

## Bootcamp IGTI: Desenvolvimento Full Stack

### Desafio Final do Curso

<b>Módulo 5</b>	<b>Desafio Final Bootcamp – versão 1.0.2</b>
-----------------	--

#### Objetivos

Exercitar os seguintes conceitos trabalhados nos Módulos 01 a 04:

- ✓ Implementação de algoritmos com JavaScript.
- ✓ Criação de APIs com **Node.js** e **Express**.
- ✓ Criação de componentes com **React** com Class Components e/ou Functional Components.
- ✓ Persistência de dados com **MongoDB** e **Mongoose**.
- ✓ Implantação de aplicações web com **Heroku**.

#### Enunciado

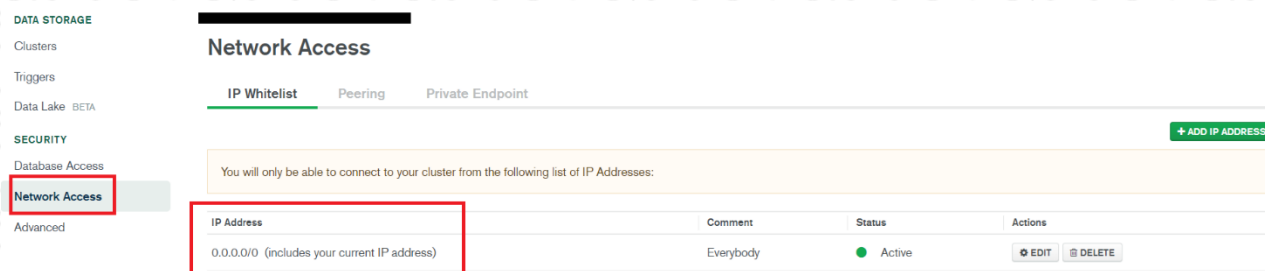
Criar uma aplicação web para **Controle Financeiro Pessoal** com **MongoDB + Node.js + React** e implantação no **Heroku**.

## Atividades

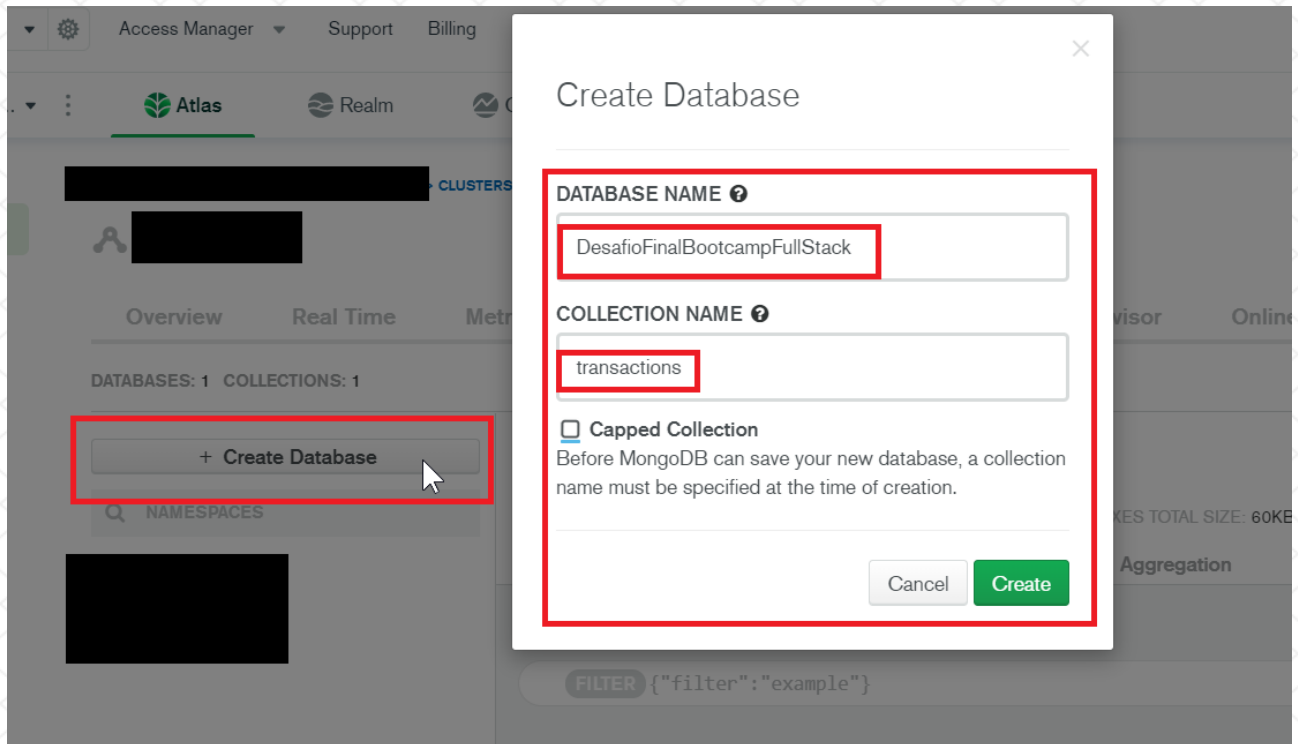
O aluno deverá desempenhar as seguintes atividades:

### **Etapa 1 – Criação de Database e Collection no MongoDB e importação dos dados originais:**

1. **Pré-requisitos:** o aluno já deve ter instalado o **Node.js** (recomenda-se a versão 12.9.1 ou superior) e o **Yarn** (recomenda-se a versão 1.22.4 ou superior, desde que se mantenha a versão 1.x) em seu computador. Além disso, o aluno já deve possuir sua conta devidamente criada no serviço [MongoDB Atlas](#) e já ter criado o seu Cluster gratuito. Para não ter problemas com conexões, configure o cluster para aceitar conexões de 0.0.0.0. Isso pode ser feito através da tela "Network Access".



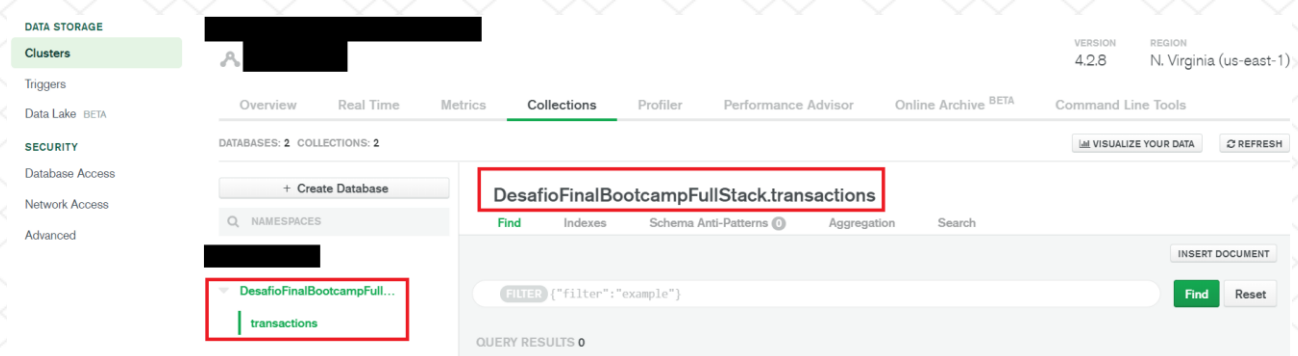
2. Acesse o Cluster já criado no MongoDB Atlas. Em seguida, acesse a tela de **Collections** e crie um novo **Database**, conforme imagem abaixo. **Sugere-se que o aluno utilize os mesmos nomes que utilizei, pois eles serão utilizados durante todo este tutorial.** Nomes diferentes podem o confundir durante o desenvolvimento do desafio, tornando mais difícil o suporte nos fóruns.



Database Name – **DesafioFinalBootcampFullStack**

Collection Name – **transactions**

3. Certifique-se de que tanto o **Database** quanto a **Collection** foram devidamente criados.



4. Acesse a aplicação Node.js fornecida pelo professor no "**Fórum de Avisos**" do Módulo (projeto "**mongodb-import**") e execute o comando **yarn** para instalar todas as dependências já definidas em **package.json**. Crie o arquivo ".env" na raiz do projeto e preencha "DB\_CONNECTION" com os dados pertinentes ao **seu Banco de Dados**. As dicas de como preencher o arquivo ".env" estão no arquivo



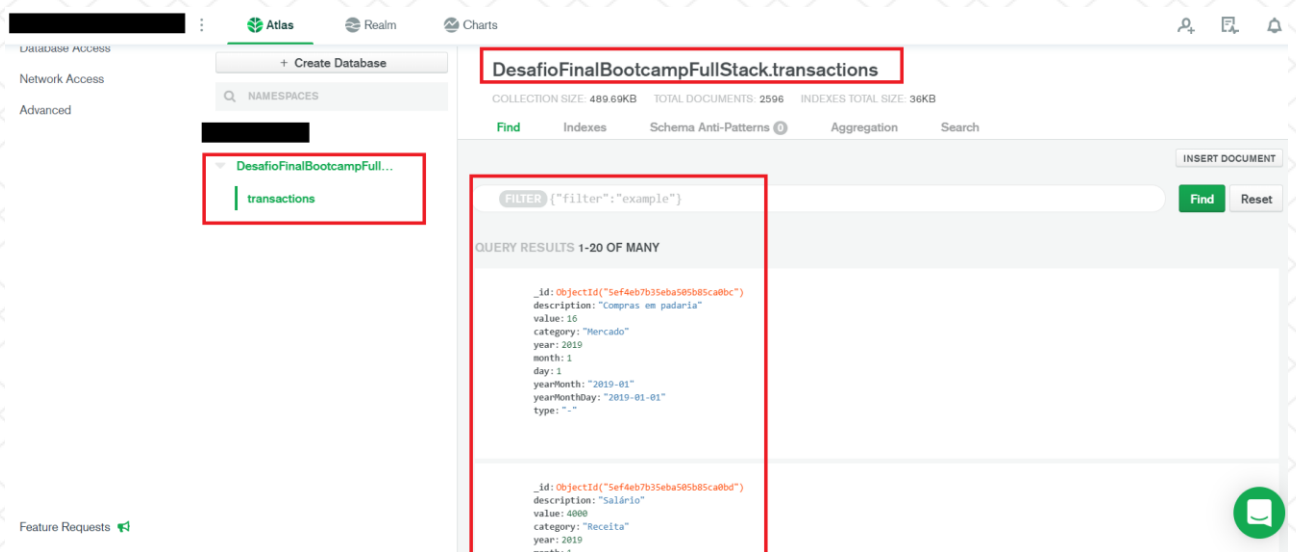
".env.example". Há um comentário iniciado por # na linha 1 com um exemplo de preenchimento e um template para ser copiado/colado no arquivo ".env". Muito cuidado com o preenchimento dos valores. **Caracteres maiúsculos e minúsculos são considerados!**

```
.env.example
1 # Exemplo: DB_CONNECTION="mongodb+srv://root:123456@rusha458tpl.mongodb.net/DesafioFinalBootcampFullStack?retryWrites=true&w=majority"
2 DB_CONNECTION="mongodb+srv://<usuario>:<senha>@<host_com_final_.mongodb.net/><cluster>?retryWrites=true&w=majority"
```

5. Acesse a pasta do projeto "**mongodb-import**" no seu terminal de comandos e execute o seguinte comando: "**yarn db**". Caso ocorra algum problema, verifique novamente os passos acima. Se tudo correr bem, a seguinte saída aparecerá no terminal e os dados originais do projeto serão copiados para o seu Banco de Dados.

```
λ yarn db
yarn run v1.22.4
$ node populateMongoDb.js
Iniciando conexão ao MongoDB...
Conectado ao MongoDB
Eliminando as collections...
Recriando as collections...
Preenchendo os documentos das collections...
Processamento finalizado!
Done in 11.86s.
```

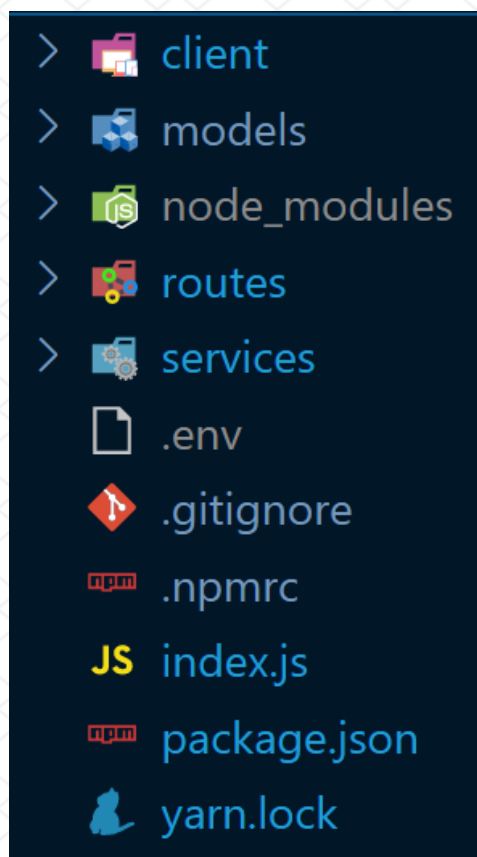
6. Confirme se os dados apareceram em sua conta no **MongoDB Atlas**. Recarregue a tela se necessário.



7. **Observação muito importante:** a execução deste script de importação deve ser feita novamente **antes** do aluno preencher o **Questionário sobre o Desafio Final**, pois serão considerados os **dados originais**. Não modifique os arquivos do projeto "**mongodb-import**", principalmente os de extensão **.json**, senão isso pode te prejudicar na resolução das questões. O arquivo que é utilizado na importação é o "**./official-db/transactionsArray.json**". Caso esse arquivo seja modificado, você pode obter novamente o projeto no "**Fórum de Avisos**" ou então acessar a pasta "**./backup**". Há uma cópia desse arquivo por lá.
8. Os registros importados foram gerados aleatoriamente através de combinações de descrições, categorias e valores. Além disso, ficou definido o período como: todos os meses do ano anterior, ano atual e ano posterior, totalizando ao todo 36 períodos (atualmente: de **jan/2019** até **dez/2021**, inclusive). Devo disponibilizar o código-fonte deste script para vocês no fórum durante o andamento do Módulo.
9. **Observação importante:** na geração do arquivo original, houve uma pequena falha e alguns registros com a descrição "**Receita xyz...**" foram criados como **despesa**. Vocês devem considerar que as **receitas** são os registros com **type === '+'** e as **despesas** são os registros com o **type === '-'**. Desconsiderem a descrição. Na aula interativa demonstrarei o bug. Preferi manter o arquivo original mesmo assim para evitar algum tipo de conflito no Desafio Final.
10. Fim da etapa 1!

## **Etapa 2 – Implementação do Back End:**

1. Recomendo utilizar como base o projeto **app-vazio**, que será disponibilizado no **Fórum de Avisos**. Nos itens a seguir, falarei mais sobre esse projeto.
2. Acesse a pasta **app-vazio** após a extração em alguma pasta no seu computador. Renomeie a pasta para **app**.
3. Acesse a pasta **app** através de um terminal de comandos e digite o comando **yarn** para instalar as dependências.
4. Verifique, estude e entenda as pastas e arquivos já contidos neste projeto:



- Pasta **client**: contém o projeto React (Front End). É o mesmo "react-projeto-base", disponibilizado durante o Módulo 03 do Bootcamp. Mais detalhes sobre a implementação com React serão vistos na **Etapa 3**.
- Pasta **models**: contém o arquivo **TransactionModel.js**, referente ao **Schema Mongoose** para a Collection **transaction**. **Este arquivo já foi implementado pelo professor**. Entretanto, nada impede o aluno de modificá-lo e adaptá-lo às suas necessidades.
- Pasta **routes**: contém o arquivo **routes.js** e deve conter as rotas, que deverão ser implementadas pelo aluno.
- Pasta **services**: contém o arquivo **transactionService.js** e deve conter a persistência de dados com o **MongoDB**, a ser também implementado pelo aluno.
- Arquivo **.env**: **deverá ser criado pelo aluno** e conter a String de conexão ao MongoDB **DB\_CONNECTION**, assim como foi feito na etapa anterior.



- Arquivo **.npmrc**: foi **criado pelo professor** e contém uma configuração do NPM, que faz com que as dependências sejam instaladas em sua versão **exata (save-exact)**. Isso garante, em regra, mais estabilidade de apps em produção.
- Arquivo **index.js**: é onde tudo começa. Já há uma implementação de **configurações** do **express** e da **conexão** com o **MongoDB**. Basta garantir que a String **DB\_CONNECTION** esteja devidamente preenchida no arquivo **.env**.

5. Verifique, estude e entenda mais alguns detalhes importantes sobre o arquivo **index.js**:

```
18  /**
19  |  * Vinculando o React ao app
20  |  */
21  app.use(express.static(path.join(__dirname, 'client/build')));
```

Este trecho de código faz com que o express hospede o React de produção (após o build).

```
23  /**
24  |  * Rota raiz
25  |  */
26  app.get('/api/', (_, response) => {
27  |  response.send({
28  |  |  message:
29  |  |  |  'Bem-vindo à API de lançamentos. Acesse /transaction e siga as orientações',
30  |  |  });
31  |  });
32
33  /**
34  |  * Rotas principais do app
35  |  */
36  app.use('/api/transaction', routes);
```

Este trecho de código faz com que a API principal do Back End (transaction) fique hospedada em **http://meu\_site\_no\_heroku.herokuapp.com/api/transaction**

```

64  /**
65  ** * Definição de porta e
66  ** * inicialização do app
67  ** */
68  const APP_PORT = process.env.PORT || 3001;
69  app.listen(APP_PORT, () => {
70    console.log(`Servidor iniciado na porta ${APP_PORT}`);
71  });
72  });
73

```

Neste trecho de código percebe-se que foi priorizado o valor de **process.env.PORT** para ser utilizado como porta do servidor de Back End. Isso será utilizado pelo **Heroku** em **produção**. Em **desenvolvimento**, será adotada a porta **3001**, pois não há a variável **PORT** no arquivo **.env local**.

- Verifique, estude e entenda alguns detalhes importantes sobre o arquivo **package.json**:

```

16  "scripts": {
17    "server": "nodemon index.js",
18    "start": "node index.js",
19    "heroku-postbuild": "cd client && npm install && npm run build"

```

Para executar o servidor local, digite **yarn server**, que irá utilizar o Nodemon para tal.

O script **heroku-postbuild** será utilizado pelo **Heroku CLI** para realizar o **build** da aplicação **React**. Mais detalhes serão vistos na documentação da **Etapa 4**.

```

21  "nodemonConfig": {
22    "ignore": [
23      "client/"
24    ]

```

Configuração importante do **Nodemon** para que ele não "escute" a pasta do React, pois ela já tem um servidor próprio de desenvolvimento. Sem isso, qualquer alteração no projeto React acarreta no reinício do servidor de Back End durante o desenvolvimento, o que não é desejável.



```

26     "engines": {
27       "node": "12.9.1"
28     }

```

Configuração importante do **Heroku** para que ele utilize a mesma versão do Node.js de desenvolvimento em produção. Se a sua versão do Node.js for **diferente** de **12.9.1**, faça a devida alteração nesse objeto de package.json e informe a versão que você está utilizando. Recomenda-se, entretanto, utilizar a versão **12.9.1**. Ela foi homologada por mim durante o desenvolvimento deste projeto.

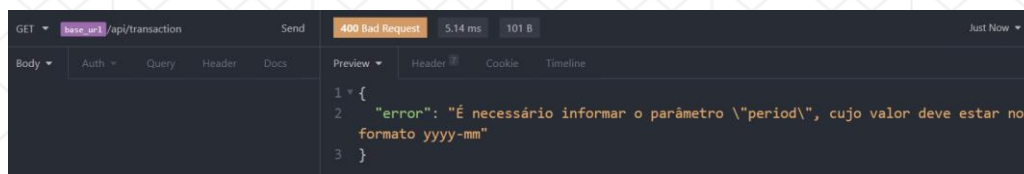
7. Ainda na pasta raiz do projeto, execute **yarn server** para inicializar o **Back End** de **desenvolvimento**. Caso ocorra algum problema, verifique novamente os passos acima. Se tudo correr bem, serão exibidas as seguintes mensagens no console:

```

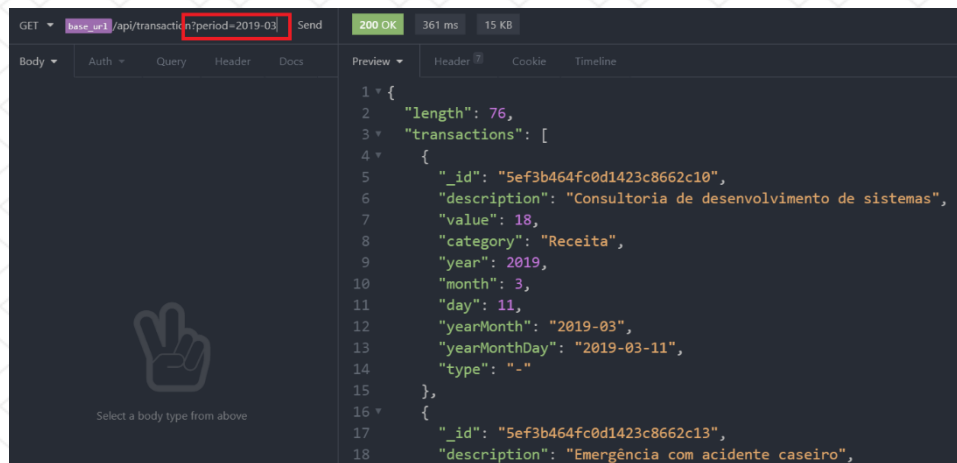
[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Iniciando conexão ao MongoDB...
Conectado ao MongoDB
Servidor iniciado na porta 3001

```

8. **Observação muito importante:** o **GET** de **transaction** deve considerar **obrigatoriamente** o **período (ano-mês)** com base no campo **yearMonth**. Ou seja, o **período deve ser obrigatoriamente informado** nesse tipo de **rota**. Isso deve ser implementado pelo **aluno** no arquivo **routes.js**.



Requisição **incorreta**, pois não foi informado o parâmetro **period**.



Requisição **correta**, pois foi informado o parâmetro **period**.

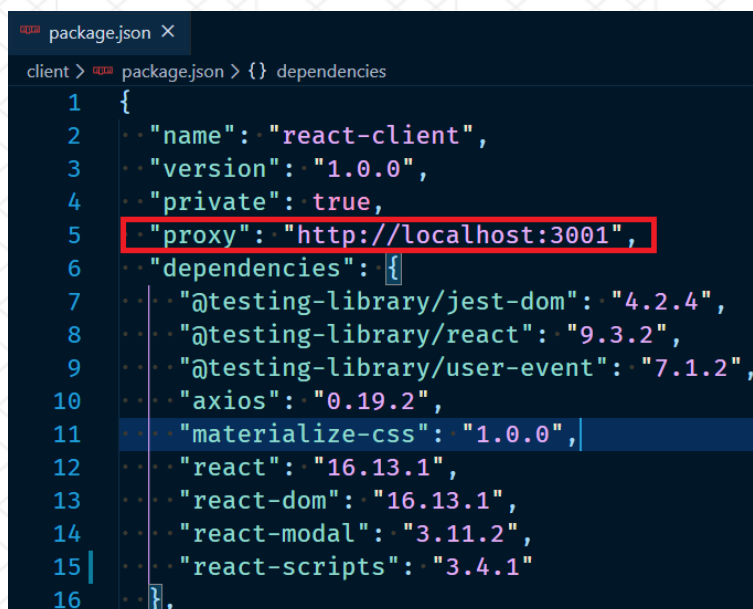
9. Agora é com você, aluno(a)! Faça as implementações pendentes em **routes.js** e **transactionService.js** com base no conteúdo visto nos Módulos 02 e 04 do Bootcamp. Sugiro também testar as **rotas** com o **Insomnia** e verificar se todo o **CRUD** está sendo de fato refletido no **MongoDB**. Fique à vontade para implementar o código da maneira que preferir. **Observação importante: considere que todos os campos de "transaction" são obrigatórios!**

10. Fim da Etapa 2!

### **Etapa 3 – Implementação do Front End com React:**

1. O Front End deve ser implementado a partir da pasta "**client**". Lá se encontram os mesmos arquivos do "**\_react-projeto-base**", que disponibilizei durante o **Módulo 03**.
2. Fica a critério do aluno a definição da **interface**. As telas a seguir são somente uma sugestão de interface que foi implementada pelo professor.
3. Acesse a pasta **client** e digite **yarn** para instalar as dependências do projeto que são, além dos pacotes relacionados ao **create-react-app**:
  - a. Pacote **materialize-css**.
  - b. Pacote **axios**.
  - c. Pacote **react-modal**.

4. O arquivo **package.json** também foi modificado pelo professor, conforme imagem abaixo. Isso é importante para os casos onde tanto o Back End quanto o Front End estão hospedados no mesmo servidor, que é o caso deste projeto. Mais informações podem ser encontradas [aqui](#).



```

package.json X
client > package.json > {} dependencies
1 {
2   "name": "react-client",
3   "version": "1.0.0",
4   "private": true,
5   "proxy": "http://localhost:3001",
6   "dependencies": {
7     "@testing-library/jest-dom": "4.2.4",
8     "@testing-library/react": "9.3.2",
9     "@testing-library/user-event": "7.1.2",
10    "axios": "0.19.2",
11    "materialize-css": "1.0.0",
12    "react": "16.13.1",
13    "react-dom": "16.13.1",
14    "react-modal": "3.11.2",
15    "react-scripts": "3.4.1"
16  },

```

5. A implementação é **obrigatória** para as seguintes **funcionalidades** no Front End, pois elas serão testadas no questionário de entrega do projeto do Desafio Final.
  - a. **Navegação** de transações **agrupadas por mês/ano** (semelhante às rotas do Back End).
  - b. **Filtro simples** a partir da **descrição da transação**. **Acentos e caracteres especiais devem ser considerados no filtro.**
  - c. Resumo com a **quantidade de lançamentos, somatório** de valores de **receita, despesa e saldo agrupados por mês/ano**.
6. Algumas telas **sugeridas** para a aplicação Front End:



## Bootcamp Full Stack - Desafio Final

### Controle Financeiro Pessoal

<

Jul/2020

>

Lançamentos: 69

Receitas: R\$ 4.000,00

Despesas: R\$ 1.710,00

Saldo: R\$ 2.290,00

+ NOVO LANÇAMENTO

Filtro

01	Lazer	Viagem para a praia	R\$ 10,00		
01	Transporte	Táxi para o aeroporto	R\$ 26,00		
01	Saúde	Internação em hospital	R\$ 33,00		
01	Receita	Salário	R\$ 4.000,00		

Estado inicial da tela (mês corrente).

### Controle Financeiro Pessoal

<

Jul/2020

>

Receitas: R\$ 4.000,00

Despesas: -R\$ 1.710,00

Nov/2019

Dez/2019

Jan/2020

Fev/2020

Mar/2020

Abr/2020

Mai/2020

Jun/2020

Jul/2020

Ago/2020

Set/2020

Out/2020

Nov/2020

Dez/2020

Jan/2021

Fev/2021

Mar/2021

Abr/2021

Mai/2021

Jun/2021

Jul/2021

Seleção de períodos através de <select>.

## Bootcamp Full Stack - Desafio Final

Edição de lançamento

☒ Despesa
 ☐ Receita

Descrição:  
 Viagem para a praia

Categoria:  
 Lazer

Valor:  
 10

01/07/2020

SALVAR

Edição de lançamentos, que **não** permite a troca do tipo (receita/despesa).

## Bootcamp Full Stack - Desafio Final

Inclusão de lançamento

☒ Despesa
 ☐ Receita

Descrição:  
 \_\_\_\_\_

Categoria:  
 \_\_\_\_\_

Valor:  
 0

01/07/2020

SALVAR

Inclusão de lançamentos.

## Bootcamp Full Stack - Desafio Final

### Controle Financeiro Pessoal

<

Jul/2020

>

Lançamentos: 7







Receitas: R\$ 0,00

Despesas: -R\$ 190,00

Saldo: -R\$ 190,00

+ NOVO LANÇAMENTO

rest

02	Lazer Almoço em restaurante	R\$ 45,00	 
06	Lazer Almoço em restaurante	R\$ 10,00	 
17	Lazer Almoço em restaurante	R\$ 46,00	 

Filtro de lançamentos a partir da descrição.

- A exclusão de lançamentos pode ser implementada através do simples clique no ícone que representa a "Lixeira", conforme imagem acima.
- Dica:** na minha implementação utilizei a tag `<select>` nos períodos, que no **React** tem o funcionamento um pouco **diferente do HTML padrão**. Mais detalhes podem ser vistos [aqui](#). Persistindo dúvidas, não deixem de perguntar no fórum. Demonstrarei, na aula interativa, como ativar o JavaScript do Materialize. Isso pode ser necessário caso o aluno queira o comportamento padrão da tag `<select>`. Como contorno, basta acrescentar a classe **browser-default** que o Materialize irá ignorar a tag.
- Fim da Etapa 3!



#### **Etapa 4 – Inclusão do código-fonte no GitHub e implantação no Heroku:**

1. Crie um repositório no GitHub e hospede o app por lá. Siga as instruções vistas durante o Módulo 04.
2. Quanto à implantação, certifique-se de que você já possui uma conta no **Heroku** e que a ferramenta [Heroku CLI](#) esteja devidamente instalada em seu computador. Teste com o seguinte comando: **heroku -v**

```
C:\projetos
λ heroku -v
heroku-cli/6.15.22-3f1c4bd (win32-x64) node-v9.3.0
```

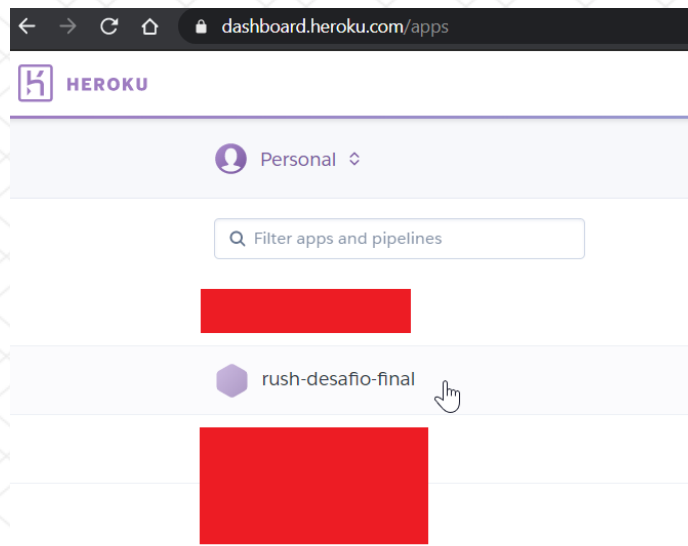
3. Acesse a pasta **app** e faça o login no **Heroku** com o comando **heroku login**.  
**Atenção:** por algum motivo que desconheço, a tela de login do **Heroku** não ficou bem visível no **Cmder** do **meu computador com Windows 10**. Se acontecer com você, faça o login no prompt de comando padrão do Windows.

```
λ heroku login
Enter your Heroku credentials:
Email: [REDACTED]
Password: [REDACTED]
Logged in as [REDACTED]
```

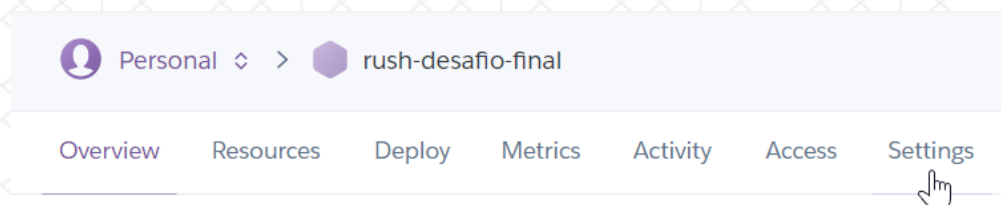
4. Crie um projeto no Heroku com o comando **heroku create nome\_do\_projeto**. Sugiro "**identificador-pessoal-desafio-final**". O nome do projeto deve ser **único** e possuir, no máximo, 30 caracteres. Caso não se importe com o nome, digite apenas **heroku create** que o próprio **Heroku** define um nome único para você.

```
λ heroku create rush-desafio-final
Creating ● rush-desafio-final... done
https://rush-desafio-final.herokuapp.com/ | https://git.heroku.com/rush-desafio-final.git
```

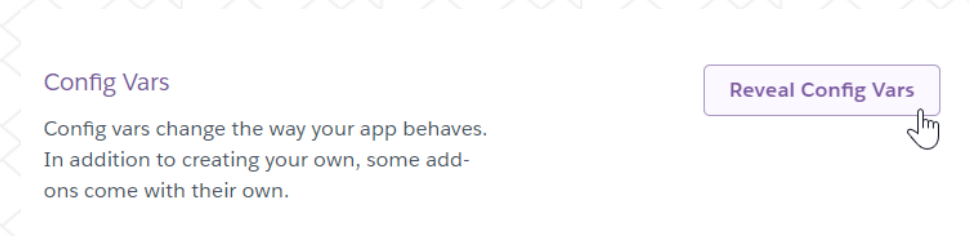
5. Acesse o site do Heroku e inclua a variável de ambiente **DB\_CONNECTION**, que foi definida no arquivo **.env**, seguindo as imagens abaixo em sequência:



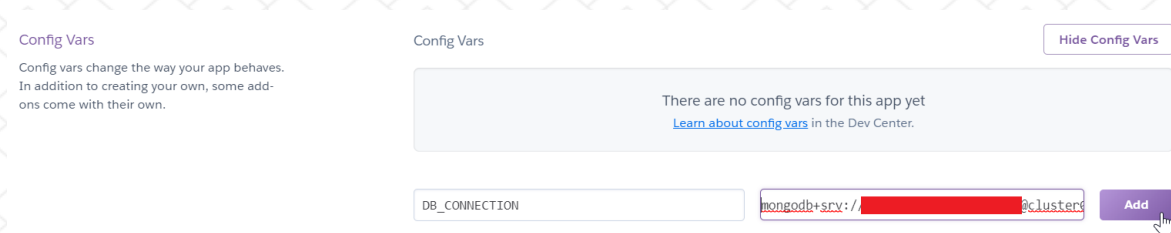
Acesse o seu projeto recém-criado.



Acesse "Settings".



Acesse "Reveal Config Vars".



Preencha DB\_CONNECTION e a String de conexão (sem aspas duplas) e clique em "Add".

6. Volte ao terminal e digite os seguintes comandos, conforme a ordem abaixo. Verifique a imagem abaixo com a saída de cada comando (alguns deles estão resumidos). Esses comandos se referem à implantação no **Heroku**, que pode ser feita via **git**.
- a. `git init`
  - b. `heroku git:remote -a "nome-do-seu-projeto"`
  - c. `git add .`
  - d. `git commit -m "heroku"`
  - e. `git push heroku master`



```
λ git init
Initialized empty Git repository in [REDACTED]

λ heroku git:remote -a rush-desafio-final
set git remote heroku to https://git.heroku.com/rush-desafio-final.git

λ git add .

λ git commit -m "heroku"
[master (root-commit) f7a2228] heroku
22 files changed, 12492 insertions(+)
create mode 100644 .gitignore
create mode 100644 .npmrc
create mode 100644 client/.gitignore
create mode 100644 client/.npmrc
create mode 100644 client/README.md
create mode 100644 client/package.json
create mode 100644 client/public/favicon.ico
create mode 100644 client/public/index.html
create mode 100644 client/public/logo192.png
create mode 100644 client/public/logo512.png
create mode 100644 client/public/manifest.json
create mode 100644 client/public/robots.txt

...

λ git push heroku master
Counting objects: 29, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (26/26), done.
Writing objects: 100% (29/29), 221.19 KiB | 8.19 MiB/s, done.
Total 29 (delta 1), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Node.js app detected
remote:
remote: -----> Creating runtime environment

...

remote:      Released v4
remote:      https://rush-desafio-final.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/rush-desafio-final.git
* [new branch]      master -> master
```

7. Se tudo deu certo, o app estará disponível conforme imagem abaixo.



## Desafio Final do Bootcamp Full Stack

Neste exemplo, o app ainda está "vazio".

A API pode ser acesada em: <http://nome-do-projeto.herokuapp.com/api/>.

8. Caso ocorra algum erro, verifique os passos anteriores. Caso persista o erro, volte ao terminal e digite **heroku logs --tail**. Digite Ctrl + C para cancelar a visualização do log. Esse log é muito importante para vocês postarem no fórum caso precisem de nosso apoio, por exemplo.
9. Caso efetuem alguma modificação no código-fonte, façam tanto o **push** no **Github** quanto o **push** no **Heroku**. Para o **push** no **Heroku** digite, após o commit, **git push heroku master**.
10. O intuito deste extenso tutorial foi de evitar todos os problemas que enfrentei ao configurar o **MongoDB** e também ao publicar o app no **Heroku**. Tentei deixar o mais detalhado possível para que vocês, alunos, foquem mais na **implementação** deste desafio.
11. Fim da Etapa 4!