**Документация к проекту**

**«Центр волонтёрства»**

Цель:

Разработать цифровое решение (веб-платформу), работающее под патронатом Донецкой региональной общественной организации «Ресурсный центр поддержки добровольчества (волонтерства)»

Функции системы:

1. Регистрация пользователей

2. Авторизация

3. Контроль IP-активности

4. Чтение данных из CSV-файла

5. Сортировка предприятий и волонтёров по категориям

6. Личный кабинет волонтёра

7. Личный кабинет предприятий

Программная реализация:

- Серверная часть: Python 3.8+ (framework: flask)

- Клиентская часть: HTML / JavaScript

- База данных: SQLite3

Требования к среде:

- Python 3.8+

- SQLite3

- Современный браузер

СОДЕРЖАНИЕ

1. Описание проекта

2. Архитектура приложения

3. Описание основных функций

4. Алгоритмы обработки данных

1. Описание проекта

**Центр волонтёрства** — это веб-приложение, предназначенное для упрощения взаимодействия между волонтёрами и организаторами мероприятий.  
Проект разработан с целью автоматизации базовых процессов: регистрации, авторизации, подтверждения email, а также начальной обработки данных пользователей.  
Он может использоваться как заготовка для более масштабной системы управления волонтёрскими проектами.

**Основные функции проекта(Некоторые из них доделаны не до конца):**

|  |  |
| --- | --- |
| **Функция** | **Описание** |
| Регистрация пользователей | Если регистрируется волонтёр, то он вводит ИНН, на почту к которому привязан ИНН приходит код подтверждения. Если регистрируется партнёр, то вводит почту, на которую приходит подтверждения. |
| Авторизация | По имеющемуся логину/инн и паролю пользователь заходит на сайт. |
| Контроль IP-активности | Автоматическое освобождение IP адреса после 30 минут простоя в покое. |
| Чтение данных из CSV-файла | Чтение данных из CSV файла и их, преобразование в удобный формат. |
| Сортировка предприятий и волонтёров по категориям | У волонтёров и у предприятий есть возможность выбрать необходимые категории, что будут отображаться в поиске. |
| Личный кабинет волонтёра | В личном кабинете волонтёр имеет доступ к своему индивидумальному номеру, достижениям, истории бонусов, настройки категорий. |
| Личный кабинет предприятий | В личном кабинете предприятелей имеется информация о компании(её название, представитель, категории). Просмотр бонусов, которые предоставляет компания. Социальные сети компании. |

**Структура проекта**

Проект структурирован по модульному принципу, обеспечивая ясность и удобство в навигации:

* main.py – основной серверный скрипт на Flask. Обрабатывает маршруты регистрации, авторизации, отправки кодов

подтверждения и другие заыпросы от клиента.

* assets/ – каталог со статическими ресурсами:
  + css/ – содержит стили для различных страниц (style\_login.css, style\_main.css, style\_profile.css).
  + js/ – JavaScript-файлы, реализующие клиентскую логику:
    - login.js – отвечает за авторизацию;
    - reg.js – реализует регистрацию, отправку кода и подтверждение;
    - main.js – обработка основной логики после входа.
* html/ – основная часть клиентского интерфейса:
  + подкаталоги par и vol содержат страницы для партнёров и волонтёров соответственно;
  + основные шаблоны: login.html, reg.html, main\_vol.html, main\_par.html, а также страницы профиля и истории.
* db\_vol\_part.db – база данных SQLite, в которой хранятся данные зарегистрированных пользователей (партнёров и волонтёров).
* тестовые данные.CSV – используется для первичной верификации ИНН волонтёров на этапе регистрации.

Регистрация волонтёров и партнёров реализована с верификацией через email:

* При регистрации волонтёра используется ИНН, который сверяется с CSV-файлом, и на привязанную к нему почту отправляется код.
* При регистрации партнёра используется login/email/password – код отправляется на указанный email.
* Код подтверждения отображается в модальном окне, имитируя реальную отправку. После ввода кода пользователь добавляется в БД.

Форма регистрации использует JavaScript (файл reg.js) для отправки данных на сервер через fetch() с заголовками MyType, MyINN, MyLogin, MyEmail и другими в зависимости от роли пользователя.

2. Описание модулей проекта

📄 main.py — серверное приложение

Роль:

- Запуск Flask-сервера;

- Обработка маршрутов (роутов);

- Отправка email-кодов подтверждения;

- Хранение и проверка IP-активности;

- Чтение данных из CSV.

Содержит:

- Импорт библиотек (Flask, Flask-Mail, csv, random, time);

- Инициализацию приложения;

- Конфигурацию SMTP для отправки писем;

- Основные переменные (тайм-ауты, список свободных страниц, список IP);

- Функции: update\_time(), send\_code\_to() и другие;

- Маршруты @app.route(...) для каждой страницы.

📁 test1/assets/ — клиентская часть

Роль:

Папка со статическими файлами (frontend), которые отдаются пользователю браузером.

🔸 HTML-файлы:

- login.html — форма входа;

- reg.html — форма регистрации;

- main.html — основная страница, доступна после входа.

🔸 CSS/JS:

- style.css — стили для внешнего вида страниц;

- script.js (если есть) — скрипты для обработки форм или валидации данных на стороне клиента.

Как подключается:

Flask конфигурируется так, чтобы отдавать файлы из этой папки:

app = Flask(\_\_name\_\_, static\_folder="./test1/assets")

🔒 Авторизация и IP-контроль

В проекте Volunteer Platform используется упрощённая система авторизации, сочетающая подтверждение через email и контроль активности по IP-адресам. Это сделано для базовой защиты от злоупотреблений и симуляции поведения полноценной сессии.

📥 Процесс авторизации

Пользователь переходит на страницу входа (/login.html) или регистрации (/reg.html).

Вводит email-адрес, после чего вызывается функция send\_code\_to(email).

На почту отправляется 8-значный код. Пользователь должен ввести его на сайте для подтверждения.

После успешного ввода кода IP-адрес пользователя заносится в список активных (db\_ip), и ему открывается доступ к защищённым страницам.

🌐 Контроль активности по IP

Для обеспечения простейшей защиты от злоупотреблений и автоматизации сессий используется механизм слежения за IP-адресами. Он реализован через переменную:

python

Копировать

Редактировать

db\_ip = [] # список IP-адресов с временными метками

Каждый элемент в db\_ip — это пара [ip-адрес, timestamp].

🔁 Обновление активности

При каждом запросе от пользователя вызывается функция:

python

Копировать

Редактировать

def update\_time(ip):

...

Функция выполняет:

Удаление IP-адресов, у которых время последней активности превышает time\_prost минут (по умолчанию 30).

Обновление времени текущего IP, если он найден.

Возврат True, если IP найден и обновлён, иначе False.

⏱️ Порог активности

Значение time\_prost = 30 означает, что если пользователь не совершал никаких действий более 30 минут, его IP будет удалён из списка активных, и он потеряет доступ к закрытым страницам.

🔐 Открытые и защищённые страницы

В коде задаётся список так называемых «свободных» маршрутов:

python

Копировать

Редактировать

sp\_svob\_fail = ["/login.html", "/reg.html", "/main.html"]

Эти страницы доступны даже без подтверждённого IP. Остальные — например, подгружаемые части интерфейса или API-запросы — могут проверяться на наличие IP в списке db\_ip.