notas:

* estabelecer qual versão do navegador do cypress (manter versão);
  + **Cypress: 13.17.0**
  + **Browser: Electron 118 (headless)**
  + **Node Version: v22.11.0**
* ampliar testes com mais cruds (pelo menos mais 20, total de 30);
  + podendo ser feito combinações (login + criar artigo)
* gerar gráficos de todos os testes;

OBS: Até o momento os testes foram feitos na implementação de react + rails.

1- Foi criando scripts de teste utilizando a biblioteca Cypress, [link para os scripts](https://drive.google.com/drive/folders/142qAr3zSaLAavu1DSRxRD03R9Z18jjBE?usp=drive_link).

* [Lista de testes](https://docs.google.com/document/d/1pZioJbDee5TzNq7CY1pjVEK8GCMGRT2AGsi3OQJKUQ8/edit?usp=sharing)

2 - Foi criado scripts em python utilizando Codecarbon para realizar as medições durante a execução dos testes Cypress, [link para os scripts](https://drive.google.com/drive/folders/1q-O7q8MPvXvBAcI9clVoJwX2WQ-nVAAz?usp=drive_link).

* codeCarbon\_cypress - responsável por executar o codecarbon e medir o teste cypress especificado;
* codeCarbon\_time - responsável por executar o codecarbon por um terterminado tempo;

3 - Para executar os testes o computador foi reiniciado e foi esperado 3 minutos, a fim de esperar o boot completo, após esse tempo os servidores Ruby e React foram inicializados, uma vez os servidores totalmente inicializados o script **codeCarbon\_cypress** foi executado, cada um dos testes cypress foi **executado 10 vezes**, e o script gera um log no seguinte formato:

* O nome do arquivo é codeCarbon\_cypress\_log\_[data e hora da execução do teste].log
* Processos em Execução - é uma lista de todos os processos que estavam em execução durante a inicialização dos testes. Podemos comparar a lista de processos entre os testes para entender possíveis interferências;
* Cypress script - informa qual foi o teste cypress executado;
* Dados do CodeCarbon - informa os dados gerais do codeCarbon;
* Log do CodeCarbon - informa todo o log do codeCarbon com dados importantes, como o consumo das CPUs e consumo em kwh, a amostragem é a cada 2 segundos;

4 - Após a execução do script que mede o consumo durante a execução do Cypress, é executado o script codeCarbon\_time, onde é passado o mesmo tempo da execução do teste cypress em segundos, a fim de compararmos o computador sem executar nada com a execução dos testes. Esse script também gera um log no seguinte formato:

* O nome do arquivo é codeCarbon\_cypress\_log\_[data e hora da execução do teste].log
* Processos em Execução - é uma lista de todos os processos que estavam em execução durante a inicialização dos testes. Podemos comparar a lista de processos entre os testes para entender possíveis interferências;
* Dados do CodeCarbon - informa os dados gerais do codeCarbon;
* Log do CodeCarbon - informa todo o log do codeCarbon com dados importantes, como o consumo das CPUs e consumo em kwh, a amostragem é a cada 2 segundos;

5 - Os logs de todos os testes podem ser encontrados nesse [link](https://drive.google.com/drive/folders/19O1MkkcqFc2-2ydDBpjQUQacGi-_5ynO?usp=sharing).

6 - Fazendo uma análise simples das médias de CPU power em cada teste, [link da planilha](https://docs.google.com/spreadsheets/d/1StPB_m5NfSxXFTo5RM5elzFOkbmBLHEWtmvJ9QKPQQU/edit?usp=sharing), podemos perceber uma diferença significativa entre a execução dos **testes da aplicação, consumo de cerca de 22w, e da máquina sem executar nenhuma aplicação, cerca de 11w.** Dentre as funcionalidades específicas, a que mais consumiu foi o **login de um usuário previamente cadastrado, média de 23,9w, e a que menos consumiu foi a de criação de um artigo, cerca de 21w.**