Criar Modelo** que redireciona o usuário para tela de cadastro de modelo. Além disso, as linhas da tabela deverão ser clicáveis e o clique na linha deve redirecionar o usuário para **Editar Modelo**. *Seguir layout conforme o mockup disponibilizado.*
10. Uma **Tela de Cadastro de Modelos**, contendo um formulário com os campos nome, responsável, tipo (tag **select** do HTML com as opções: bermuda, biquíni, bolsa, boné, calça, calçados, camisa, chapéu, saia.), coleção relacionada (listar todas as coleções cadastradas no json-server) e dois “radio group” para verificar se o modelo possui bordado e se possui estampa. Ao clicar no botão salvar, chamar evento de onSubmit e cadastrar unidade via POST na rota /modelos do json-server, o botão “Cancelar” fará com o que o usuário seja enviado para Listagem de Modelos. Todos os campos do formulário são obrigatórios. *Seguir layout conforme o mockup disponibilizado.*
11. Uma **Tela de Edição de Modelos**, contendo um formulário com os mesmos campos presentes no **Cadastro de Modelos**. Essa tela precisa receber um parâmetro na rota para identificar que modelo o usuário deseja editar, mostrar no formulário os dados do modelo ao carregar as informações sobre ele. Ao clicar no botão salvar, chamar evento de onSubmit e atualizar a unidade via PUT na rota /modelos/:id do json-server, o botão “Cancelar” fará com o que o usuário seja enviado para Listagem de Modelos, além disso um botão Excluir deverá aparecer na tela e o clique será chamado o método DELETE na rota /modelos/:id para remover o modelo, ao fim da exclusão retornar o usuário para “Listagem de Modelos”. Todos os campos do formulário são obrigatórios. *Seguir layout conforme o mockup disponibilizado.*
12. Para aplicação, deve ser pensado em aproveitar o melhor dos recursos do Angular para redução de código e evitar complexidade e grandes blocos em um só componente, portanto, utilize na aplicação os conceitos aprendidos de layouts, componentes para um melhor projeto.

2.1 GRAVAÇÃO DE VÍDEO

Além do desenvolvimento deste sistema você deverá gravar um vídeo, com tempo máximo de 3 minutos, abordando os seguintes questionamentos:

- Qual o objetivo do sistema? E demonstração de funcionamento.
- O que deve ser realizado para executar o sistema?
- Como você organizou as tarefas antes de começar a desenvolver?
- Quais as *branches* você criou e quais os objetivos para cada uma?
- Você acha que faltou algo no seu código que você poderia melhorar?

Você poderá gravar na vertical ou na horizontal. É importante que apareça seu rosto e esteja em um local com boa iluminação. Para realizar a entrega do vídeo, coloque em uma pasta do **Google Drive** em modo leitor para qualquer pessoa com o link, e compartilhe o mesmo na submissão do projeto no AVA. Uma dica interessante é você inserir o vídeo no README.md do seu projeto no repositório do Github.

3 ROTEIRO DA APLICAÇÃO

A aplicação deverá conter os requisitos apresentados anteriormente, sendo codificada em **html**, **css** e **javascript** através do *framework* **Angular**, 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: [LAB Clothing Collection](#)

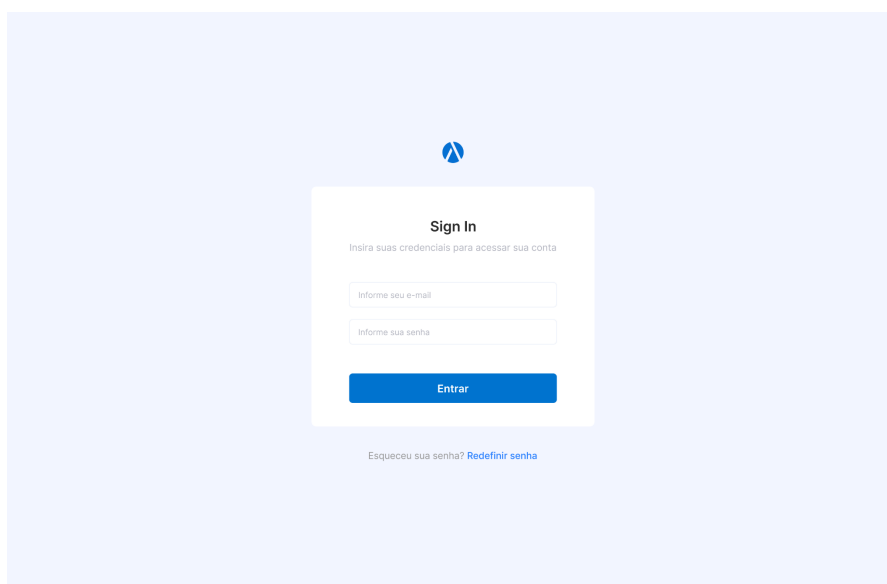


Imagem 1: Tela de Login

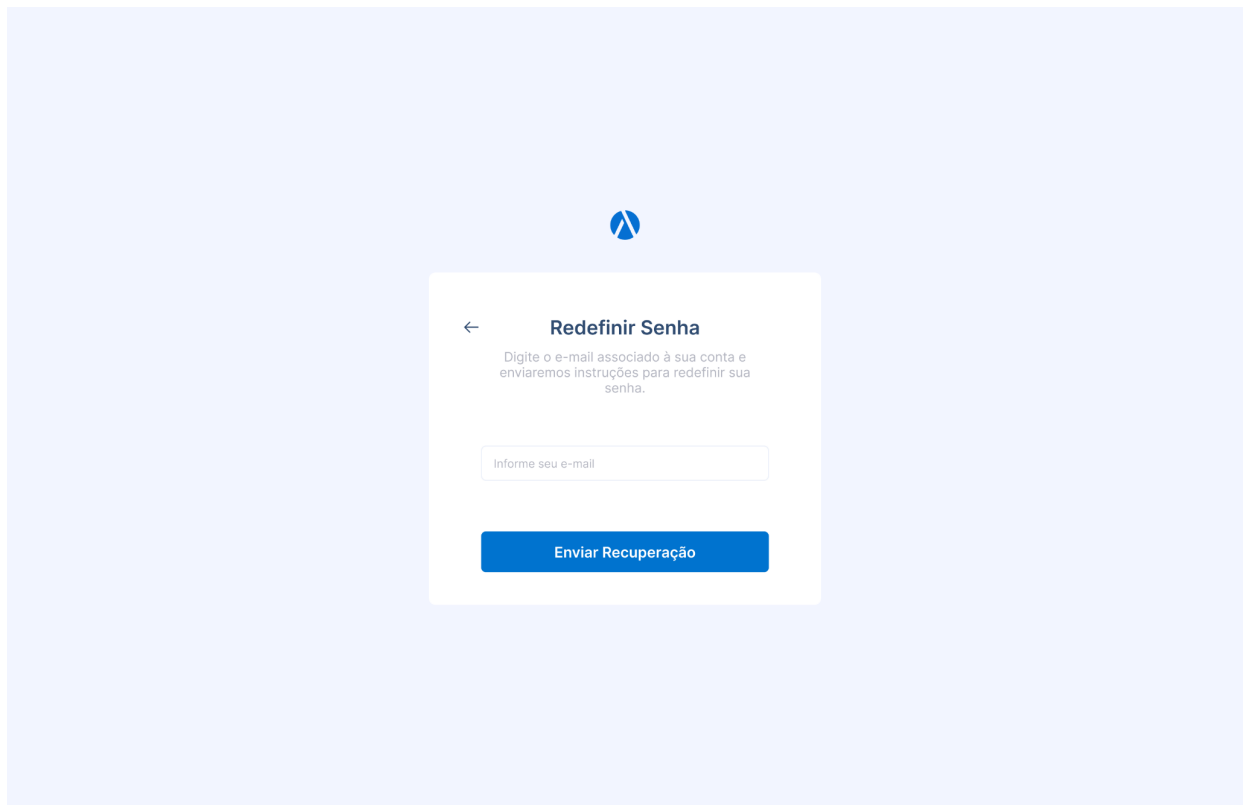


Imagem 2: Tela de Recuperação de Senha

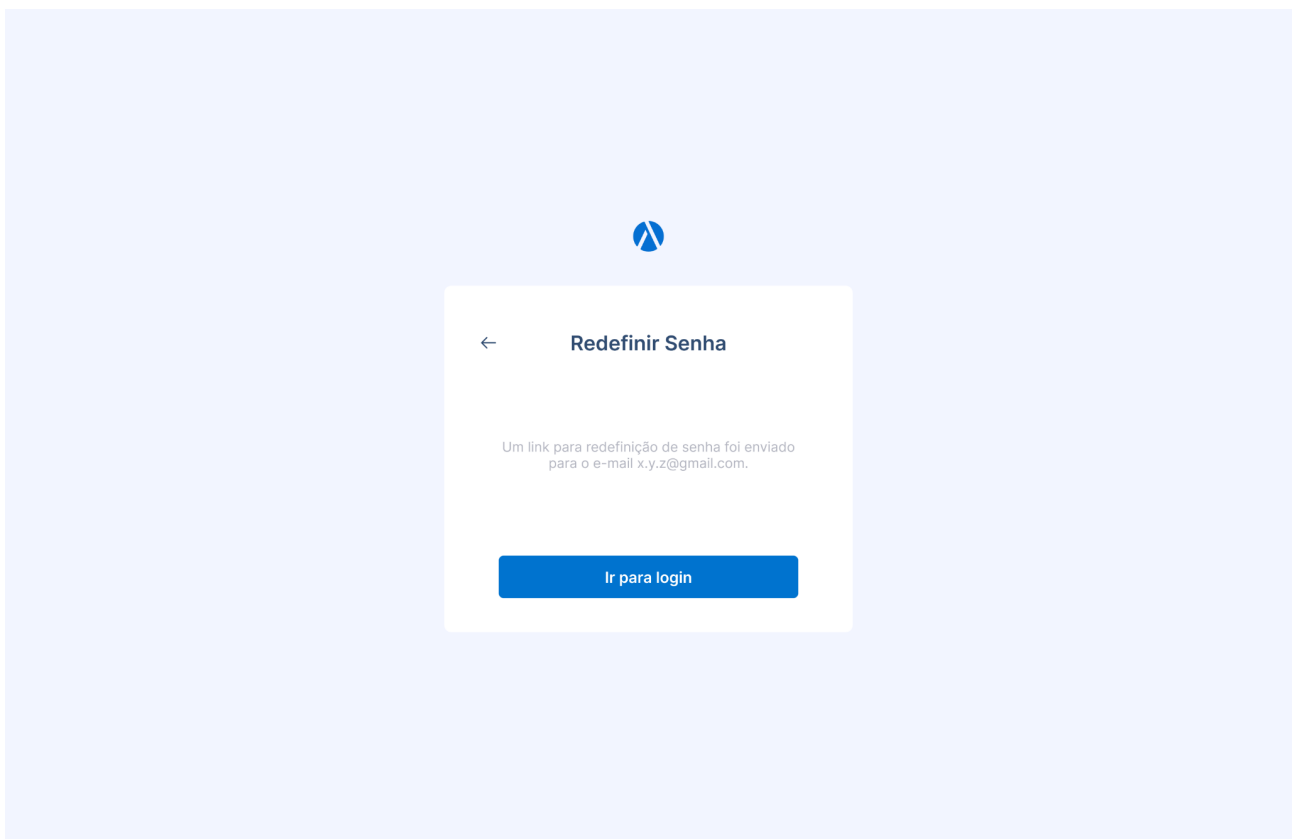



Imagem 3: Tela de Recuperação de Senha (Link enviado)




Sign Up

Insira seus dados para criar uma conta

Criar Conta

Esqueceu sua senha? [Redefinir senha](#)

Imagem 3: Tela de Criação de conta



MENU

Dashboard

Coleções

Modelos

Obter Ajuda

Comentários

Dashboard

Coleções

25

Modelos

64

Orçamento Médio (R\$)

4700

↑ 25%

Comparado com 01/23

Maiores Orçamentos

Coleção	Responsável	Modelos	Orçamento
Adidas	Yan Esteves	10	R\$ 9500,00
Renner	Yan Esteves	10	R\$ 3500,00
PatBO	Yan Esteves	10	R\$ 8700,00
Nike .	Yan Esteves	10	R\$ 9100,00

Imagem 3: Tela de Home/Dashboard

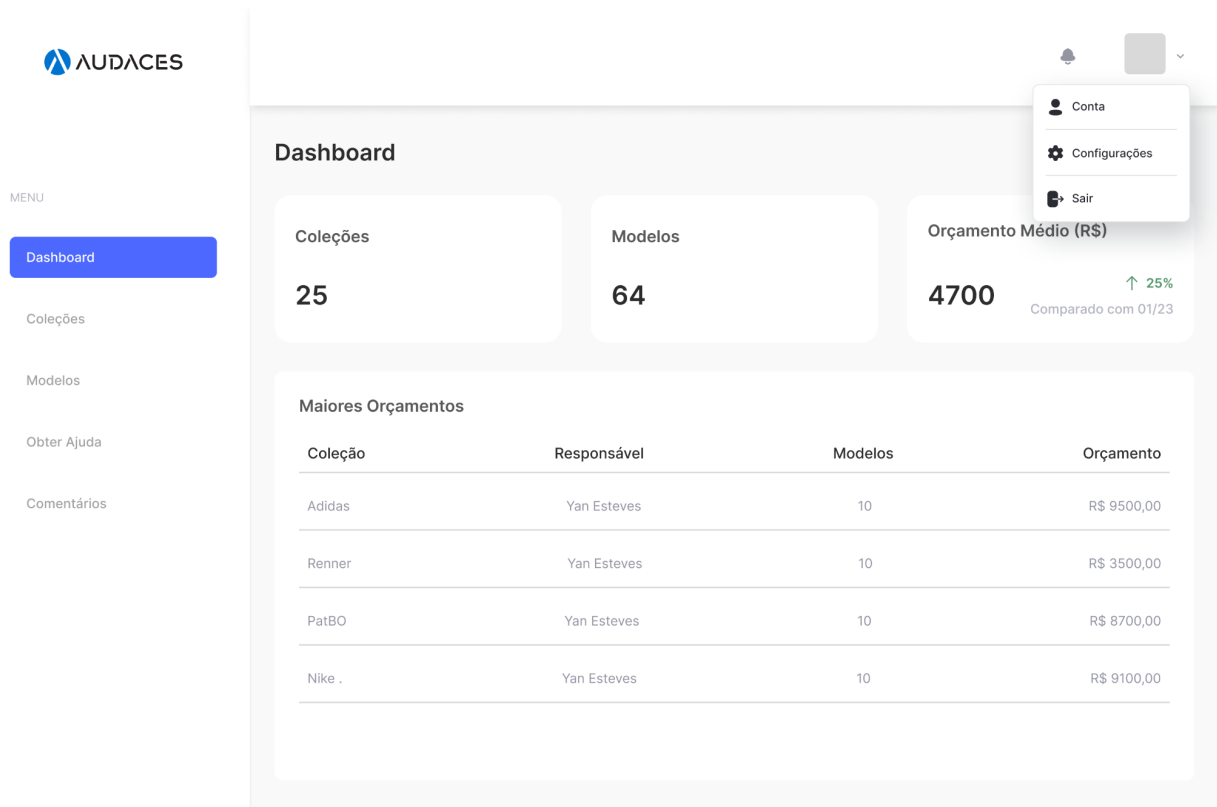


Imagem 4: Tela de Home/Dashboard

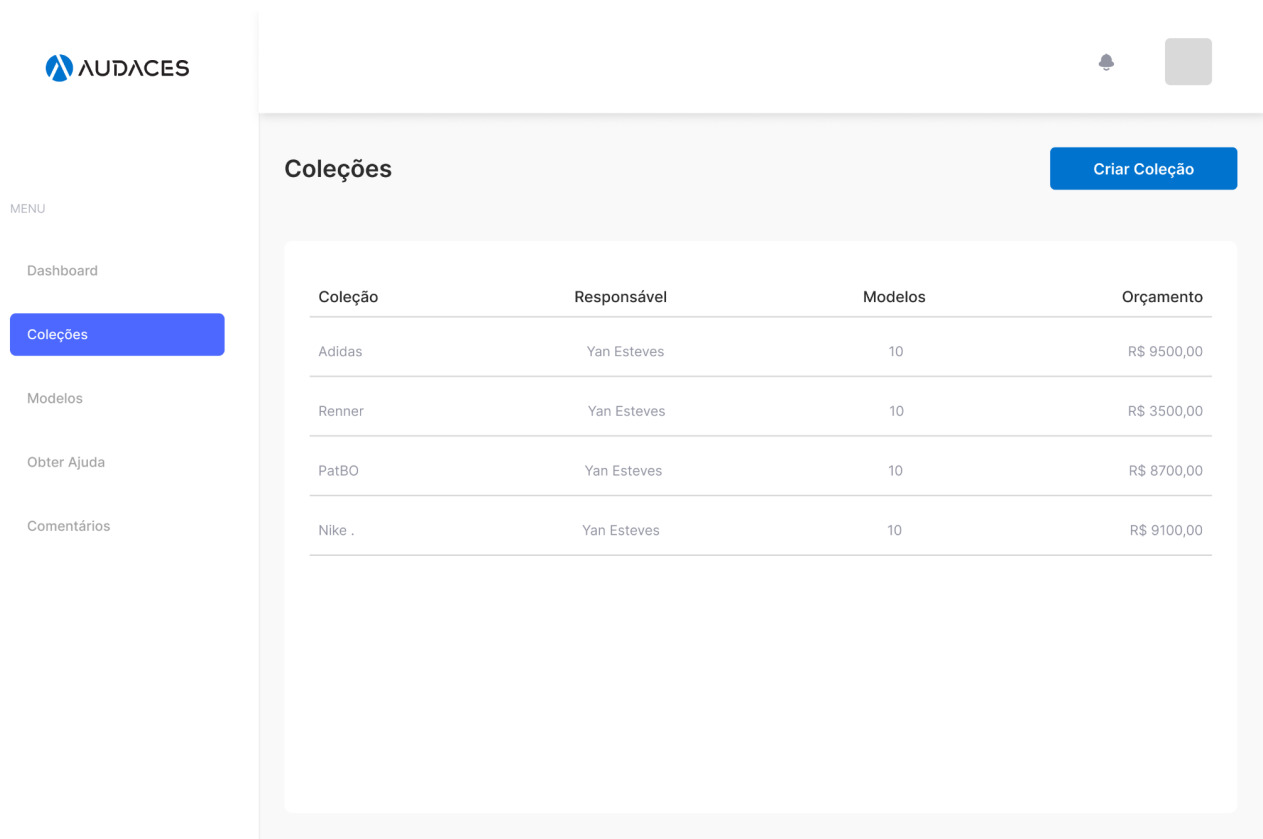



Imagem 3: Tela de Listagem de Coleções



MENU

Dashboard


Coleções

Modelos

Obter Ajuda

Comentários


Criar Coleção



Cancelar

Salvar

Imagem 4: Tela de Cadastro de Coleção



MENU

Dashboard


Coleções

Modelos

Obter Ajuda

Comentários

Editar Coleção



Excluir

Cancelar

Atualizar

Imagem 5: Tela de Edição de Coleção

8

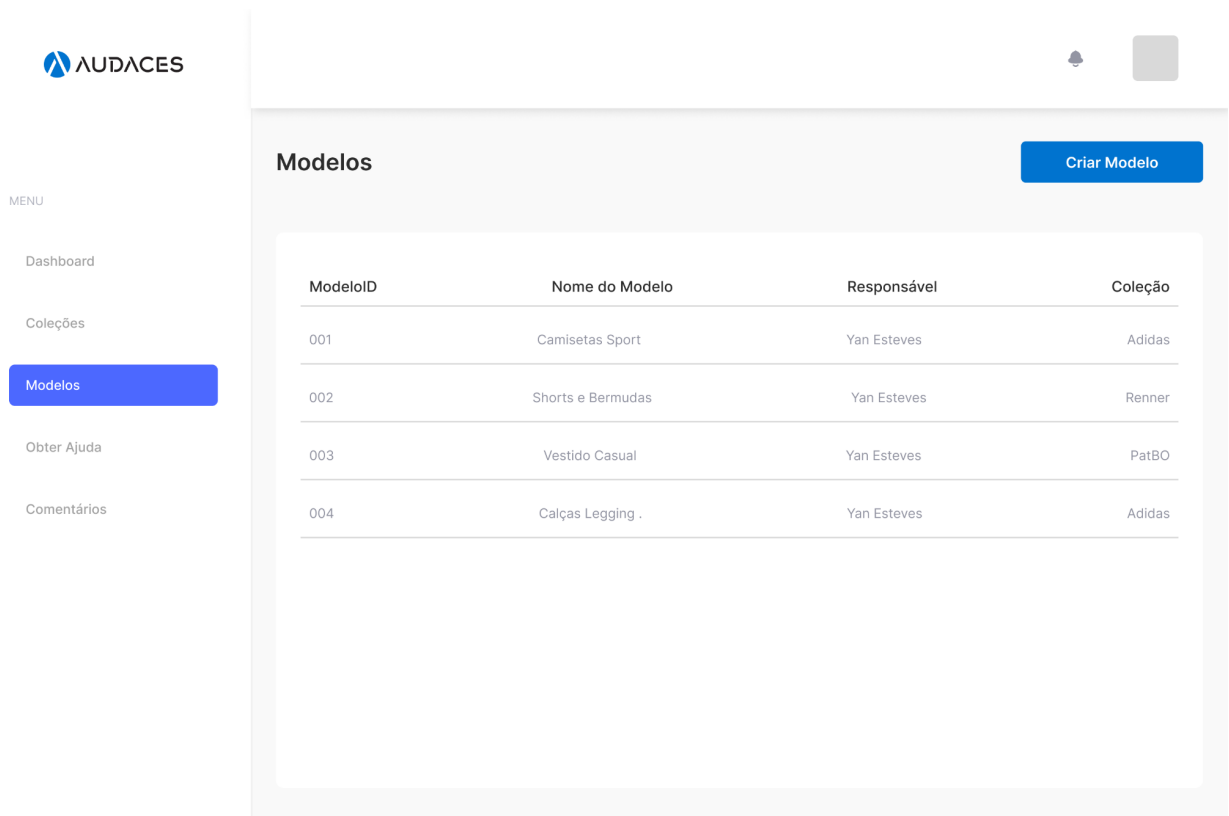


Imagem 6: Tela de Listagem de Modelos

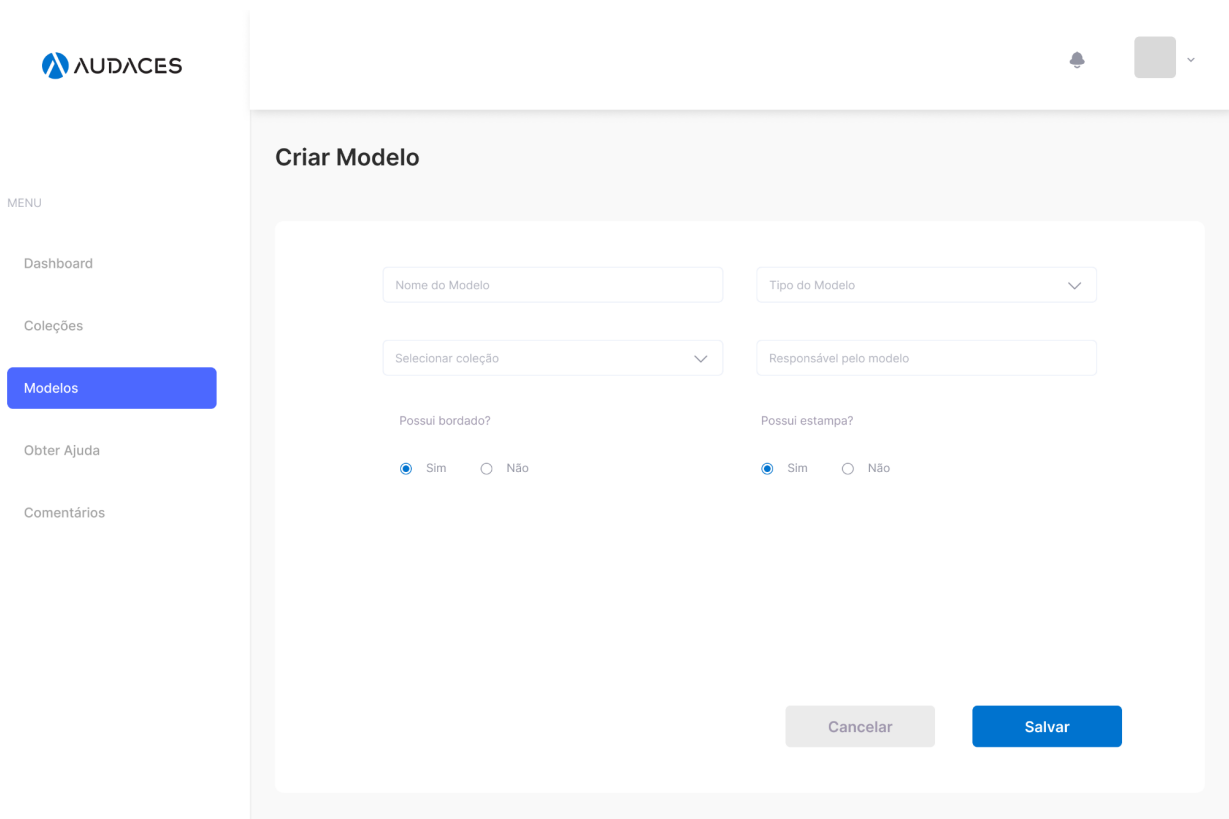


Imagem 7: Tela de Criar Modelo

MENU

Dashboard

Coleções

Modelos

Obter Ajuda

Comentários

Editar Modelo

Nome do Modelo

Tipo do Modelo

Selecionar coleção

Responsável pelo modelo

Possui bordado?

Possui estampa?

☒ Sim
 ☐ Não

☒ Sim
 ☐ Não

Excluir

Cancelar

Atualizar

Imagem 8: Tela de Editar Modelo

O fluxo do projeto pode ser conferido por meio deste link: [Fluxo](#)

4 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 **60% 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.

Nº	Critério de Avaliação	0	1	1,5	2
1	Cards no Trello	Não foram desenvolvidos cards no trello para	Foram desenvolvidos alguns cards mas não detalhou o	Alguns cards foram bem declarados, mas alguns itens importantes da	O Aluno declarou bem as tarefas a serem realizadas e documentou nos cards as

		organização do projeto.	que era para ser desenvolvido.	aplicação não foram documentados.	necessidades de cada tarefa.
Nº	Critério de Avaliação	0	0,5	0,75	1
2	Vídeo de até 3 minutos explicando a solução	O aluno não realizou a gravação de um vídeo explicando o que e como foi desenvolvido.	O aluno gravou um vídeo mas não explicou por completo cada etapa do desenvolvimento.	Foi gravado um vídeo, apresentando alguns detalhes técnicos mas não passou pelo sistema.	O aluno gravou um vídeo sobre a aplicação, percorrendo ela toda e explicando cada detalhe técnico aplicado nos recursos.
3	Desenvolveu a página de Dashboard?	Não desenvolveu a página de Dashboard.	Inseriu os cards e a tabela, porém os valores estavam estáticos (Não usou chamadas para API para a montagem dos dados).	Inseriu os cards utilizando as informações dinâmicas da API, porém a tabela estava com os dados estáticos ou não ordenados como previsto (Não usou chamadas para API para a montagem dos dados).	Inseriu os cards e a tabela ordenada usando informações da API (Usou chamadas para API para a montagem dos dados).
4	Conseguiu desenvolver o componente de menu da aplicação?	Não conseguiu desenvolver o componente de menu da aplicação e nem o roteamento.	Conseguiu desenvolver o componente de menu, porém não configurou o roteamento da aplicação.	Conseguiu desenvolver o componente de menu e o roteamento da aplicação, mas não importou o componente de Menu no lugar correto da aplicação.	Conseguiu desenvolver o componente de menu e o roteamento da aplicação, além de importar o componente de Menu no lugar correto da aplicação.
5	Desenvolveu a tela de cadastro e edição de coleções?	Não desenvolveu uma tela de cadastro de coleções.	Inseriu os campos, porém não desenvolveu o formulário com validação para validar no submit.	Inseriu os campos de input com validação, porém não implementou o evento de onSubmit para cadastrar ou atualizar.	O aluno inseriu os campos corretamente e o sistema cadastra/atualiza os valores na rota /colecoes.
6	Desenvolveu a tela de cadastro de modelos e edição de modelos?	Não desenvolveu uma tela de cadastro /	Inseriu os campos, porém não desenvolveu o formulário com	Inseriu os campos de input com validação, porém não implementou	Inseriu todos os campos corretamente e cadastrou e

		edição de modelos.	validação para validar no onSubmit.	o evento de onSubmit (Cadastrar ou atualizar via POST no json-server).	atualizou os valores na rota /modelos via POST dentro do evento de onSubmit após a validação de formulário.
8	Desenvolveu uma página que apresenta um design agradável e intuitivo?	Desenvolveu uma página que não apresenta um design agradável e intuitivo	Desenvolveu uma página e inseriu alguns estilos, como tipo/tamanho/cor de fonte, largura/altura/margem de alguns elementos, mas todos inseridos diretamente no HTML.	Desenvolveu uma página e inseriu alguns estilos, bem organizados no seu próprio arquivo, separado do HTML.	Conseguiu apresentar estilos textuais e de posicionamento básico, num arquivo separado do HTML, também estilizou o fundo da página, os botões, a lista e economizou texto de botão quando não havia muito espaço.
9	Desenvolveu um código que está bem organizado e é facilmente legível, conforme as boas práticas propostas pelos grandes nomes do desenvolvimento de software?	Não conseguiu inserir um código legível e sem uma boa organização de pastas do projeto.	O código TS desenvolvido está bagunçado, com nomes de variáveis e funções não explicativas, sem indentação correta e não segue os padrões de boas práticas, quanto a organização do Angular.	O código TS desenvolvido está separado em um arquivo diferente do HTML, mas ainda apresenta algum problema de organização/legibilidade (indentação incorreta, nomes de variáveis/funções não explicativos)	O código JS desenvolvido está separado em um arquivo diferente do HTML, bem organizado, com nomes de funções e variáveis explicativos e indentação correta.
10	Desenvolveu uma tela de Login, Cadastro e Esqueci a senha com formulário com validação e conforme o mockup?	Não conseguiu desenvolver alguma das telas do critério.	Conseguiu implementar as telas mas não desenvolveu com formulário reativo e validação.	Conseguiu desenvolver as telas com o formulário com validação, porém não desenvolveu o evento de onSubmit nas páginas.	Conseguiu desenvolver todos os elementos das páginas e eventos de onSubmit do formulário.

5 ENTREGA

O código desenvolvido deverá ser submetido no **GitHub**, e o link deverá ser disponibilizado junto com o link do **Trello** e o link do **vídeo** na tarefa **Módulo 1 - Projeto Avaliativo**, presente na semana 12 do AVA até o dia **02/04/2023** às **23h55**.

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

- Yan Esteves - **yanestevesufjf**
- Operação DEVinHouse - **devinhouse-operacao**

Importante:

1. Não serão aceitos projetos submetidos **após a data limite da atividade**, e, ou **alterados** depois de entregues. Lembre-se de **não modificar** o código no GitHub até receber sua nota e feedback.
2. Não esqueça de **submeter os links no AVA**. Não serão aceitos projetos em que os links não tenham sido submetidos.

6 PLANO DE PROJETO

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

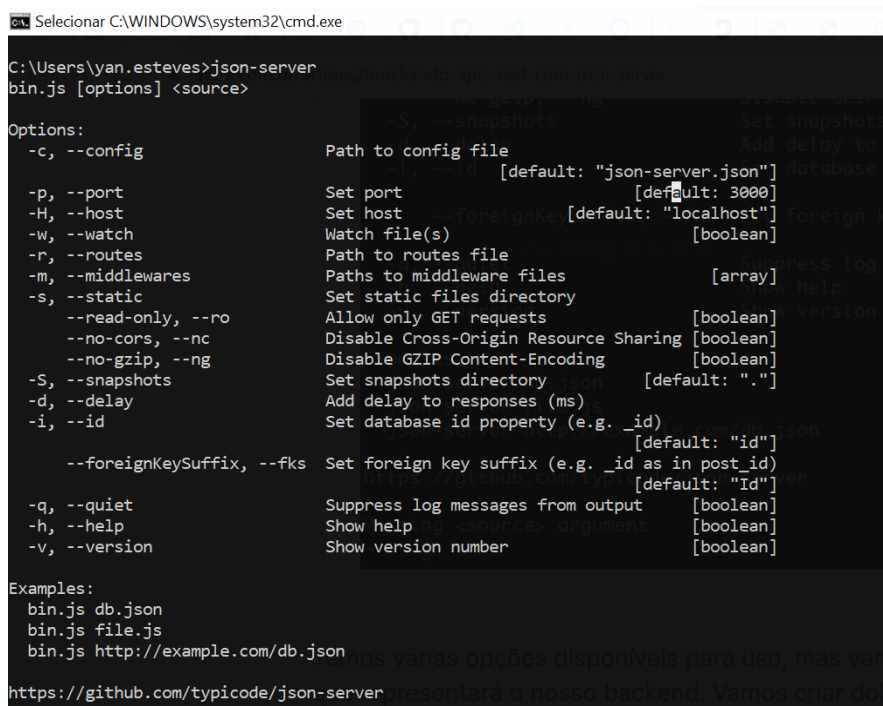
- **HTML**: principais tags como head, title, body, div, h1, form, input, button, ul, li. Atributos de tags como class, id, type.
- **CSS**: estilizar a página, os botões, inputs, alterar atributos dos elementos da tela de acordo com a interação do usuário para uma melhor experiência do usuário (UX), Alinhamento de elementos com flex-box.
- **Javascript**: variáveis, arrays, funções .map(), .filter(), .reducer(), manipulação de eventos onChange, onSubmit, json-server.
- **Angular**: componentes, rotas, módulos, conexão com serviços, uso de Angular Material/Bootstrap, validação de formulários.

7 TUTORIAL JSON-SERVER

O projeto precisará de uma *API Fake*, e para isso utilizaremos o plugin *json-server* que deverá ser instalado globalmente por meio do *npm* com o seguinte comando no *CMD*:

```
npm install -g json-server
```

Com o plugin instalado, poderemos utilizar o comando *json-server* e verificar se está instalado e os parâmetros que podemos utilizar, abaixo vemos uma imagem após o comando anterior:



```
Selecionar C:\WINDOWS\system32\cmd.exe
C:\Users\yan.esteves>json-server
bin.js [options] <source>

Options:
  -c, --config          Path to config file
                        [default: "json-server.json"]
  -p, --port            Set port
                        [default: 3000]
  -H, --host            Set host
                        [default: "localhost"]
  -w, --watch           Watch file(s)
                        [boolean]
  -r, --routes          Path to routes file
  -m, --middlewares    Paths to middleware files
                        [array]
  -s, --static          Set static files directory
  --read-only, --ro    Allow only GET requests
                        [boolean]
  --no-cors, --nc      Disable Cross-Origin Resource Sharing
                        [boolean]
  --no-gzip, --ng      Disable GZIP Content-Encoding
                        [boolean]
  -S, --snapshots       Set snapshots directory
                        [default: "."]
  -d, --delay           Add delay to responses (ms)
  -i, --id              Set database id property (e.g. _id)
                        [default: "id"]
  --foreignKeySuffix, --fks Set foreign key suffix (e.g. _id as in post_id)
                        [default: "Id"]
  -q, --quiet           Suppress log messages from output
                        [boolean]
  -h, --help            Show help
                        [boolean]
  -v, --version         Show version number
                        [boolean]

Examples:
  bin.js db.json
  bin.js file.js
  bin.js http://example.com/db.json

https://github.com/typicode/json-server
```

Imagem 9: Instalação do json-server

Para mais informações sobre *json-server*, você pode acessar aqui: [Simulando API](#)

Repositório de exemplo com um *db.json* para o projeto, baixar o arquivo e modificar de acordo com os campos dos formulários existentes: [Repositório](#)

Para executar nossa API Fake, utilizamos o comando *json-server nome_arquivo.json* como na imagem abaixo:

C:\WINDOWS\system32\cmd.exe - json-server db.json

```
C:\Users\yan.esteves\Documents\json-server>dir
O volume na unidade C é OS
O Número de Série do Volume é 5296-2D41

Pasta de C:\Users\yan.esteves\Documents\json-server

13/05/2022  19:46    <DIR> .
13/05/2022  19:46    <DIR> ..
13/05/2022  19:47    db.json 558 db.json
                1 arquivo(s) 558 bytes
                2 pasta(s) 41.217.912.832 bytes disponíveis

C:\Users\yan.esteves\Documents\json-server>json-server db.json

\{^_^}/ hi!
Loading db.json
Done

Resources
http://localhost:3000/projetos
http://localhost:3000/tarefas

Home
http://localhost:3000

Type s + enter at any time to create a snapshot of the database
```

Imagem 10: Iniciando a API Fake

Após estes passos o servidor será iniciado na porta 3000, como declarado em Resources. Assim poderemos utilizar os webservice para estarem consumindo as informações presentes no JSON que adicionamos.