Projeto ACCB - GUI

Projeto desenvolvido para o projeto de iniciação tecnológica

Progresso de Desenvolvimento

- [x] Interface
- [x] Banco de Dados
- [x] Log de Erros
- [x] CRUD de Componentes
- [x] Backup de Pesquisa
- [x] Configuração de Estabelecimentos
- [x] Configuração de Produtos
- [x] Log de Produtos
- [x] Melhoria de reconhecimento de produto
- [x] Listagem de Pesquisas
- [x] Pesquisa de Pesquisas
- [x] Filtragem de Pesquisas
- [x] Transformar csv_to_xlsx em db_to_xlsx
- [x] Limitar à somente uma aba de navegação
- [x] Fechar o programa caso atualizada a página e não seja o prório programa.
- [x] Checar se o chrome está instalado
- [x] Popup de backup de pesquisa anterior
- [ ] Botão para iniciar a partir de um backup caso a pessoa ja tenha fechado o popup anterior.

Informações de Desenvolvimento

Python

Para rodar o projeto preferencialmente inicie um ambiente virtual com :

```
    pip install virtualenv
    python -m venv <nome>
```

Em seguida abra a pasta do ambiente e clone o repositório em questão com :

```
    cd <nome>
    git clone https://github.com/smvasconcelos/ACCB_IT.git --single-branch --
branch desktop-web
```

E por último inicie o ambiente virtual e instale as dependências do python para iniciar o projeto :

```
    cd Scripts
    activate.bat
    cd ..
    pip install -r requirements.txt
```

Conda

Para rodar o projeto preferencialmente inicie um ambiente virtual com :

```
    conda create --name ACCB
```

Em seguida clone o repositório em questão com :

```
    git clone https://github.com/smvasconcelos/ACCB_IT.git --single-branch --
branch desktop-web
```

E por último inicie o ambiente virtual e instale as dependências do python para iniciar o projeto :

```
    conda activate ACCB
    pip install -r requirements.txt
```

Agora é só rodar o projeto com python -m flask run ou python app.py.

Observações

Para que seja possível gerar um exe sem uma janela de console do windows é necessário alterar um arquivo fonte do selenium, este que se encontra em :

```
    \Lib\site-packages\selenium\webdriver\common\service.py
    Altere então :
    self.process = subprocess.Popen(cmd, env=self.env,
close_fds=platform.system() != 'Windows', stdout=self.log_file,
stderr=self.log_file, stdin=PIPE)
    para :
    self.process = subprocess.Popen(cmd, stdin=PIPE, stdout=PIPE ,stderr=PIPE,
shell=False, creationflags=0x08000000)
```

E em seguida é só executar o comando pyinstaller app.spec.

Criando um .spec Novo

```
pyi-makespec  --noconsole --onefile app.py
```

.spec Completo :

```
pyi-makespec --noconsole --onefile --add-data="templates;templates" --add-
data="static;static" --add-data="schema.sql;." --add-data="itabuna.json;." --
add-data="ilheus.json;." --name="ACCB" --icon=logo.ico --
paths="E:\Uesc\Scrapper\web\ACCB\Lib\flask_material\templates\material" --
hidden-import=engineio.async_drivers.eventlet --hidden-import=flask_material --
uac-admin --additional-hooks-dir=. app.py
```

Gerando nova documentação com pdoc

```
    pdoc app.py database scrapper -o doc
```

Caso necessário instalar a ultima versão do pyinstaller

```
- Neste caso existem algumas ferramentas necessárias para realizar esta etapa :
https://pyinstaller.readthedocs.io/en/stable/bootloader-building.html#build-
using-cygwin-and-mingw

- Eu recomendo essa utilizar o compilador para C mingw64 e não 32 bits que pode
ser baixado no link : https://sourceforge.net/projects/mingw-
w64/files/Toolchains%20targetting%20Win32/Personal%20Builds/mingw-
builds/installer/mingw-w64-install.exe/download, não esqueça de mudar a
arquitetura para x86_64

- git clone https://github.com/pyinstaller/pyinstaller

- cd pyinstaller

- cd bootloader

- python ./waf distclean all - builda o bootloader para o sistema em questão.

- cd ..

- python setup.py install - instala o pyinstaller no ambiente ativo no momento
```