

scraper_gui.py

```
# Import der GUI-Bibliothek
from tkinter import *

# Import der Sprachpakete
import language_packages.lp_en as lp_en
import language_packages.lp_de as lp_de
import language_packages.lp_fr as lp_fr

# Import der Scraper
from scraper import *

root = Tk()

# Standwerte
language_pack = lp_en

error_input_empty = ""
error_input_contains_numbers = ""
error_input_contains_special_characters = ""
booking_url = language_pack.url_booking
scraping_in_progress = ""

order_question = language_pack.order_question
sort_alphabet = language_pack.sort_alphabet
sort_price = language_pack.sort_price
no_sort = language_pack.no_sort

storage_question = language_pack.storage_question
generate_file = language_pack.generate_file
generate_script = language_pack.generate_script
generate_all = language_pack.generate_all
upload_to_db = language_pack.db_upload

db_upload_question = language_pack.db_upload_question
db_host_input = language_pack.db_host_input_label
db_user_input = language_pack.db_user_input_label
db_password_input = language_pack.db_password_input_label
```

```
db_name_input = language_pack.db_name_input_label
db_send_data = language_pack.send_data

process_done = language_pack.process_done
restart = language_pack.restart

def start():
    # Stellt die Sprache um
    def select_language(lp):
        global language_pack

        global error_input_empty
        global error_input_contains_numbers
        global error_input_contains_special_characters
        global booking_url
        global scraping_in_progress

        global order_question
        global sort_alphabet
        global sort_price
        global no_sort

        global storage_question
        global generate_file
        global generate_script
        global generate_all
        global upload_to_db

        global db_upload_question
        global db_host_input
        global db_user_input
        global db_password_input
        global db_name_input
        global db_send_data

        global process_done
        global restart

        if lp == "EN":
            language_pack = lp_en
```

```
elif lp == "DE":
    language_pack = lp_de
elif lp == "FR":
    language_pack = lp_fr

sel_lang.config(text = language_pack.selector)
label.config(text = language_pack.selection)
go_next_to_second_page_button.config(text = language_pack.next)

go_back_to_start_page_button.config(text = language_pack.back)
scrape_button.config(text = language_pack.start_scraping)
location_question.config(text = language_pack.location_question)
error_input_empty = language_pack.error_input_empty
error_input_contains_numbers = language_pack.error_input_empty
error_input_contains_special_characters =
language_pack.error_input_contains_special_characters
booking_url = language_pack.url_booking
scraping_in_progress = language_pack.scraping_in_progress

order_question = language_pack.order_question
sort_alphabet = language_pack.sort_alphabet
sort_price = language_pack.sort_price
no_sort = language_pack.no_sort

storage_question = language_pack.storage_question
generate_file = language_pack.generate_file
generate_script = language_pack.generate_script
generate_all = language_pack.generate_all
upload_to_db = language_pack.db_upload

db_upload_question = language_pack.db_upload_question
db_host_input = language_pack.db_host_input_label
db_user_input = language_pack.db_user_input_label
db_password_input = language_pack.db_password_input_label
db_name_input = language_pack.db_name_input_label
db_send_data = language_pack.send_data

process_done = language_pack.process_done
restart = language_pack.restart
```

```
# Lädt die Startseite
def start_page():
    global language_pack
    language_pack = lp_en
    location_question.destroy()
    location_input.destroy()
    go_back_to_start_page_button.destroy()
    scrape_button.destroy()
    error_label.destroy()

    start()

# Lädt die zweite Seite
def second_page():
    sel_lang.destroy()
    R1.destroy()
    R2.destroy()
    R3.destroy()
    label.destroy()
    go_next_to_second_page_button.destroy()

    location_question.pack()
    location_input.pack()
    error_label.pack()
    scrape_button.pack()
    go_back_to_start_page_button.pack()

# Lädt die dritte Seite
def third_page():
    location_question.destroy()
    location_input.destroy()
    error_label.destroy()
    scrape_button.destroy()
    go_back_to_start_page_button.destroy()

    order_question_label.pack()
    sort_alphabetically_button.pack()
    sort_price_button.pack()
    no_sort_button.pack()
```

```
# Lädt die vierte Seite
def fourth_page():
    order_question_label.destroy()
    sort_alphabetically_button.destroy()
    sort_price_button.destroy()
    no_sort_button.destroy()

    storage_question_label.pack()
    create_csv_button.pack()
    create_sql_button.pack()
    create_both_button.pack()
    upload_to_local_db_button.pack()

# Lädt die fünfte Seite
def fifth_page():
    storage_question_label.destroy()
    create_csv_button.destroy()
    create_sql_button.destroy()
    create_both_button.destroy()
    upload_to_local_db_button.destroy()

    db_upload_question_label.pack()
    db_host_input_label.pack()
    db_host_input_field.pack()
    db_user_input_label.pack()
    db_user_input_field.pack()
    db_password_input_label.pack()
    db_password_input_field.pack()
    db_name_input_label.pack()
    db_name_input_field.pack()
    db_send_data_button.pack()

# Lädt die sechste Seite
def sixth_page():
    storage_question_label.destroy()
    create_csv_button.destroy()
    create_sql_button.destroy()
    create_both_button.destroy()
    upload_to_local_db_button.destroy()
```

```

db_upload_question_label.destroy()
db_host_input_label.destroy()
db_host_input_field.destroy()
db_user_input_label.destroy()
db_user_input_field.destroy()
db_password_input_label.destroy()
db_password_input_field.destroy()
db_name_input_label.destroy()
db_name_input_field.destroy()
db_send_data_button.destroy()

process_done_label.pack()
restart_button.pack()

# Prüft die Eingabe
def check_input():
    global error_input_empty
    global error_input_contains_numbers
    global error_input_contains_special_characters
    def containsNumber(value):
        if True in [char.isdigit() for char in value]:
            return True
        else:
            return False
    def containsSpecialCharacter(value):
        if all(char.isalpha() or char.isspace() for char in value):
            return False
        else:
            return True

    search_term = location_input.get()
    if search_term == "" or search_term.isspace():
        error_label.config(text = error_input_empty)
    elif containsNumber(search_term) == True:
        error_label.config(text = error_input_contains_numbers)
    elif containsSpecialCharacter(search_term) == True:
        error_label.config(text = error_input_contains_special_characters)
    else:
        error_label.config(text = scraping_in_progress)
        scrape_booking(booking_url, search_term)

```

```

        third_page()

def sort_by_abc():
    sort_alphabetically()
    fourth_page()

def sort_by_pricing():
    sort_by_price()
    fourth_page()

def generate_csv():
    create_csv()
    sixth_page()

def generate_sql():
    create_sql()
    sixth_page()

def generate_both():
    create_csv()
    create_sql()
    sixth_page()

def send_data():
    host = db_host_input_field.get()
    user = db_user_input_field.get()
    password = db_password_input_field.get()
    name = db_name_input_field.get()

    upload_to_database(host, user, password, name)
    sixth_page()

# Startet die Anwendung neu
def restart_app():
    process_done_label.destroy()
    restart_button.destroy()
    reset_dataframe()
    start()

```

```

# Elemente der Startseite

sel_lang = Label(text = language_pack.selector)
sel_lang.pack()

R1 = Radiobutton(root, text="English", value=1,
                  command= lambda: select_language("EN"))
R1.pack()

R2 = Radiobutton(root, text="Deutsch", value=2,
                  command= lambda: select_language("DE"))
R2.pack()

R3 = Radiobutton(root, text="Français", value=3,
                  command= lambda: select_language("FR"))
R3.pack()

R1.select()

label = Label(text=language_pack.selection)
label.pack()

go_next_to_second_page_button = Button(root, text=language_pack.next,
command=second_page)
go_next_to_second_page_button.pack()

# Elemente der zweite Seite
location_question = Label(text = language_pack.location_question)
location_input = Entry()
error_label = Label()
scrape_button = Button(text = language_pack.start_scraping,
command=check_input)
go_back_to_start_page_button = Button(root, text = language_pack.back,
command=start_page)

# Elemente der dritte Seite
order_question_label = Label(text = order_question)
sort_alphabetically_button = Button(text = sort_alphabet,
command=sort_by_abc)
sort_price_button = Button(text = sort_price, command=sort_by_pricing)

```

```

no_sort_button = Button(text = no_sort, command=fourth_page)

# Elemente der vierte Seite
storage_question_label = Label(text = storage_question)
create_csv_button = Button(text = generate_file, command=generate_csv)
create_sql_button = Button(text = generate_script, command=generate_sql)
create_both_button = Button(text = generate_all, command=generate_both)
upload_to_local_db_button = Button(text = upload_to_db,
command=fifth_page)

# Elemente der fünfte Seite
db_upload_question_label = Label(text = db_upload_question)
db_host_input_label = Label (text = db_host_input)
db_host_input_field = Entry()
db_user_input_label = Label (text = db_user_input)
db_user_input_field = Entry()
db_password_input_label = Label (text = db_password_input)
db_password_input_field = Entry()
db_name_input_label = Label (text = db_name_input)
db_name_input_field = Entry()
db_send_data_button = Button(text = db_send_data, command=send_data)

# Elemente der letzte Seite
process_done_label = Label(text = process_done)
restart_button = Button(text = restart, command=restart_app)

start()
root.mainloop()

```

scraper.py

```

from selenium import webdriver
from selenium.webdriver.common.by import By
import time
import pandas as pd
from bs4 import BeautifulSoup

```

```

import requests
import language_packages.lp_en as lp_en
import language_packages.lp_de as lp_de
import language_packages.lp_fr as lp_fr
import os
import pymysql
from pymysql import Error

# Globale Variablen
searched_city = " "
dataframe = pd.DataFrame(columns=['name', 'address', 'city', 'country',
'link', 'price', 'image'])

# Durchsucht Booking.com nach Hotels in der Stadt, die als Eingabe
angegeben wurde
def scrape_booking(url, input):
    global dataframe
    global searched_city
    searched_city = input

    # Einrichtung des Selenium WebDriver
    options = webdriver.ChromeOptions()
    options.add_argument("start-maximized")

    options.add_experimental_option('excludeSwitches', ['enable-logging'])
    options.add_experimental_option('useAutomationExtension', False)
    driver = webdriver.Chrome(options=options)

    driver.get(url)

    # Schritte zum Navigieren zur Ergebnisseite
    driver.find_element(by=By.CSS_SELECTOR,
value='#\:Ra9\:').send_keys(input)
    time.sleep(5)
    driver.find_element(by=By.CSS_SELECTOR, value='#b2indexPage >
div.c85f9f100b.cb6c8dd99f > div > div > div > div.dabce2e809 >
div.b8ef7618ca.b2ed5869b1.b78b2b2275.e8efa318be > div > button').click()
    time.sleep(5)
    driver.find_element(by=By.CSS_SELECTOR, value='#indexsearch >
div.hero-banner-searchbox > div > div > form >

```

```

div.ffa9856b86.db27349d3a.cc9bf48a25 > div:nth-child(2) > div > div >
button:nth-child(2)').click()

    time.sleep(5)
    driver.find_element(by=By.CSS_SELECTOR, value='#indexsearch >
div.hero-banner-searchbox > div > div > form >
div.ffa9856b86.db27349d3a.cc9bf48a25 > div:nth-child(2) > div >
div.dec3155a06 > div > nav > div.bc13adc8ab.cd743661d2 > ul >
li:nth-child(2) > button').click()

    time.sleep(5)
    driver.find_element(by=By.CSS_SELECTOR,
value='#flexible-searchboxdatepicker >
div.alb3f50dcd.efdd20a9c1.b2fe1a41c3.c1b465858f.d46673fe81 >
div:nth-child(1) > div > div.alb3f50dcd.alf3ecff04.b2fe1a41c3 >
label:nth-child(2) > span').click()

    time.sleep(5)
    driver.find_element(by=By.CSS_SELECTOR,
value='#flexible-searchboxdatepicker >
div.alb3f50dcd.efdd20a9c1.b2fe1a41c3.c1b465858f.d46673fe81 >
div:nth-child(2) > div > div.alb3f50dcd.alf3ecff04.b2fe1a41c3 >
label:nth-child(1) > span').click()

    time.sleep(5)
    driver.find_element(by=By.CSS_SELECTOR, value='#indexsearch >
div.hero-banner-searchbox > div > div > form >
div.ffa9856b86.db27349d3a.cc9bf48a25 > div:nth-child(4) > button').click()

    time.sleep(5)

# Extrahiert Ergebnisinformationen von der aktiven Seite
def extract_results():
    result_cards = driver.find_elements(by=By.CSS_SELECTOR,
value='div[data-testid="property-card"]')
    new_result = {}

    # Durchläuft die Ergebnisse der aktiven Seite in einer Schleife und
    speichert sie im Datenrahmen
    for result in result_cards:
        name = result.find_element(by=By.CSS_SELECTOR,
value='div[data-testid="title"]').text.strip()
        link =
result.find_element(By.CSS_SELECTOR, 'a[data-testid="title-link"]').get_att
ribute('href')

```

```

    price = result.find_element(By.CSS_SELECTOR,
'[data-testid="price-and-discounted-price"]').text
    image = result.find_element(By.CSS_SELECTOR,
'[data-testid="image"]').get_attribute('src')

    hotel_url = requests.get(link)
    doc = BeautifulSoup(hotel_url.text, 'html.parser')
    get_address = doc.find('span', class_='hp_address_subtitle
js-hp_address_subtitle jq_tooltip').text
    address = get_address.split(', ')

    for element in address:
        numbers_in_string = sum(character.isdigit() for character in
element)
        if numbers_in_string >= 4:
            city = element.strip()

    if address[0] == city:
        if url == lp_en.url_booking:
            street = "Not available"
        elif url == lp_de.url_booking:
            street = "Nicht verfügbar"
        elif url == lp_fr.url_booking:
            street = "Non disponible"
    else:
        street = address[0].strip()

    last_element = len(address) - 1
    country = address[last_element].strip()

    new_result = {'name': name, 'address': street, 'city': city,
'country': country, 'link': link, 'price': price, 'image': image}

    dataframe.loc[len(dataframe)] = new_result
    # print(new_result)

# Zählt die Anzahl der gesamten Ergebnisseiten
total_pages = int(driver.find_element(By.CSS_SELECTOR,
'div[data-testid="pagination"] li:last-child').text)
# print(f"Total pages: {total_pages}")

```

```
# Durchläuft alle Ergebnisseiten in einer Schleife und ruft dabei jedes
Mal die Datenextraktionsfunktion auf
```

```
for page in range(total_pages):
    extract_results()
    next_page = ""
    if url == lp_en.url_booking:
        next_page = "Next page"
    elif url == lp_de.url_booking:
        next_page = "Nächste Seite"
    elif url == lp_fr.url_booking:
        next_page = "Page suivante"
    next_page_btn = driver.find_element(By.XPATH,
f'//button[contains(@aria-label,"{next_page}")]')
    next_page_btn.click()
    time.sleep(10)
```

```
# Sortiert die Zeilen im Datenrahmen in alphabetischer Reihenfolge nach
Namen
```

```
def sort_alphabetically():
    dataframe.sort_values('name')
```

```
# Sortiert die Zeilen im Datenrahmen in aufsteigender Reihenfolge nach
Preis
```

```
def sort_by_price():
    dataframe.sort_values('price', ascending=True)
```

```
# Erstellt eine CSV-Datei aus einem Datenrahmen
```

```
def create_csv():
    global searched_city
    dataframe_name = searched_city + "_scraped_data.csv"
    folder = "CSVs"
    storage_path = os.path.join(folder, dataframe_name)
    dataframe.to_csv(storage_path, index=False, encoding='utf-8')
```

```
# Erzeugt ein SQL-Skript aus einem Datenrahmen
```

```
def create_sql():
    global searched_city
    table_name = searched_city + '_scraped_data'
    folder = "SQLs"
```

```

storage_path = os.path.join(folder, table_name)
counter = 0
id = 1

f = open(storage_path + '.sql', "a", encoding="utf-8")
f.write(f'CREATE TABLE `{table_name}` (`ID` int(7) NOT NULL
AUTO_INCREMENT, `name` varchar(255) NOT NULL, `address` varchar(255) NOT
NULL, `city` varchar(255) NOT NULL, `country` varchar(255) NOT NULL,
`link` text NOT NULL, `price` varchar(255) NOT NULL, `image` text NOT
NULL, PRIMARY KEY (ID)) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;')
f.write(f'INSERT INTO `{table_name}` (`ID`, `name`, `address`, `city`,
`country`, `link`, `price`, `image`) VALUES')
for element in range(len(dataframe.index)):
    if id != len(dataframe.index):
        f.write("(" + str(id) + ", '" +
dataframe.at[dataframe.index[counter], 'name'] + "', '" +
dataframe.at[dataframe.index[counter], 'address'] + "', '" +
dataframe.at[dataframe.index[counter], 'city'] + "', '" +
dataframe.at[dataframe.index[counter], 'country'] + "', '" +
dataframe.at[dataframe.index[counter], 'link'] + "', '" +
dataframe.at[dataframe.index[counter], 'price'] + "', '" +
dataframe.at[dataframe.index[counter], 'image'] + "')")
        f.write(",")
        counter += 1
        id += 1
    else:
        f.write("(" + str(id) + ", '" +
dataframe.at[dataframe.index[counter], 'name'] + "', '" +
dataframe.at[dataframe.index[counter], 'address'] + "', '" +
dataframe.at[dataframe.index[counter], 'city'] + "', '" +
dataframe.at[dataframe.index[counter], 'country'] + "', '" +
dataframe.at[dataframe.index[counter], 'link'] + "', '" +
dataframe.at[dataframe.index[counter], 'price'] + "', '" +
dataframe.at[dataframe.index[counter], 'image'] + "')")
        f.write(";")

# Lädt Daten aus einem Datenrahmen in eine Tabelle in einer Datenbank hoch
def upload_to_database(h, u, p, d):
    global searched_city

```

```

try:
    # Connect to the database
    connection = pymysql.connect(host= h,
                                  user= u,
                                  password= p,
                                  db= d)

    cursor=connection.cursor()

    counter = 0
    table_name = searched_city + '_scraped_data'
    sql_create = f'CREATE TABLE `{table_name}` (`ID` int(7) NOT NULL
AUTO_INCREMENT, `name` varchar(255) NOT NULL, `address` varchar(255) NOT
NULL, `city` varchar(255) NOT NULL, `country` varchar(255) NOT NULL,
`link` text NOT NULL, `price` varchar(255) NOT NULL, `image` text NOT
NULL, PRIMARY KEY (ID)) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci'
    cursor.execute(sql_create)
    sql = f"INSERT INTO `{table_name}` (`name`, `address`, `city`,
`country`, `link`, `price`, `image`) VALUES (%s, %s, %s, %s, %s, %s, %s)"
    for element in range(len(dataframe.index)):
        cursor.execute(sql, (dataframe.at[dataframe.index[counter],
'name'],dataframe.at[dataframe.index[counter],
'address'],dataframe.at[dataframe.index[counter], 'city'],
dataframe.at[dataframe.index[counter],
'country'],dataframe.at[dataframe.index[counter], 'link'],
dataframe.at[dataframe.index[counter], 'price'],
dataframe.at[dataframe.index[counter], 'image'])))
        counter += 1

    connection.commit()

except Error as e:
    print(e)

finally:
    connection.close()

# Leert den Datenrahmen

```

```
def reset_dataframe():  
    dataframe.iloc[:0]
```

lp_en.py

```
# English language pack  
selection = "Language selected: English"  
selector = "Choose a language"  
location_question = "From what city/town do you want to get a hotel list  
of?"  
error_input_empty = "You must insert something before you can continue!"  
error_input_contains_numbers = "Numbers are not allowed as input!"  
error_input_contains_special_characters = "Special characters are not  
allowed as input!"  
next = "Next"  
back = "Back"  
url_booking =  
"https://www.booking.com/index.html?label=gen173bo-1DCAEoggI46AdICVgDaCyIA  
QGyAQm4ARfIAQzYAQPoAQH4AQOIAGGYAgKoAgO4AtjLoqEGwAIB0gIkODUyNmY2NmQtZjM2ZS0  
0MTIzLWFKNmUtMTRhYTRiMzE5MzEw2AIE4AIB&sid=beb1b7a77d9ad5c18d04f2936d257405  
&lang_changed=1&sb_price_type=total&lang=en-us&soz=1&selected_currency=USD  
"  
start_scraping = "Start scraping"  
scraping_in_process = "The Scraper is now looking for information. Please  
have a bit patience."  
order_question = "Would you like to sort the gathered info in a specific  
fashion?"  
sort_alphabet = "Sort alphabetically"  
sort_price = "Sort by price (ascendant)"  
no_sort = "Do not sort"  
storage_question = "How would you like to save the collected data?"  
generate_file = "Generate CSV"  
generate_script = "Generate SQL"  
generate_all = "Generate both files"  
db_upload = "Upload to a database"  
db_upload_question = "Please insert the correct credentials in each input  
field."  
db_host_input_label = "Host name"  
db_user_input_label = "Username"
```

```
db_password_input_label = "Password"
db_name_input_label = "Database name"
send_data = "Send data"
process_done = "The Scraping process has reached its end. If you wish, you
can restart the application by clicking on this button."
restart = "Restart"
```

lp_de.py

```
# Deutsches Sprachpaket
selection = "Gewählte Sprache: Deutsch"
selector = "Wählen Sie eine Sprache"
location_question = "Von welcher Stadt möchten Sie eine Hotelliste
erhalten?"
error_input_empty = "Sie müssen etwas einfügen, bevor Sie fortfahren
können!"
error_input_contains_numbers = "Zahlen sind als Eingabe nicht erlaubt!"
error_input_contains_special_characters = "Sonderzeichen sind als Eingabe
nicht erlaubt!"
next = "Weiter"
back = "Zurück"
url_booking =
"https://www.booking.com/index.de.html?label=gen173bo-1DCAEoggI46AdICVgDaC
yIAQGYAQm4ARfIAQzYAQPoaAQH4AQOIAGGYAgKoAgO4AtjLoqEGwAIB0gIkODUyNmY2NmQtZjM2
ZS00MTIzLWFKNmUtMTRhYTRiMzE5MzEw2AIE4AIB&sid=beb1b7a77d9ad5c18d04f2936d257
405&lang_changed=1&sb_price_type=total&lang=de&soz=1&selected_currency=EUR
"
start_scraping = "Scraper starten"
scraping_in_process = "Der Scraper sucht jetzt nach Informationen. Bitte
haben Sie ein wenig Geduld."
order_question = "Möchten Sie die gesammelten Informationen auf eine
bestimmte Weise sortieren?"
sort_alphabet = "Alphabetisch sortieren"
sort_price = "Nach Preis sortieren (aufsteigend)"
no_sort = "Nicht sortieren"
storage_question = "Wie möchten Sie die gesammelten Daten speichern?"
generate_file = "CSV generieren"
generate_script = "SQL generieren"
generate_all = "Beide Dateien generieren"
```

```
db_upload = "In eine Datenbank hochladen"
db_upload_question = "Bitte geben Sie die korrekten Zugangsdaten in jedes
Eingabefeld ein."
db_host_input_label = "Hostname"
db_user_input_label = "Benutzername"
db_password_input_label = "Passwort"
db_name_input_label = "Name der Datenbank"
send_data = "Daten senden"
process_done = "Der Scraping-Prozess hat sein Ende erreicht. Wenn Sie
möchten, können Sie die Anwendung neu starten, indem Sie auf diese
Schaltfläche klicken."
restart = "Neu starten"
```

lp_fr.py

```
# Paquet de langue française
selection = "Langue sélectionnée: Français"
selector = "Choisissez une langue"
location_question = "De quelle ville/village souhaitez-vous obtenir une
liste d'hôtels?"
error_input_empty = "Vous devez insérer quelque chose avant de pouvoir
continuer!"
error_input_contains_numbers = "Les chiffres ne sont pas autorisés en
entrée!"
error_input_contains_special_characters = "Les caractères spéciaux ne sont
pas autorisés en entrée !"
next = "Avant"
back = "Arrière"
url_booking =
"https://www.booking.com/index.fr.html?label=gen173bo-1DCAEoggI46AdICVgDaC
yIAQGYAQm4ARfIAQzYAQPoAQH4AQOIAGGYAgKoAgO4AtjLoqEGwAIB0gIkODUyNmY2NmQtZjM2
ZS00MTIzLWFKNmUtMTRhYTJmZmE5MzEw2AIE4AIB&sid=beb1b7a77d9ad5c18d04f2936d257
405&lang_changed=1&sb_price_type=total&lang=fr&soz=1&selected_currency=EUR
"
start_scraping = "Lancer le scraper"
scraping_in_process = "Le Scraper est actuellement à la recherche
d'informations. Veuillez faire preuve d'un peu de patience."
order_question = "Souhaitez-vous trier les informations recueillies d'une
manière spécifique ?"
```

```
sort_alphabet = "Trier par ordre alphabétique"
sort_price = "Trier par prix (ascendant)"
no_sort = "Ne pas trier"
storage_question = "Comment souhaitez-vous sauvegarder les données collectées?"
generate_file = "Générer un CSV"
generate_script = "Générer un SQL"
generate_all = "Générer les deux fichiers"
db_upload = "Charger dans une base de données"
db_upload_question = "Veuillez insérer les informations d'identification correctes dans chaque champ de saisie."
db_host_input_label = "Nom d'hôte"
db_user_input_label = "Nom d'utilisateur"
db_password_input_label = "Mot de passe"
db_name_input_label = "Nom de la base de données"
send_data = "Envoyer des données"
process_done = "Le processus de scraping est arrivé à son terme. Si vous le souhaitez, vous pouvez relancer l'application en cliquant sur ce bouton."
restart = "Recommencer"
```