**Приложение Г. Текст программы**

АННОТАЦИЯ

Данное консольное приложение предназначено для организации и управления информацией о компаниях и пользователях, предоставляя удобный интерфейс для регистрации и авторизации пользователей, а также для администраторов, отвечающих за управление контентом. Основное применение включает в себя возможность добавления, редактирования и удаления информации о компаниях, а также автоматизированный сбор данных о компаниях с использованием Selenium WebDriver. Пользователи могут просматривать список всех компаний, добавлять компании в избранное и просматривать свои избранные компании. Администраторы обладают расширенными возможностями для управления данными, включая создание учетных записей администраторов и обеспечение актуальности информации о компаниях. Это делает приложение полезным инструментом как для обычных пользователей, желающих легко находить и управлять информацией о компаниях, так и для администраторов, обеспечивающих качество и актуальность данных.

В разделе «Текст программы» указано наименование программы, область применения программы, модули программы в виде таблицы с указанием описания и размера каждого модуля, код программы.

1

1. ТЕКСТ ПРОГРАММЫ

1.1. Наименование объекта

Наименование – «Консольное приложение для управления компаниями».

1.2. Область применения объекта

Консольное приложение предназначено для организации и управления информацией о компаниях, предоставляя пользователям удобный доступ к данным компаний, возможность добавления компаний в список избранных, а также административные функции для управления контентом. Основное применение включает регистрацию и авторизацию пользователей, автоматизированный сбор и обновление данных о компаниях с использованием Selenium WebDriver, управление списком избранных компаний и административное управление контентом, включая добавление, изменение и удаление информации о компаниях. Это делает приложение полезным инструментом как для обычных пользователей, желающих легко находить и управлять информацией о компаниях, так и для администраторов, обеспечивающих актуальность и качество данных.

1.3. Модули

Таблица 1. Модули

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Описание | Количество строк | Размер  файла в  Кб |
| 1 | connect\_db | Функция для подключения к базе данных PostgreSQL.. | 20 | 4 |
| 2 | create\_tables | Функция для создания необходимых таблиц в базе данных, если они не существуют. | 40 | 5 |
| 3 | create\_admin | Функция для создания учетной записи администратора. | 40 | 7 |
| 4 | ensure\_admin\_exists | Функция для проверки наличия администратора в базе данных и создания при отсутствии. | 20 | 6 |
| 5 | register\_user | Функция для регистрации нового пользователя. | 30 | 8 |
| 6 | login\_user | Функция для авторизации пользователя. | 30 | 2 |
| 7 | user\_authentication | Функция для меню авторизации, позволяющая пользователю войти, зарегистрироваться или продолжить без авторизации. | 30 | 1 |
| 8 | view\_all\_companies | Функция для отображения списка всех компаний в базе данных. | 10 | 1 |
| 9 | add\_to\_favorites | Функция для добавления компании в список избранных пользователем. | 15 | 1 |
| 10 | view\_favorites | Функция для просмотра списка избранных компаний пользователя. | 15 | 1 |
| 11 | user\_menu | Меню пользователя с опциями для просмотра компаний, добавления в избранное и просмотра избранного. | 25 | 2 |
| 12 | admin\_add\_company | Функция для добавления новой компании в базу данных. | 20 | 1 |
| 13 | admin\_modify\_company | Функция для изменения информации о существующей компании. | 25 | 1 |
| 14 | admin\_delete\_company | Функция для удаления компании из базы данных | 20 | 1 |
| 15 | admin\_menu | Меню администратора с опциями для просмотра, добавления, изменения и удаления компаний. | 25 | 1 |
| 16 | main | Основная функция, объединяющая все компоненты приложения: подключение к БД, настройка WebDriver, сбор данных, вызов меню. | 60 | 3 |

1.4. Код программы

import bcrypt

import psycopg2

from psycopg2 import sql

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.chrome.service import Service

from selenium.webdriver.chrome.options import Options

from webdriver\_manager.chrome import ChromeDriverManager

import time

import sys

def connect\_db():

try:

print("Попытка подключения к базе данных...")

conn = psycopg2.connect(

dbname="Fabric",

user="postgres",

password="123",

host="localhost",

port="5432"

)

print("Подключение успешно.")

return conn

except Exception as e:

print(f"Ошибка подключения к базе данных: {e}")

sys.exit(1)

def create\_tables(conn):

try:

print("Создание таблиц, если они не существуют...")

cur = conn.cursor()

cur.execute("""

CREATE TABLE IF NOT EXISTS users (

id SERIAL PRIMARY KEY,

username TEXT UNIQUE NOT NULL,

password\_hash TEXT NOT NULL

);

""")

cur.execute("""

SELECT column\_name

FROM information\_schema.columns

WHERE table\_name='users' and column\_name='is\_admin';

""")

if not cur.fetchone():

cur.execute("ALTER TABLE users ADD COLUMN is\_admin BOOLEAN DEFAULT FALSE;")

print("Добавлен столбец is\_admin в таблицу users.")

cur.execute("""

CREATE TABLE IF NOT EXISTS companies (

id SERIAL PRIMARY KEY,

url TEXT UNIQUE,

company\_name TEXT,

product\_section TEXT,

company\_description TEXT,

products TEXT,

contacts TEXT,

email TEXT,

phone TEXT

);

""")

cur.execute("""

CREATE TABLE IF NOT EXISTS favorites (

user\_id INTEGER REFERENCES users(id) ON DELETE CASCADE,

company\_id INTEGER REFERENCES companies(id) ON DELETE CASCADE,

PRIMARY KEY (user\_id, company\_id)

);

""")

conn.commit()

cur.close()

print("Таблицы успешно созданы или обновлены.")

except Exception as e:

print(f"Ошибка при создании таблиц: {e}")

conn.rollback()

def create\_admin(conn):

try:

cur = conn.cursor()

print("\n--- Создание Администратора ---")

while True:

username = input("Введите имя пользователя администратора: ").strip()

if not username:

print("Имя пользователя не может быть пустым.")

continue

cur.execute("SELECT \* FROM users WHERE username = %s;", (username,))

if cur.fetchone():

print("Такое имя пользователя уже существует. Попробуйте другое.")

else:

break

while True:

password = input("Введите пароль администратора: ")

password\_confirm = input("Подтвердите пароль администратора: ")

if password != password\_confirm:

print("Пароли не совпадают. Попробуйте снова.")

elif len(password) < 6:

print("Пароль должен содержать не менее 6 символов.")

else:

break

password\_hash = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt()).decode('utf-8')

try:

cur.execute(

"INSERT INTO users (username, password\_hash, is\_admin) VALUES (%s, %s, TRUE);",

(username, password\_hash)

)

conn.commit()

print("Администратор успешно создан!")

except Exception as e:

conn.rollback()

print(f"Ошибка при создании администратора: {e}")

finally:

cur.close()

except Exception as e:

print(f"Ошибка при создании администратора: {e}")

def ensure\_admin\_exists(conn):

try:

cur = conn.cursor()

cur.execute("SELECT COUNT(\*) FROM users WHERE is\_admin = TRUE;")

count = cur.fetchone()[0]

if count == 0:

print("В базе данных нет администратора. Необходимо создать администратора.")

create\_admin(conn)

cur.close()

except Exception as e:

print(f"Ошибка при проверке наличия администратора: {e}")

def register\_user(conn):

try:

cur = conn.cursor()

print("\n--- Регистрация Пользователя ---")

while True:

print("Пожалуйста, введите имя пользователя:")

username = input("Имя пользователя: ").strip()

if not username:

print("Имя пользователя не может быть пустым.")

continue

cur.execute("SELECT \* FROM users WHERE username = %s;", (username,))

if cur.fetchone():

print("Такое имя пользователя уже существует. Попробуйте другое.")

else:

break

while True:

print("Пожалуйста, введите пароль:")

password = input("Пароль: ")

password\_confirm = input("Подтвердите пароль: ")

if password != password\_confirm:

print("Пароли не совпадают. Попробуйте снова.")

elif len(password) < 6:

print("Пароль должен содержать не менее 6 символов.")

else:

break

password\_hash = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt()).decode('utf-8')

try:

cur.execute(

"INSERT INTO users (username, password\_hash, is\_admin) VALUES (%s, %s, FALSE);",

(username, password\_hash)

)

conn.commit()

print("Регистрация прошла успешно!")

except Exception as e:

conn.rollback()

print(f"Ошибка при регистрации: {e}")

finally:

cur.close()

except Exception as e:

print(f"Неизвестная ошибка в функции регистрации: {e}")

def login\_user(conn):

try:

cur = conn.cursor()

print("\n--- Авторизация Пользователя ---")

username = input("Имя пользователя: ").strip()

password = input("Пароль: ")

cur.execute("SELECT id, password\_hash, is\_admin FROM users WHERE username = %s;", (username,))

result = cur.fetchone()

if result:

user\_id, stored\_password\_hash, is\_admin = result

if bcrypt.checkpw(password.encode('utf-8'), stored\_password\_hash.encode('utf-8')):

print("Авторизация успешна!")

cur.close()

return user\_id, is\_admin

else:

print("Неверный пароль.")

else:

print("Пользователь не найден.")

cur.close()

return None, None

except Exception as e:

print(f"Ошибка при авторизации: {e}")

return None, None

def user\_authentication(conn):

try:

while True:

print("\n--- Меню Авторизации ---")

print("1. Войти")

print("2. Зарегистрироваться")

print("3. Продолжить без авторизации")

choice = input("Выберите действие (1/2/3): ").strip()

if choice == '1':

user\_id, is\_admin = login\_user(conn)

if user\_id:

return user\_id, is\_admin

elif choice == '2':

register\_user(conn)

elif choice == '3':

print("Продолжаем без авторизации.")

return None, False

else:

print("Некорректный выбор. Пожалуйста, выберите 1, 2 или 3.")

except Exception as e:

print(f"Ошибка в функции аутентификации: {e}")

return None, False

def view\_all\_companies(conn):

try:

cur = conn.cursor()

cur.execute("SELECT id, company\_name, url FROM companies;")

companies = cur.fetchall()

if not companies:

print("Нет доступных компаний.")

else:

print("\n--- Список всех компаний ---")

for company in companies:

print(f"ID: {company[0]}, Название: {company[1]}, URL: {company[2]}")

cur.close()

except Exception as e:

print(f"Ошибка при просмотре компаний: {e}")

def add\_to\_favorites(conn, user\_id):

try:

cur = conn.cursor()

company\_id = input("Введите ID компании для добавления в избранное: ").strip()

if not company\_id.isdigit():

print("Некорректный ID компании.")

return

company\_id = int(company\_id)

cur.execute("SELECT \* FROM companies WHERE id = %s;", (company\_id,))

if not cur.fetchone():

print("Компания с таким ID не существует.")

return

cur.execute(

"INSERT INTO favorites (user\_id, company\_id) VALUES (%s, %s) ON CONFLICT DO NOTHING;",

(user\_id, company\_id)

)

conn.commit()

print("Компания добавлена в избранное.")

cur.close()

except Exception as e:

conn.rollback()

print(f"Ошибка при добавлении в избранное: {e}")

def view\_favorites(conn, user\_id):

try:

cur = conn.cursor()

cur.execute("""

SELECT c.id, c.company\_name, c.url

FROM companies c

JOIN favorites f ON c.id = f.company\_id

WHERE f.user\_id = %s;

""", (user\_id,))

favorites = cur.fetchall()

if not favorites:

print("У вас нет избранных компаний.")

else:

print("\n--- Ваши Избранные Компании ---")

for company in favorites:

print(f"ID: {company[0]}, Название: {company[1]}, URL: {company[2]}")

cur.close()

except Exception as e:

print(f"Ошибка при просмотре избранных компаний: {e}")

def user\_menu(conn, user\_id):

while True:

print("\n--- Меню Пользователя ---")

print("1. Просмотреть все компании")

print("2. Добавить компанию в избранное")

print("3. Просмотреть избранные компании")

print("4. Выйти")

choice = input("Выберите действие (1/2/3/4): ").strip()

if choice == '1':

view\_all\_companies(conn)

elif choice == '2':

add\_to\_favorites(conn, user\_id)

elif choice == '3':

view\_favorites(conn, user\_id)

elif choice == '4':

print("Выход из системы.")

break

else:

print("Некорректный выбор. Пожалуйста, выберите 1, 2, 3 или 4.")

def admin\_add\_company(conn):

try:

cur = conn.cursor()

print("\n--- Добавление Новой Компании ---")

url = input("Введите URL компании: ").strip()

company\_name = input("Введите название компании: ").strip()

product\_section = input("Введите раздел продукции: ").strip()

company\_description = input("Введите описание компании: ").strip()

products = input("Введите товары компании (через запятую): ").strip()

contacts = input("Введите контакты компании: ").strip()

email = input("Введите электронную почту: ").strip()

phone = input("Введите телефон: ").strip()

cur.execute("""

INSERT INTO companies (url, company\_name, product\_section, company\_description, products, contacts, email, phone)

VALUES (%s, %s, %s, %s, %s, %s, %s, %s)

ON CONFLICT (url) DO NOTHING;

""", (url, company\_name, product\_section, company\_description, products, contacts, email, phone))

conn.commit()

print("Компания успешно добавлена.")

cur.close()

except Exception as e:

conn.rollback()

print(f"Ошибка при добавлении компании: {e}")

def admin\_modify\_company(conn):

try:

cur = conn.cursor()

print("\n--- Изменение Компании ---")

company\_id = input("Введите ID компании для изменения: ").strip()

if not company\_id.isdigit():

print("Некорректный ID компании.")

return

company\_id = int(company\_id)

cur.execute("SELECT \* FROM companies WHERE id = %s;", (company\_id,))

company = cur.fetchone()

if not company:

print("Компания с таким ID не существует.")

return

print("Введите новые значения (оставьте пустым, чтобы не изменять):")

url = input(f"URL [{company[1]}]: ").strip() or company[1]

company\_name = input(f"Название компании [{company[2]}]: ").strip() or company[2]

product\_section = input(f"Раздел продукции [{company[3]}]: ").strip() or company[3]

company\_description = input(f"Описание компании [{company[4]}]: ").strip() or company[4]

products = input(f"Товары компании [{company[5]}]: ").strip() or company[5]

contacts = input(f"Контакты [{company[6]}]: ").strip() or company[6]

email = input(f"Электронная почта [{company[7]}]: ").strip() or company[7]

phone = input(f"Телефон [{company[8]}]: ").strip() or company[8]

cur.execute("""

UPDATE companies

SET url = %s,

company\_name = %s,

product\_section = %s,

company\_description = %s,

products = %s,

contacts = %s,

email = %s,

phone = %s

WHERE id = %s;

""", (url, company\_name, product\_section, company\_description, products, contacts, email, phone, company\_id))

conn.commit()

print("Компания успешно обновлена.")

cur.close()

except Exception as e:

conn.rollback()

print(f"Ошибка при изменении компании: {e}")

def admin\_delete\_company(conn):

try:

cur = conn.cursor()

print("\n--- Удаление Компании ---")

company\_id = input("Введите ID компании для удаления: ").strip()

if not company\_id.isdigit():

print("Некорректный ID компании.")

return

company\_id = int(company\_id)

cur.execute("SELECT \* FROM companies WHERE id = %s;", (company\_id,))

if not cur.fetchone():

print("Компания с таким ID не существует.")

return

confirm = input("Вы уверены, что хотите удалить эту компанию? (y/n): ").strip().lower()

if confirm != 'y':

print("Удаление отменено.")

return

cur.execute("DELETE FROM companies WHERE id = %s;", (company\_id,))

conn.commit()

print("Компания успешно удалена.")

cur.close()

except Exception as e:

conn.rollback()

print(f"Ошибка при удалении компании: {e}")

def admin\_menu(conn):

while True:

print("\n--- Меню Администратора ---")

print("1. Просмотреть все компании")

print("2. Добавить новую компанию")

print("3. Изменить существующую компанию")

print("4. Удалить компанию")

print("5. Выйти")

choice = input("Выберите действие (1/2/3/4/5): ").strip()

if choice == '1':

view\_all\_companies(conn)

elif choice == '2':

admin\_add\_company(conn)

elif choice == '3':

admin\_modify\_company(conn)

elif choice == '4':

admin\_delete\_company(conn)

elif choice == '5':

print("Выход из системы.")

break

else:

print("Некорректный выбор. Пожалуйста, выберите 1, 2, 3, 4 или 5.")

def main():

conn = connect\_db()

create\_tables(conn)

ensure\_admin\_exists(conn)

user\_id, is\_admin = user\_authentication(conn)

print("Настройка Selenium WebDriver...")

try:

chrome\_options = Options()

chrome\_options.add\_argument('--headless') # Запуск браузера в фоновом режиме

chrome\_options.add\_argument('--no-sandbox')

chrome\_options.add\_argument('--disable-dev-shm-usage')

service = Service(ChromeDriverManager().install())

driver = webdriver.Chrome(service=service, options=chrome\_options)

print("Selenium WebDriver успешно настроен.")

except Exception as e:

print(f"Ошибка при настройке Selenium WebDriver: {e}")

conn.close()

sys.exit(1)

urls = [

"https://fabricators.ru/proizvoditel/kaneks",

"https://fabricators.ru/proizvoditel/gazprom",

"https://fabricators.ru/proizvoditel/yaroslavskiy-kozhevennyy-zavod-hrom",

"https://fabricators.ru/proizvoditel/vepsskiy-srub",

"https://fabricators.ru/proizvoditel/moskovskiy-kombinat-shampanskih-vin-mkshv"

]

try:

cur = conn.cursor()

except Exception as e:

print(f"Ошибка при создании курсора: {e}")

driver.quit()

conn.close()

sys.exit(1)

def upsert\_company(data):

try:

insert\_query = """

INSERT INTO companies (url, company\_name, product\_section, company\_description, products, contacts, email, phone)

VALUES (%s, %s, %s, %s, %s, %s, %s, %s)

ON CONFLICT (url) DO UPDATE SET

company\_name = EXCLUDED.company\_name,

product\_section = EXCLUDED.product\_section,

company\_description = EXCLUDED.company\_description,

products = EXCLUDED.products,

contacts = EXCLUDED.contacts,

email = EXCLUDED.email,

phone = EXCLUDED.phone;

"""

cur.execute(insert\_query, (

data['url'],

data['company\_name'],

data['product\_section'],

data['company\_description'],

data['products'],

data['contacts'],

data['email'],

data['phone']

))

conn.commit()

print(f"Данные для {data['url']} успешно сохранены в базе данных.")

except Exception as e:

conn.rollback()

print(f"Ошибка при вставке данных для {data['url']}: {e}")

for urls in urls:

try:

print(f"\nПереход на URL: {url}")

driver.get(url)

print("Ожидание загрузки страницы...")

time.sleep(3)

except Exception as e:

print(f"Ошибка при загрузке URL {url}: {e}")

continue

data = {'url': url}

try:

print("Извлечение названия компании...")

data['company\_name'] = driver.find\_element(By.CSS\_SELECTOR, 'h1.col-wt').text

except Exception as e:

data['company\_name'] = "Название не найдено"

print(f"Не удалось найти название компании: {e}")

try:

print("Извлечение продукции...")

data['product\_section'] = driver.find\_element(By.CSS\_SELECTOR, '#section-st-block2 ul').text

except Exception as e:

data['product\_section'] = "Раздел с продукцией не найден"

print(f"Не удалось найти раздел с продукцией: {e}")

try:

print("Извлечение описания компании...")

data['company\_description'] = driver.find\_element(By.CSS\_SELECTOR,

'#section-st-block1 .about-project p').text

except Exception as e:

data['company\_description'] = "Описание компании не найдено"

print(f"Не удалось найти описание компании: {e}")

try:

print("Извлечение товаров компании...")

product\_items = driver.find\_elements(By.CSS\_SELECTOR, '.text-bl-grid .content-prod .title-site--h4')

data['products'] = ", ".join([item.text for item in product\_items])

except Exception as e:

data['products'] = "Товары компании не найдены"

print(f"Не удалось найти товары компании: {e}")

try:

print("Извлечение контактов...")

data['contacts'] = driver.find\_element(By.CSS\_SELECTOR, '#section-st-block5 .content-contact').text

except Exception as e:

data['contacts'] = "Контакты не найдены"

print(f"Не удалось найти контакты: {e}")

try:

print("Извлечение электронной почты...")

email\_element = driver.find\_element(By.CSS\_SELECTOR, '.content-contact a[href^="mailto:"]')

data['email'] = email\_element.get\_attribute('href').replace('mailto:', '')

except Exception as e:

data['email'] = 'Электронная почта не найдена'

print(f"Не удалось найти электронную почту: {e}")

try:

print("Извлечение телефона...")

phone\_element = driver.find\_element(By.CSS\_SELECTOR, '.content-contact a[href^="tel:"]')

data['phone'] = phone\_element.get\_attribute('href').replace('tel:', '')

except Exception as e:

data['phone'] = 'Телефон не найден'

print(f"Не удалось найти телефон: {e}")

print(f"\n--- Информация для сайта: {url} ---")

print("Название компании:", data['company\_name'])

print("Продукция:", data['product\_section'])

print("Описание компании:", data['company\_description'])

print("Товары компании:", data['products'])

print("Контакты:", data['contacts'])

print("Электронная почта:", data['email'])

print("Телефон:", data['phone'])

upsert\_company(data)

try:

cur.close()

except Exception as e:

print(f"Ошибка при закрытии курсора: {e}")

try:

driver.quit()

print("Selenium WebDriver закрыт.")

except Exception as e:

print(f"Ошибка при закрытии Selenium WebDriver: {e}")

if user\_id:

if is\_admin:

admin\_menu(conn)

else:

user\_menu(conn, user\_id)

if not conn.closed:

try:

conn.close()

print("Соединение с базой данных закрыто.")

except Exception as e:

print(f"Ошибка при закрытии соединения с базой данных: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

main()