

Projeto 16. Checklist

Projeto rejeitado sem revisão

- Executar o comando `npm run start` ou `npm run dev` resulta em um erro.
- A exceção `node_modules` não foi adicionada ao `.gitignore`.
- Nem todas as funcionalidades necessárias foram implementadas.
- Há perguntas direcionadas para os revisores no projeto.
- Erros críticos que foram encontrados durante uma revisão (ou revisões) anterior (es), e não foram resolvidos.

Requisitos do projeto

Geral

- O projeto contém:
 - `package.json` ;
 - `.editorconfig` ;
 - `.eslintrc` necessário para estender a configuração do `airbnb-base` . Ele também contém as `devDependencies` necessárias para que o linter funcione corretamente;
 - `.gitignore` . O arquivo `.gitignore` contém pelo menos uma pasta `node_modules` .
 - O arquivo `README.md`
- O arquivo README.md contém os seguintes itens:
 1. O nome do projeto
 2. Uma descrição do projeto e sua funcionalidade
 3. Uma descrição das tecnologias e técnicas utilizadas
 4. Imagens, GIFs ou capturas de tela que detalham os recursos do projeto (altamente recomendado)
 5. Uma demonstração em vídeo do seu projeto (altamente recomendado).

O projeto está em conformidade com os seguintes requisitos de estilo de código

- camelCase é usado para os nomes de funções e variáveis.
- Somente substantivos são usado como nomes de variáveis.
- Os nomes de variáveis descrevem o que está armazenado neles. Se o projeto tiver diversas variáveis com dados parecidos, então estas variáveis terão nomes exclusivos, mas descritivos.
- Os nomes descritivos são usados para as funções, refletindo o que elas fazem.
- Os nomes das funções começam com um verbo.
- Classes JS e componentes funcionais foram nomeados usando substantivos e começam com letra maiúscula.
- Os nomes não devem incluir abreviações impróprias ou pouco claras.

A funcionalidade foi implementada de acordo com o roteiro

- Todas as seções do projeto foram programadas.
- 6 cartões foram criados e renderizados via JavaScript.

Nome da sessão

Ao mesmo tempo:

- Não há dependências desnecessárias em `package.json`.
- Uma exceção foi adicionada em `_id`.
- O comando `npm run start` inicia o servidor no `localhost:3000`.
- O comando `npm run dev` inicia o servidor em `localhost:3000` com a recarga automática habilitada.
- Quando o app iniciar, ele se conecta ao servidor MongoDB:
- O esquema do usuário, com os seguintes campos, foi descrito no aplicativo: `nome` (string de 2 a 30 caracteres, campo obrigatório), `sobre` (string de 2 a 30 caracteres, campo obrigatório), e `avatar` (string para uma URL, campo obrigatório). Todos os campos foram validados.

- O esquema do cartão, com os seguintes campos, é descrito no aplicativo: `nome` (string de 2 a 30 caracteres, campo obrigatório), `link` (string para um URL, campo obrigatório), `proprietário` (ObjectId, campo obrigatório), `curtidas` (array ObjectId, vazio por obrigatório), e `createdAt` (data de criação, valor padrão `Date.now`).
- Os campos `avatar` e `link` foram validados usando uma expressão regular. O modelo é capaz de encontrar URL nos seguintes formatos:
- Modelos chamados `usuários` e `card` foram criados e exportados dentro dos arquivos do esquema.
- Quando atualizar um usuário ou cards, `new: true` é passado para opções.

Rotas para os usuários estão funcionando:

- `GET /users` — retorna todos os usuários;
- `GET /users/:userId` — retorna um usuário por `_id`;
- `POST /users` — cria um novo usuário
- `PATCH /users/me` — atualiza um perfil.
- `PATCH /users/me/avatar` — atualiza um avatar.

Rotas para cards estão funcionando:

- `GET /cards` — retorna todos os cards do banco de dados;
- `POST /cards` — cria um card com `nome` e `link` passados no corpo da solicitação. `proprietário` deve ser definido.
- `DELETE /cards/:cardId` — deleta um card por `_id`;
- `PUT /cards/:cardId/likes` — curte um card;
- `DELETE /cards/:cardId/likes` — não curte um card;

Se algo der errado em alguma das solicitações, o servidor retorna uma resposta com um erro, e o status correspondente:

- **400** — dados inválidos passados aos métodos para criar um card/usuário ou para atualizar um perfil de usuário ou avatar

- **404** — não existe usuário com o id solicitado ou a solicitação foi enviada para um endereço inexistente
- **500** — erro padrão. Acompanhado pela mensagem: "Ocorreu um erro no servidor";
- Antes de enviar um erro, certifique-se de verificá-lo como mostrado abaixo:
- Os controladores garantem o envio de uma mensagem de erro.
- Os status de erro são movidos para constantes.
- Uma resposta de erro tem apenas o campo `message` (mensagem). A mensagem de erro corresponde com seu tipo.
 - se o servidor responder com sucesso, o JSON foi retornado.
 - O servidor não trava em solicitações inválidas, e não há erros no console.

Além disso

- A pasta `data` foi removida do projeto. Todos os dados devem ser retirados do banco de dados.
- O projeto utiliza o ES6 sempre que possível.
- O projeto tem uma estrutura clara, o código está dividido em rotas, modelos, e controladores:

- ▲ controllers
 - 📄 cards.js
 - 📄 users.js
- ▲ models
 - 📄 card.js
 - 📄 user.js
- ▶ node_modules
- ▲ routes
 - JS cards.js
 - JS users.js
- ⚙ .editorconfig
- ⚙ .eslintrc
- JS app.js
- 📄 package.json