ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ

«ШАГ В БУДУЩЕЕ, МОСКВА»

регистрационный номер

название факультета

название кафедры

Автоматическая запись к врачу
с получением уведомлений

название работы

Автор: Новокшанов Евгений Андреевич
фамилия, имя, отчество
г. Москва, ГБОУ Физматшкола №2007, 11 «Б»
наименование учебного заведения, класс

Научный руководитель:
фамилия, имя, отчество
место работы
звание, должность
подпись научного руководителя

Москва - 2020

Аннотация

Автоматическая запись к врачу с получением уведомлений (Quick appointment with a doctor – QAWD) — это приложение, которое сможет заметно улучшить вашу жизнь в сфере записи к врачу. Если вы стоите в онлайн очередях или очередях поликлиник, то с QAWD вы сможете освободить свое время и посвятить его любимым занятиям и хобби, потому что данное веб-приложение будет выполнять запись к врачу без вашего непосредственного наблюдения за его работой. Вам остается только нажать на кнопку и получить уведомление на почту о успешной записи.

Цель работы:

Разработать программу для автоматической записи к врачу с получением уведомлений.

Задачи:

а) решить проблему потери времени людей, которые стоят в онлайн-очередях для записи на прием к врачу;
б) облегчить процесс записи к врачу на определенное время и дату.

Методы и инструменты:

а) программа написана на языке Python 3.7;
б) использована библиотека Selenium WebDriver (библиотека для управления браузерами);
в) работа с веб-сервисами;
г) программа адаптирована для сайта, который был специально для этого создан, что бы не нарушать работу реальных сайтов клиник.
д) в качестве примера сайта медицинского учреждения используется сайт (https://zhenyanovokshanov1.wixsite.com/klinika/blank-5), созданный с использованием бесплатного конструктора сайтов «Wix».

Содержание

1 Введение

Люди давно уже не стоят часами в очередях, чтобы записаться к врачу. Существует множество сервисов, через которые можно выполнить запись, не выходя из дома, однако и часто требует много времени. Люди, имеющие онкологические и сердечно-сосудистые заболевания, не могут записаться на прием, т.к. все время занято, и они вынуждены постоянно следить, не освободилось ли место в нужное время.

Я решил разобраться с этой проблемой и написать собственное веб-приложение, которое сможет автоматически искать свободное время для записи. Это освободило бы много времени у людей, которые в этом нуждаются.

В основу программы легли следующие идеи:

а) Быстрая запись к врачу;
б) Автоматический поиск свободного времени для записи;
в) Понятный и простой интерфейс, которым будет удобно пользоваться не только людям среднего возраста, но и людям преклонного возраста.

Веб-приложение является наиболее удобным в использовании, т.к. оно не занимает память на устройстве, и пользоваться им можно на разных устройствах (ПК, смартфон, планшет, ноутбук) с различными операционными системами.

В чем же заключаются отличия данного приложения от аналогов в Google Play, App Store и сайтов государственных услуг? Оно заключается в том, что пользователю не нужно будет заходить непосредственно на сайт медицинского учреждения, необходимо будет лишь открыть вкладку в браузере по ссылке и вписать свои данные, необходимые для записи. Далее программа выполнит всю работу сама и отправит вам сообщение об успешной записи к врачу на указанную вами почту.

Описание работы программы:

После введения определенных параметров записи (интервалы даты и времени, специализация врача, Ф.И.О. врача) выполняется автоматический поиск и отслеживание доступного для записи к врачу даты и времени. После того, как программа проведет запись, она отправит уведомление с указанием даты и времени приема у врача.

2 Основная часть

2.1 Используемые методы

Приложение написано на языке Python 3. Также для создания веб-интерфейса использовался язык HTML5.

Я выбрал язык Python, потому что он прост в использовании и для него написано множество библиотек, которые упрощают работу.

Приложение состоит из нескольких функциональных блоков:

- **Интерфейс (HTML-страница)** – форма для ввода данных пользователя;
- **Связующая программа** – является интерфейсом для взаимодействия браузера пользователя с основной программой;
- **Основная программа** – данная программа выполняет поиск свободного времени и запись к врачу;
- **Уведомления** – программа отправки пользователю сообщения об успешной записи к врачу.

На рисунке Рисунок *1*представлена связь основных блоков программы между собой:

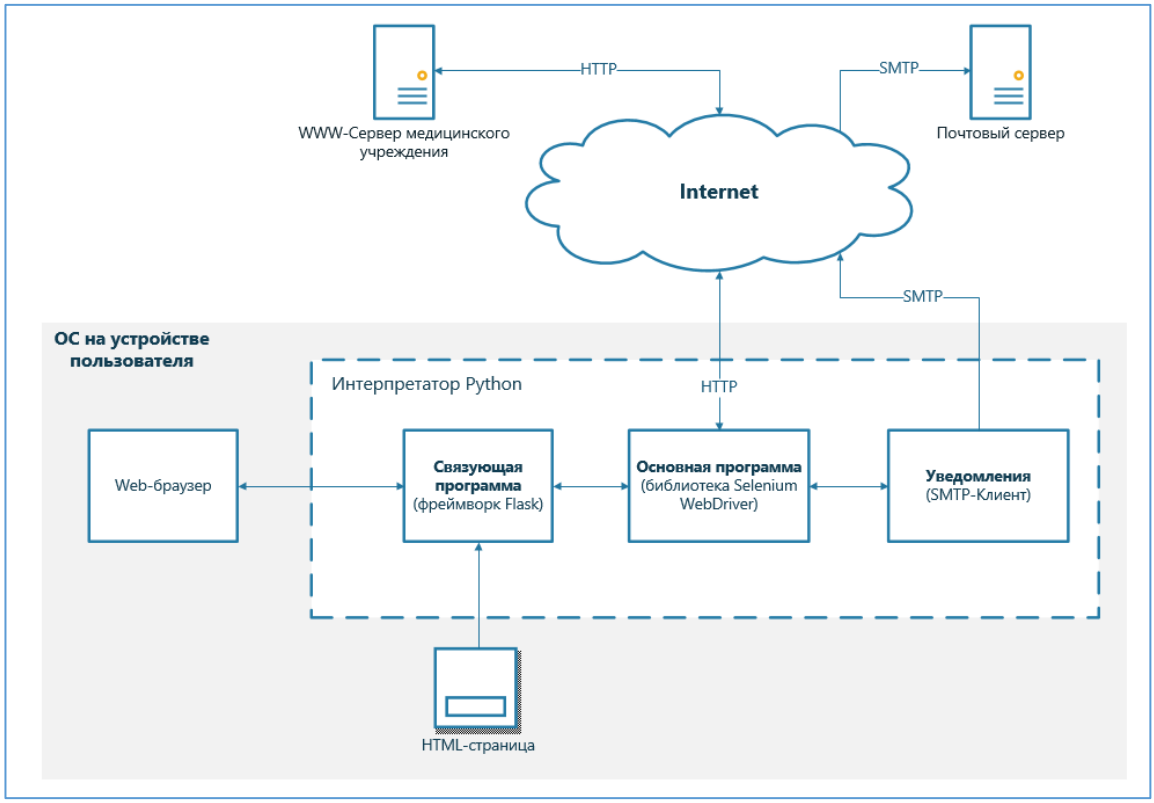

Рисунок 1 - Связь основных блоков программы между собой

На рисунке (Рисунок *2*) представлены основные шаги выполнения программы.

Рисунок 2 - Основные шаги выполнения программы

2.2 Интерфейс (HTML-страница)

При переходе по ссылке на веб-приложение, пользователь попадает на HTML-страницу, где отображаются поля, необходимые для ввода данных записи к врачу.

- Имя;
- Электронная почта;
- Номер телефона;
- Направление (например, Офтальмолог);
- Врач;
- Интервал времени, в которое вам удобно посетить врача;
- Интервал дней, в которые вам необходимо посетить врача.

После того, как пользователь заполнит все вышеперечисленные поля, необходимо нажать на кнопку «Записаться».

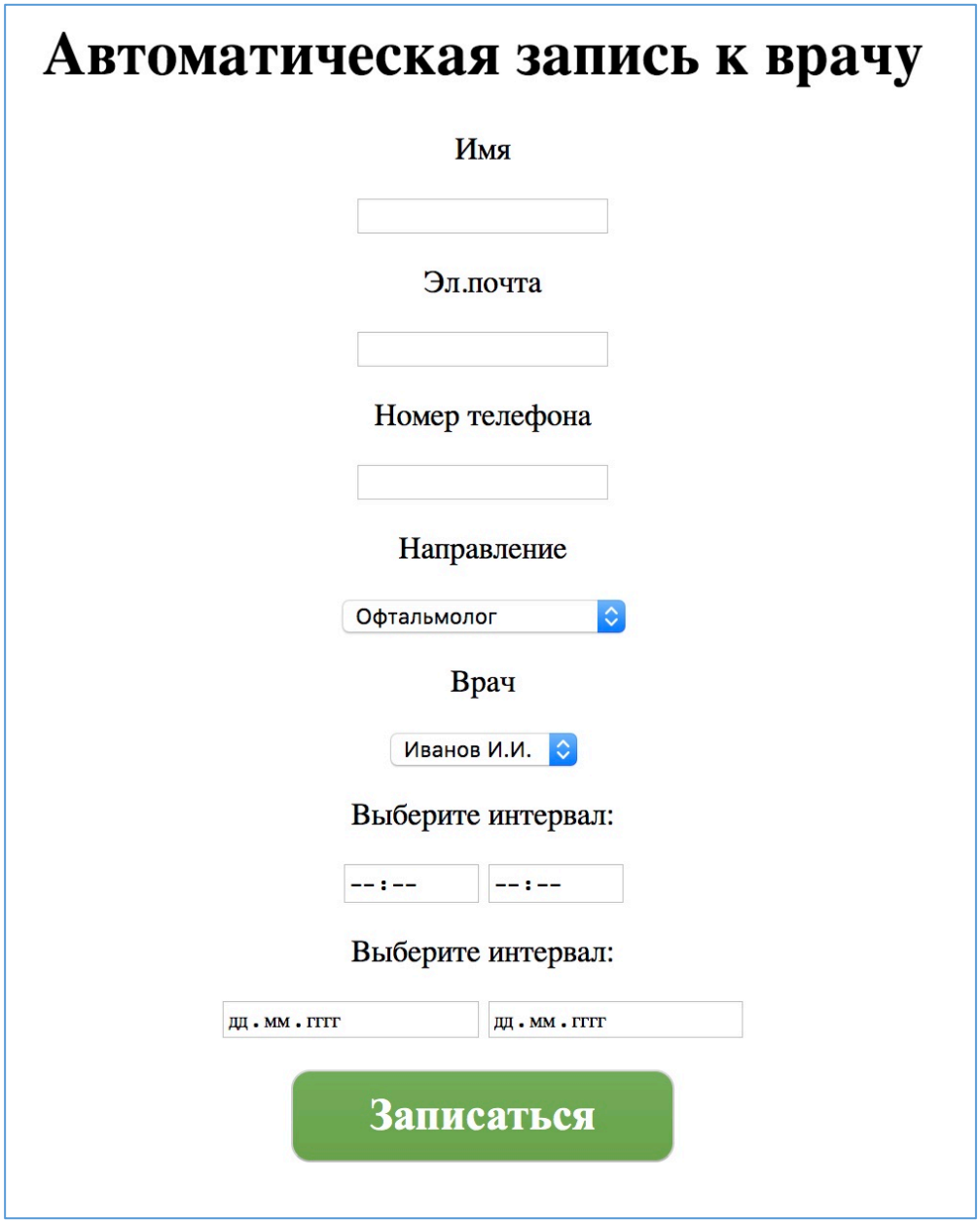

Рисунок 3- Интерфейс программы

После нажатия на кнопку «Записаться», данные пользователя передаются в связующую программу.

При выборе направления и врача, открывается окно с датой выбора направления и доктора соответственно. На рисунке Рисунок *4* показан пример выбора направления врача.

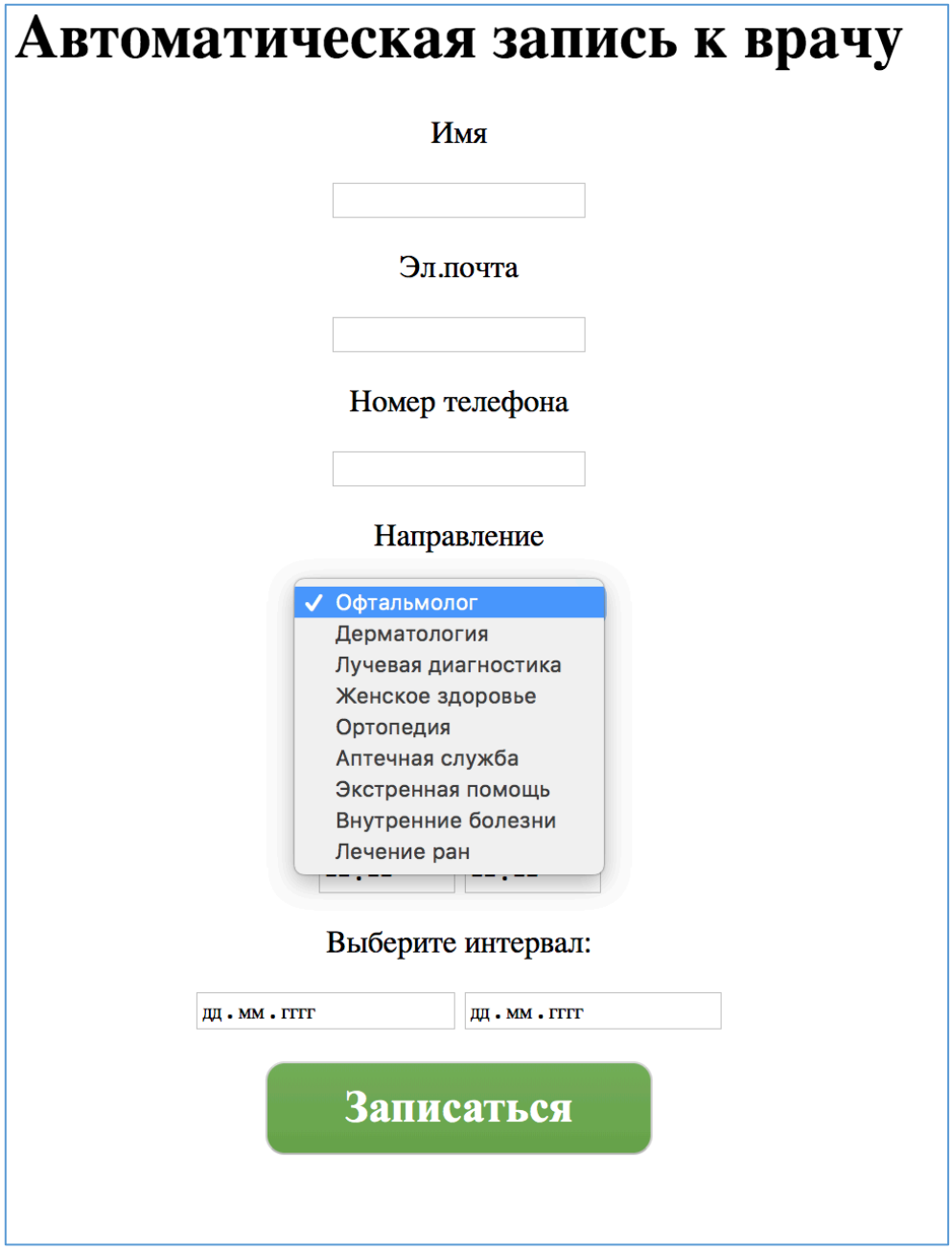

Рисунок 4 - Выбор направления врача

После выбора направления, пользователю необходимо выбрать самого врача. Пример выбора врача показан на рисунке Рисунок *5*.

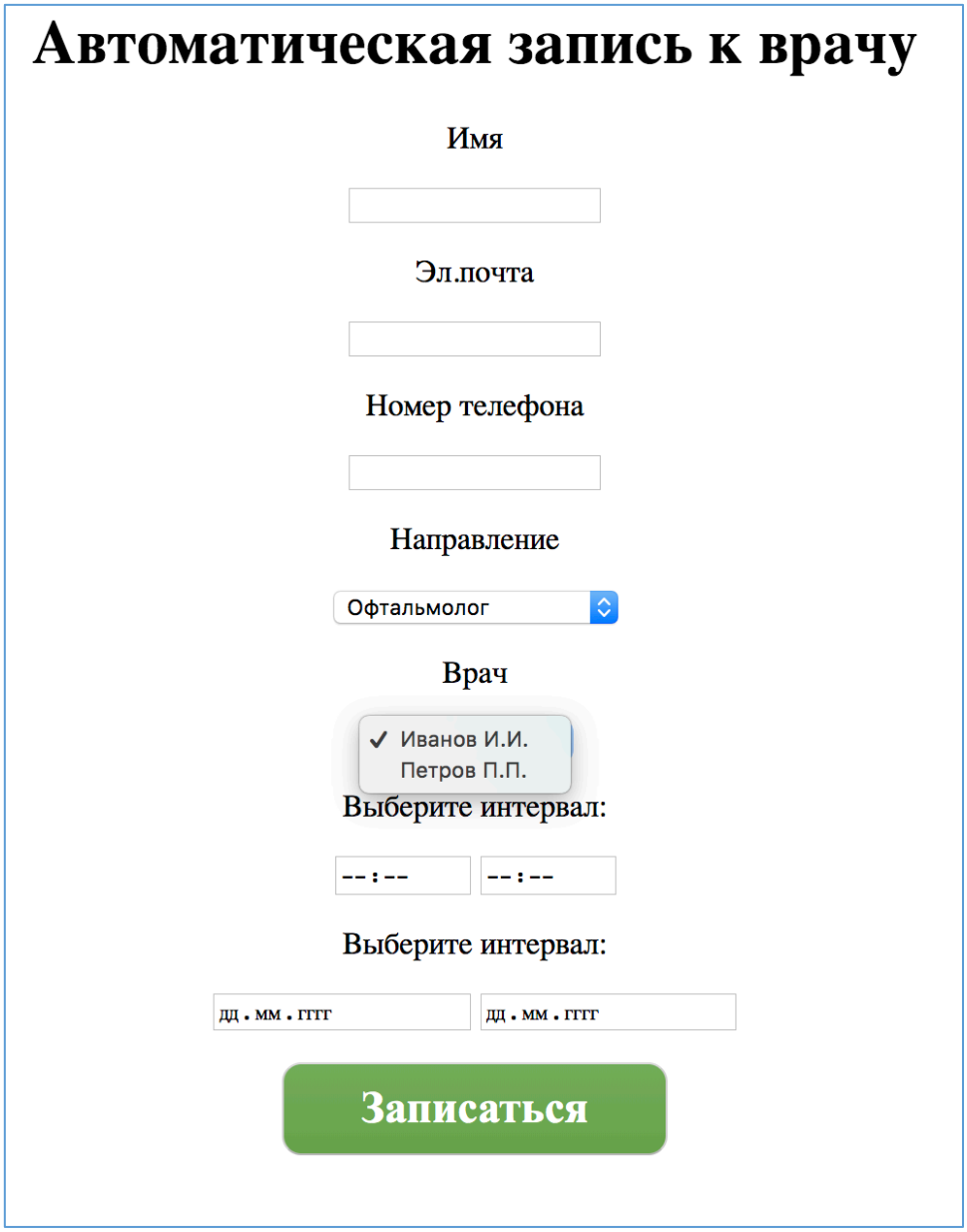

Рисунок 5 - Выбор врача

2.3 Связующая программа

Связующая программа - является веб-приложением, разработанным на основе фреймворка Flask, отвечает за отображение в веб-браузере пользователя интерфейса для ввода и редактирования данных. Введенные пользователем данные передаются в Основную программу. Данное веб-приложение предоставляет возможность изменять данные регистрации. Это может быть

необходимо, если пользователь указал неудобное для себя время или ошибся при указании почтового адреса или имени.

Далее представлен рисунок (Рисунок *6*), демонстрирующий работу данной части программы:

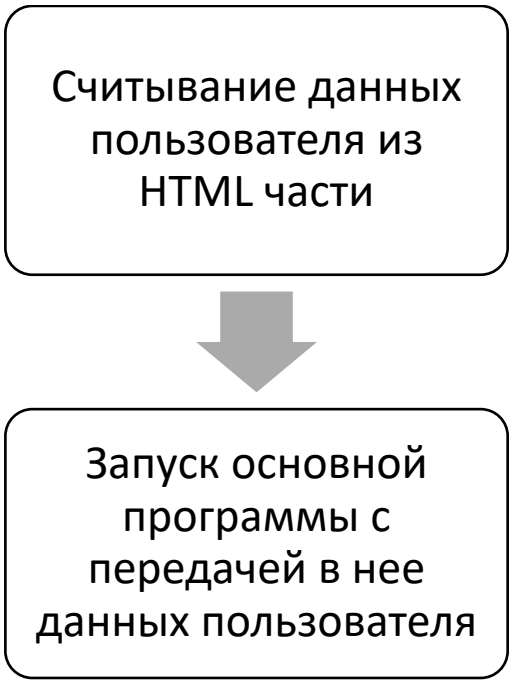

Рисунок 6 - Работа связующей программы

2.4 Основная программа

В **Основной программе** осуществляется запись к врачу по данным, которые были переданы Связующей программой.

Все действия выполняются с использованием библиотеки Selenium WebDriver. Selenium WebDriver – это программная библиотека для управления браузерами. WebDriver представляет собой драйверы для различных браузеров и клиентские библиотеки на разных языках программирования, предназначенные для управления этими драйверами.

Сфера применения библиотеки Selenium WebDriver:

Чаще всего Selenium WebDriver используется для тестирования функционала веб-сайтов/веб-ориентированных приложений. Можно выделить следующие плюсы автоматизированного тестирования веб-приложений:

- возможность проводить чаще регрессионное тестирование;
- быстрое предоставление разработчикам отчета о состоянии продукта;
- получение потенциально бесконечного числа прогонов тестов;
- обеспечение поддержки Agile и экстремальным методам разработки;
- сохранение строгой документации тестов;
- обнаружение ошибок, которые были пропущены на стадии ручного тестирования.

Функционал WebDriver позволяет использовать его не только для тестирования, но и для администрирования веб-сервисов, сократив до возможного предела количество действий, производимых вручную и это, используется в предлагаемой разработке.

Основная программа состоит из нескольких функций:

- selection_specialty - функция выбора направления;
- choice_doctor - функция выбора доктора;
- choice_time – функция выбора подходящего времени;
- choice_data – функция выбора подходящей даты;
- go_to_registration, enter_user_data - функции, которые отвечают за регистрацию;
- write_to_doctor - функция, отвечающая за активацию вышеперечисленных.

Все функции реализованы с использованием «кликов» (имитации выбор (нажатие) пользователем определенных элементов на HTML-странице).

Функция specialty_selection (a) - отвечает за выбор направления врача. После перехода на соответствующую страницу сайта, функция начнет проверку «кнопок» на соответствие с данным направлением. Поиск нужной кнопки осуществляется путем поиска элементов по XPATH:

```
By.XPATH, f'*//h3[contains(text(), "{a}")]'
```

Функция doctor_choice (target_name) - ищет на сайте необходимого доктора с помощью метода XPATH по тексту “doctor_name”

Функции data_choice и time_choce работаю вместе:

Функция data_choice - выбирает дату и проверяет, есть ли свободное время в этот день путем вызова функции time_choce. Именно поэтому в эту функцию передается 4 переменных:

– интервал дат: data_start, data_finish;

– интервал времени: time_start, time_finish.

Так же это обеспечивает «гибкость» программы, поэтому ее можно будет адаптировать для работы с другими сайтами. Самую главную роль в этом блоке выполняет последняя функция, которая обеспечивает связь между Связующей программой и функциями, которые отвечают за запись. Так же эта функция отправляет данные в программу отправки сообщения.

Далее на рисунке (Рисунок *7*) представлена работа основной программы:

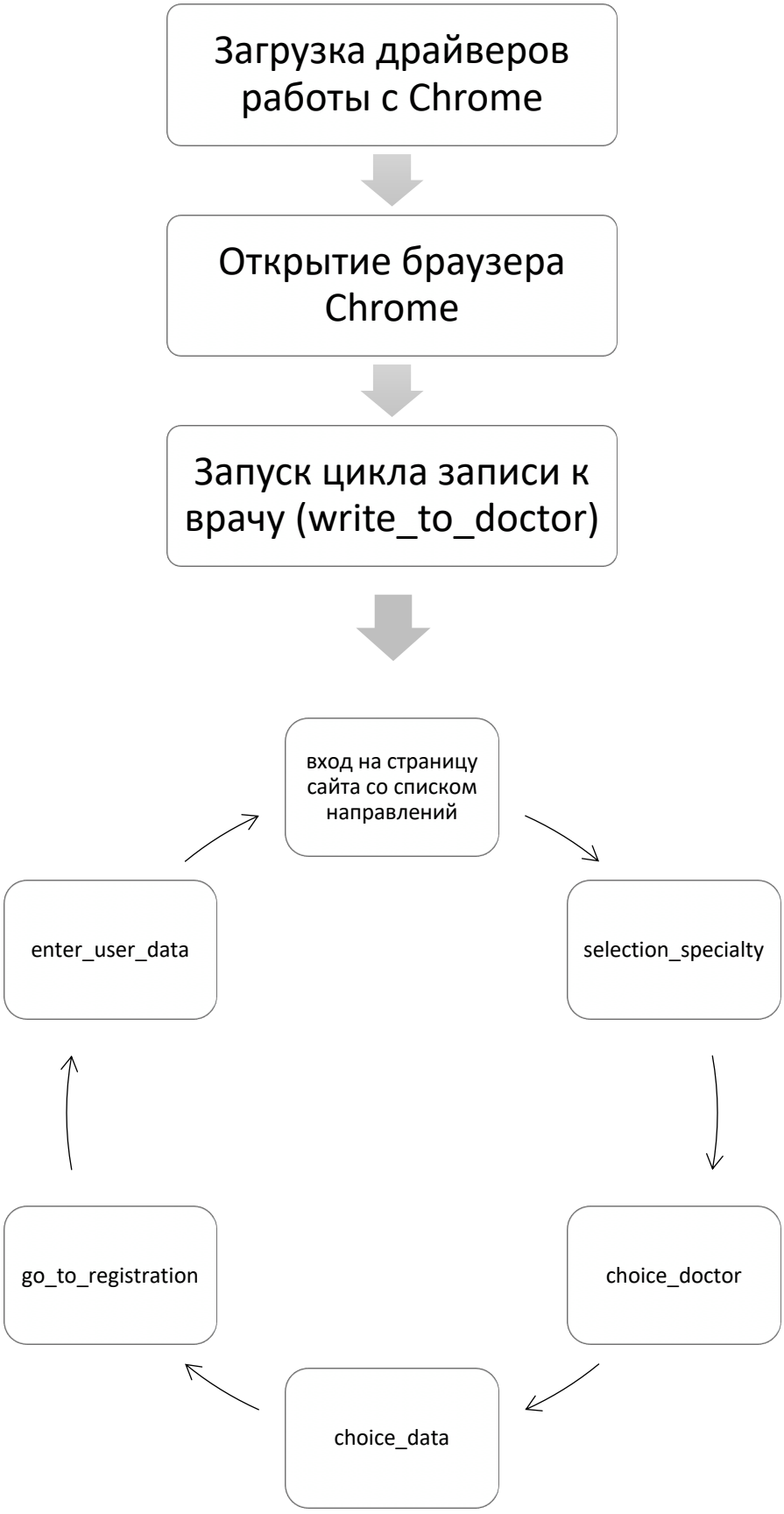

Рисунок 7 - Работа основной программы

При успешной регистрации происходит вызов программы Уведомления, для отправки сообщения пользователю с передачей в нее данных даты и времени, а так же имени пользователя и ФИО доктора.

При сбое в работе цикла выводится ошибка, которая говорит об определенной проблеме в работе программы. Это поможет разработчику при работе с программой определить место сбоя.

Виды ошибок представлены в таблице **Ошибка! Источник ссылки не найден.**.

Таблица 1- Виды ошибок

Текст ошибки	Источник ошибки
не удалось перейти на сайт	driver.get('https://zhenyanovokshanov1.wixsite.com/klinika/blank-5')
не удалось найти направление	selection_specialty(direction)
не удалось найти доктора	choice_doctor(doctor)
не удалось найти подходящее время. Проявите терпение, Вы в очереди!	data_send, time_send = choice_data(data_start, data_finish, time_start, time_finish); go_to_registration()

2.5 Уведомления

Данная часть программы предназначена, для уведомления пользователя об успешной записи. На почту, указанную пользователем, отправляется сообщение в форме:

```
"Уважаемый, {user}! Вы записаны на прием к {doctor} на {time} {day}"
```

Пользователь будет не только предупрежден о записи, но узнает дату, время и доктора, к которому записан.

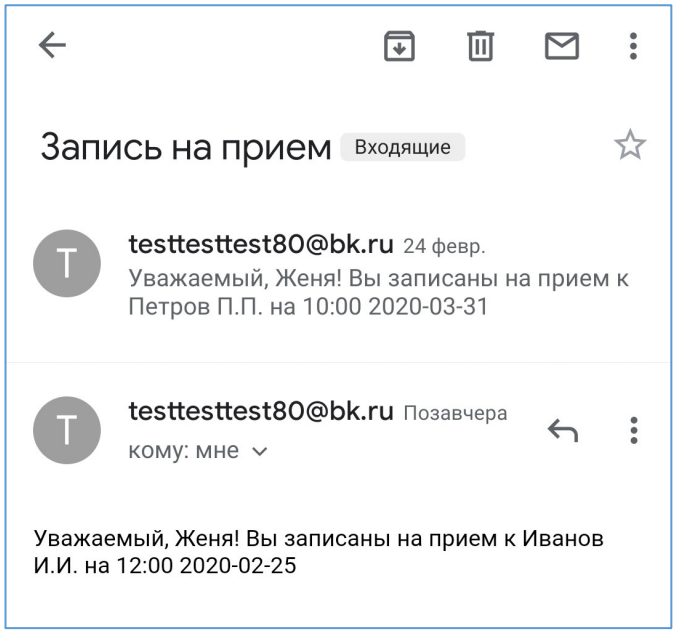

Рисунок 8 - Пример уведомления об успешной записи на прием к врачу,

Для отправки электронного сообщения, программа использует smtp-клиент, реализованный с использованием библиотеки Python 3.7 – EMAIL

3 Заключение

Разработанная программа осуществляет поиск свободного времени для записи к необходимому врачу и непосредственно, саму запись, с уведомлением пользователя по электронной почте.

За время работы над этим проектом я изучил работу веб-приложения, научился работать с библиотекой Selenium WebDriver, научился писать HTML-код страницы веб-приложения.

У меня получилась концепция приложения, которую легко можно адаптировать для любого сайта государственных услуг.

В перспективе программу можно поместить на сервере. Тогда работа основной программы не будет зависеть от наличия соединения устройства пользователя с интернетом.

4 Список использованных источников

а) Python 3. Самое необходимое / Н. А. Прохоренок, В. А. Дронов

б) Для написания программы отправки сообщений:

- https://habr.com/ru/company/pechkin/blog/281915/
- http://codius.ru/articles/Python_Как_отправить_письмо_на_электронную_почту
- https://htmlacademy.ru/courses/46/run/14

в) Для написаний HTML-кода:

- http://htmlbook.ru/faq/kak-vyrovnyat-tekst-po-tsentru
- http://htmlbook.ru/html/h1
- https://www.bestcssbuttongenerator.com/#/15

г) Для работы с Selenium WebDriver:

- https://fooobar.com/questions/45107/can-selenium-webdriver-open-browser-windows-silently-in-background
- https://selenium-python.readthedocs.io/waits.html
- https://habr.com/ru/post/250947/
- https://habr.com/ru/post/273089/
- https://habr.com/ru/post/180357/

д) Для создания веб-приложения использую библиотеку Flask:

- https://flask-russian-docs.readthedocs.io/ru/0.10.1/

5 Приложения

5.1 Приложение А

Интерфейс (HTML-код):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <style>
   .text {
    text-align:  center;
   }
    .myButton
    {
    box-shadow:inset 0px 0px 0px 0px #c4ccc0;
    background:linear-gradient(to bottom, #74ad5a 5%, #68a54b 100%);
    background-color:#74ad5a;
    border-radius:10px;
    display:inline-block;
    cursor:pointer;
    color:#ffffff;
    font-family:Times New Roman;
    font-size:23px;
    font-weight:bold;
    padding:10px 40px;
    text-decoration:none;
    }
    .btn
    {
    /* по-умолчанию для <button>, но пригодится для <a> */
    display: inline-block;
    text-align: center;
    text-decoration: none;
    /* создаём маленькие отступы, если кнопки перенесутся на две строки */
    margin: 2px 0;
    /* невидимая граница (понадобится для цвета при наведении/фокусе) */
    border: solid 1px transparent;
    border-radius: 4px;
    /* размер строится из текста и отступов (без width/height) */
    padding: 0.5em 1em;
    /* убедитесь, что достаточно контраста! */
    color: #ffffff;
    background-color: #9555af;
    }
    </style>
</head>
<body>
    <div class="text">
        <form action="" method="post">

            <h1>Автоматическая запись к врачу</h1>
            <p>
                <label for="username">Имя</label>
```

```
</p>
<p>
    <input type="text" name="username">
</p>


<p>
    <label for="email">Эл.почта</label>
</p>
<p>
    <input type="text" name="email">
</p>

<p>
    <label for="phone_number">Номер телефона</label>
</p>
<p>
    <input type="text" name="phone_number">
</p>




<p>
    <label for="direction">Направление</label>
</p>

<select name="direction">
    <option value="direction_1">Офтальмолог</option>

    <option value="direction_2">Дерматология</option>

    <option value="direction_3">Лучевая диагностика</option>

    <option value="direction_4">Женское здоровье</option>

    <option value="direction_5">Ортопедия</option>

    <option value="direction_6">Аптечная служба</option>

    <option value="direction_7">Экстренная помощь</option>

    <option value="direction_8">Внутренние болезни</option>

    <option value="direction_9">Лечение ран</option>


</select>
<p>
    <label for="doctor">Врач</label>
</p>
<select name="doctor">
    <option value="doctor_1">Иванов И.И.</option>

    <option value="doctor_2">Петров П.П.</option>



</select>
<p>
```

```
            Выберите интервал:
        </p>
        <p>
            <input type="time" name="time1">
            <input type="time" name="time2">

        </p>
        <p>
            Выберите интервал:
        </p>
        <p>
            <input type="date" name="calendar1">
            <input type="date" name="calendar2">
        </p>


            <button class="myButton" type="submit">Записаться</button>

    </form>
</div>
</body>
</html>
```

Связующая программа:

```
from flask import Flask, render_template, request
from Klinika_way import write_to_doctor
from threading import Thread


app = Flask(__name__)

directions = ['Офтальмолог', 'Дерматология', 'Лучевая диагностика', 'Женское
здоровье',
                  'Ортопедия', 'Аптечная служба', 'Экстренная помощь',
'Внутренние болезни', 'Лечение ран',]
doctors_names = ['Иванов И.И.', 'Петров П.П.',]
# ...
@app.route('/login/', methods=['post', 'get'])
def login():
    if request.method == 'POST':
        user = request.form.get('username')
        email = request.form.get('email')
        phone_number = request.form.get('phone_number')

        direction = request.form.get('direction')

        direction = directions[int(direction[-1]) - 1]
        # запрос к данным формы
        doctor = request.form.get('doctor')

        doctor = doctors_names[int(doctor[-1]) - 1]

        time1 = request.form.get('time1')
        time2 = request.form.get('time2')
```

```
        data1 = request.form.get('calendar1')
        data2 = request.form.get('calendar2')

        print(user, email, phone_number, direction, doctor, data1, data2, time1,
time2)
        Thread(target=write_to_doctor, args=(user, email, phone_number,
direction, doctor, data1, data2, time1, time2)).start()

    return render_template('login.html')


# ...

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=4567)
```

Основная программа:

```
import time
import datetime
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from webdriver_manager.chrome import ChromeDriverManager
from email_send import send_massege

WINDOW_SIZE = "1920,1080"
chrome_options = Options()
chrome_options.add_argument("--headless")
chrome_options.add_argument("--window-size=%s" % WINDOW_SIZE)
driver = webdriver.Chrome(ChromeDriverManager().install(),
chrome_options=chrome_options)


driver.implicitly_wait(20)


# ----------------------------------------------------------------------------
----------------------------
def selection_specialty(a):
    specialty_necessary = WebDriverWait(driver,
30).until(EC.visibility_of_element_located((
        By.XPATH, f'*//h3[contains(text(), "{a}")]')))
    specialty_necessary.click()
    driver.switch_to.frame(driver.find_element_by_xpath('.//iframe'))

    time.sleep(2)
    next_page_doctor_choice = driver.find_element_by_tag_name("button")
    next_page_doctor_choice.click()



# ----------------------------------------------------------------------------
----------------------------
def choice_doctor(doctor_name):
```

```
    doctor_dropdown_menu = WebDriverWait(driver,
30).until(EC.visibility_of_element_located((
        By.XPATH, './/span[@data-hook="dropdown-select-label"]'
    )))
    doctor_dropdown_menu.click()
    doctor_necessary = WebDriverWait(driver,
30).until(EC.visibility_of_element_located((
        By.XPATH, f'.//li[contains(text(), "{doctor_name}")]'
    )))
    doctor_necessary.click()




# ------------------------------------------------------------------------
-------------------------
def choice_time(time_start, time_finish):
    try:
        elements_of_time =
driver.find_elements_by_xpath('.//div//span[@class="ng-binding" and not(@data-
hook)]')
    except:
        return 0

    if len(elements_of_time) > 0:
        for element_of_time in elements_of_time:
            if time_start <= element_of_time.text <= time_finish:
                element_of_time.click()
                return element_of_time
        return 0
    else:
        return 0


# ------------------------------------------------------------------------
-------------------------
def choice_data(data_start, data_finish, time_start, time_finish):
    data_today = str(datetime.date.today())
    elem_data = [data_today]
    time = 0
    if data_start > data_finish:
        data_start, data_finish = data_finish, data_start
    if time_start > time_finish:
        time_start, time_finish = time_finish, time_start
    while  elem_data[-1] <= data_finish:
        elem_data = driver.find_elements_by_xpath('.//td//td[@data-date]')

        for i in range(len(elem_data)):
            elem_data[i] =  elem_data[i].get_attribute('data-date')
        elem_data = sorted(list(set(elem_data)))

        for data in  elem_data:
            if data_start <= data <= data_finish:
                data_possible = WebDriverWait(driver,
30).until(EC.visibility_of_element_located((
                    By.XPATH, f'*//td[@data-date="{data}"]'
                )))
                data_possible.click()
```

```
                time = choice_time(time_start, time_finish)
                if time:
                    break
                show_month = WebDriverWait(driver,
30).until(EC.visibility_of_element_located((
                    By.XPATH, f'.//div[contains(text(), "Показать месяц")]'
                )))
                show_month.click()
        else:
            driver.find_element_by_xpath('*//button [@aria-label = "next
month"]').click()

        if time:
            break
    return data, time.text


# ----------------------------------------------------------------------------
--------------------------
def go_to_registration():
    page_reistration = WebDriverWait(driver,
30).until(EC.visibility_of_element_located((
        By.XPATH,
        '/html/body/main/div/bks-app/div/div[1]/boost-calendar-
page/div[2]/boost-visitor-sidebar/section/div[1]/div/button/boost-next-step-
label/span'
    )))
    page_reistration.click()


# ----------------------------------------------------------------------------
--------------------------
def enter_user_data(user, email, phone_number):
    fields = driver.find_elements_by_tag_name('input')

    fields[0].send_keys(user)
    fields[1].send_keys(email)
    fields[2].send_keys(phone_number)

    completion_registration =
driver.find_element_by_xpath('.//div[@class="sidebar"]//span[@data-hook="next-
step-label"]')
    completion_registration.click()


# ----------------------------------------------------------------------------
--------------------------

def write_to_doctor(user, email, phone_number, direction, doctor, data_start,
data_finish, time_start, time_finish):
    while True:
        try:
            driver.get('https://zhenyanovokshanov1.wixsite.com/klinika/blank-5')
        except:
            print('Ошибка: не удалось перейти на сайт.')
        try:
            selection_specialty(direction)
        except:
            print('Ошибка: не удалось найти направление')
        try:
            choice_doctor(doctor)
```

```
        except:
            print('Ошибка: не удалось найти доктора')
        try:
            data_send, time_send = choice_data(data_start, data_finish,
time_start, time_finish)
            go_to_registration()
        except:
            print('Ошибка: не удалось найти подходящее время. Проявите терпение,
Вы в очереди!')
        try:
            enter_user_data(user, email, phone_number)

        except:

            print('Ошибка: похоже, что вы преодолели все препятствия, но не
одолели последнего босса')
        else:
            print("Регистрация прошла успешно!")
            send_massege(email, time_send, data_send, user, doctor)
            break
        time.sleep(10)
    # driver.quit()
```

Программа отправки сообщений:

```
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
def send_massege(addr_to, time, day, user, doctor):
    addr_from = "********"

    password  = "********"

    msg = MIMEMultipart()
    msg['From']    = addr_from
    msg['To']      = addr_to
    msg['Subject'] = 'Запись на прием'

    body = f"Уважаемый, {user}! Вы записаны на прием к {doctor} на {time} {day}"
    msg.attach(MIMEText(body, 'plain'))


    smtpObj = smtplib.SMTP_SSL('smtp.mail.ru', 465)
    smtpObj.login(addr_from, password)
    smtpObj.send_message(msg)
```

Пароль был скрыт по понятным причинам!

5.2 Приложение Б

Устанавливаемое ПО:

1. Интерпретатор Python 3.7
2. Библиотеки (установка с помощью pip install):
   - email
   - smtplib
   - flask
   - selenium
   - threading2
   - webdriver_manager
3. Программа(все это должно располагаться в одной папке):
   - Klinika_way.py
   - web-appa.py
   - email-send.py
   - Папка templates с расположенным в ней HTML-файлом: login.html