Avaliação para automação de teste

FRONTEND:

- Desenvolver uma arquitetura consistente para **automação WEB**.
- Poderá ser feito em qualquer linguagem da sua escolha (Java, JS, C#, Ruby...).

Observação: Nossos projetos estão sendo desenvolvidos na linguagem Ruby.

- Seguir boas práticas de automação/desenvolvimento
  I. CleanCode.
  II. PageObjects.
  III. Arquitetura BDD com Gherkin.
  IV. Readme atualizado.
  V. Massa de dados dinâmica.

Sprint 1:

Story User: Cadastrar produtos no carrinho

Como futuro cliente da Livelo

Quero poder adicionar produtos no carrinho

De modo que possa usufruir após meus acessos.

Critério de Aceite:

- Deverá adicionar 3 produtos diferentes no carrinho (Caminho utilizando CEP)
  I. Não precisar ter cadastro.
  II. Deverá conter cenários em Gherkin (Arquitetura BDD).
- Deverá conter validação de **tipo**, **nome**, e **quantidade de pontos**.
- Deverá conter 1 cenário para remover produto do carrinho, e garantir que o mesmo foi retirado do carrinho.
- Deverá validar “carrinho vazio”
- Gerar relatório HTML.

Ambiente para desenvolvimento:

- https://www.livelo.com.br/home

BACKEND:

- Desenvolver uma arquitetura consistente para **automação BackEnd (APIs)**.
- Poderá ser feito em qualquer linguagem da sua escolha (Java, JS, C#, Ruby...)
  - Diferencial RUBY.

Observação: Nossos projetos estão sendo desenvolvidos na linguagem Ruby.

- ✓ **Dica:**
  I. Caso utilize a linguagem Ruby utilizar GEM HTTPARTY.
  II. Caso utiliza a linguagem Java utilizar Rest-Assurend.
  III. Caso utilize Java Script utilizar a arquitetura Cypress.

- Seguir boas práticas de automação/desenvolvimento
  I. CleanCode.
  II. PageObjects.
  III. Arquitetura BDD com Gherkin.
  IV. Readme atualizado.
  V. Massa de dados dinâmica.

Sprint 2:

Story User: Validar baralho

Como jogador de baralho

Quero poder validar meu Deck de cartas

De modo que possa usufruir durante meus jogos.

Critério de Aceite:

- Deverá conter validações de chamada dos métodos GET e POST
- Deverá conter as seguintes validações:
  i) Status.
  ii) Tempo de execução.
  iii) Validação de campos (Body) e seus retornos (dinâmico).
  iv) Validações dinâmicas para chamadas sequenciais.
  v) Apresentar 1 cenário de falha
  vi) Gerar relatório Json e HTML

Ambiente para desenvolvimento:

GET - https://deckofcardsapi.com/api/deck/new/shuffle/?deck_count=1

POST - https://deckofcardsapi.com/api/deck/<INFORMARIDDECK/draw/?count=2