AVALIAÇÃO – DESENVOLVEDOR DE AUTOMAÇÃO
JULIANNY ALVES DA SILVA

1. Os seletores evidenciados são:

1_Membros: a.js-change-card-members

```
<div class=”window-sidebar”>
        <div class=”window-module”>
                <div class=”u-clearfix”>
                        <a class=”button-link js-change-card-members”></a>
```

2_Etiquetas: a.js-edit-labels

```
<div class=”window-sidebar”>
        <div class=”window-module”>
                <div class=”u-clearfix”>
                        <a class=”button-link js-edit-labels”></a>
```

3_Checklist: a.js-add-checklist-menu

```
<div class=”window-sidebar”>
        <div class=”window-module”>
                <div class=”u-clearfix”>
                        <a class=”button-link js-add-checklist-menu”></a>
```

4_Mover: a.js-move-card

```
<div class=”window-sidebar”>
        <div class=”window-module”>
                <div class=”u-clearfix”>
                        <a class=”button-link js-move-card”></a>
```

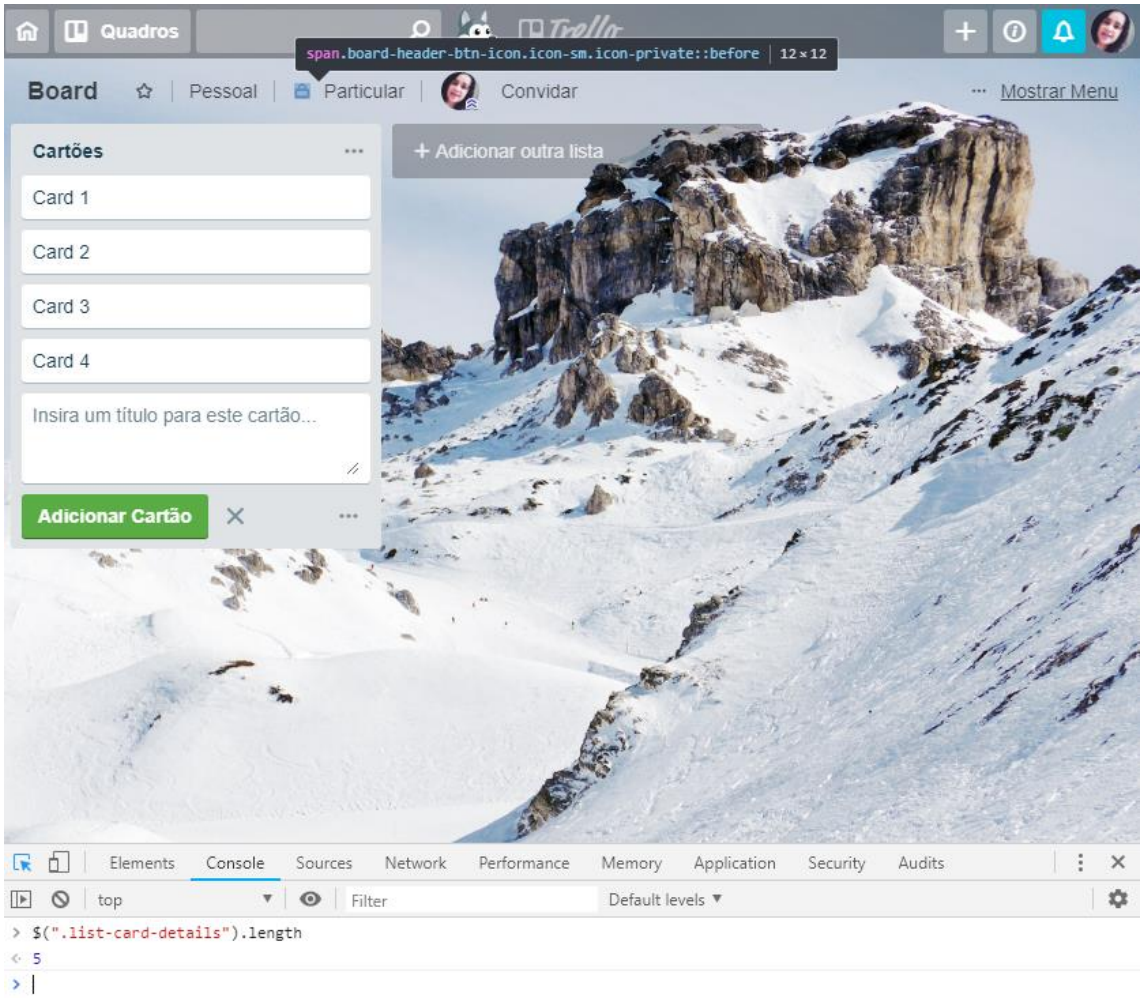
2. Desenvolver um projeto de automação de testes demanda muito planejamento das atividades que serão executadas para se obter o resultado esperado, e principalmente modelagem do projeto para que consiga obter a cobertura e qualidade necessária para agregar valor ao produto de software. Logo é importante o desenvolvedor de testes conhecer bem a arquitetura do sistema que vai testar, quais soluções estão sendo aplicadas, como tais manipulam os componentes e dados do sistema. E se tratando de um sistema web, que em tempo de execução os objetos sofrem em seu ciclo de vida, alterações de estados e manipulações, é importante medir o desempenho e tempo de renderização dos objetos. Pois uma vez com tais métricas é possível implementar casos de testes que verifica se a ordem de renderização esta correta, o tempo gasto está nos limites esperados, e as verificações dos scripts de testes sejam escritos de forma que de acordo com as interações tenham tratamentos de controle de espera(wait();), e não sofram interrupções e instabilidades.

3. cardsStatusFeito.json criado:

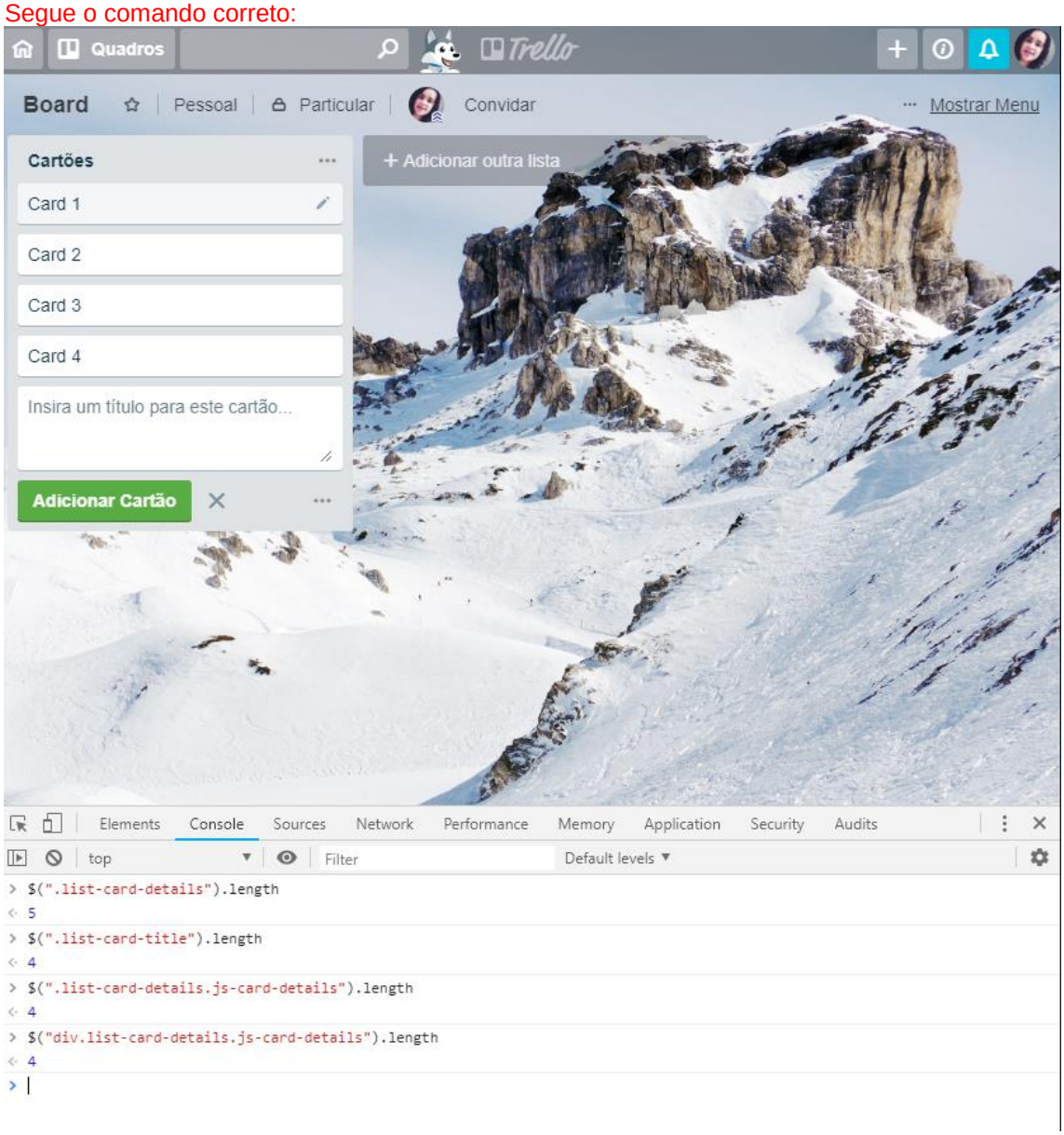
```
[
 {
   "Titulo": "Documentar REQ01",
   "Descricao": "Escrever documentacao do Requisito REQ01",
   "Data Entrega": "01/02/2019",
   "Tags": ["gestao", "req01"]
 },
 {
   "Titulo": "Documentar REQ02",
   "Descricao": "Escrever documentacao do Requisito REQ02",
   "Data Entrega": "01/02/2019",
   "Tags": ["gestao", "req02"]
 },
 {
```

```
  "Titulo": "Desenvolver REQ01",
  "Descricao": "Desenvolver Requisito REQ01",
  "Data Entrega": "07/02/2019",
  "Tags": ["dev", "req01", "qa-aprovado"]
 },
 {
  "Titulo": "Desenvolver REQ02",
  "Descricao": "Desenvolver Requisito REQ02",
  "Data Entrega": "12/02/2019",
  "Tags": ["dev", "req02", "qa-reprovado"]
 }
]
```

4. Segue o quadro criado com os 4 cartões:



5. Segue o comando correto:



6. A ferramenta utilizada é o cypress.io: https://www.cypress.io/

O projeto encontra armazenado no GitHub: https://github.com/juliannyas/redmine_test

Para executar os testes basta seguir as orientações do readme do projeto: https://github.com/juliannyas/redmine_test/blob/master/README.md

Ou no arquivo .zip em anexo.

Instalar as dependências do projeto:

npm install

Executar a suite de testes:

npm run cypress:open