**Титульный лист**

# Проект по теме: *Автоматизированная система удаленного управления теплицей.*

# Над анализом и реализацией всей работы, трудились ученики 10 “A” класса ГБОУ Школа 1532.

# Также для более быстрого выполнения поставленной перед нами задачи, мы распределили на каждого часть работы.

# *Воронов Никита Рустамович* - разработал приложение

# Мандрыка Арина Юрьевна, Шершина Анна Исаевна *-* Сделали веб-интерфейс

# *Алимов Альберт Радикович,* Донченко Дмитрий Антонович - Подготовили документацию и презентацию

**Анализ технических требований**

Далее прочитав условие **“**Юные фермеры решили автоматизировать свою теплицу, установив в нее автоматизированную систему контроля и управления климатом.”

*Мы начали рассматривать и обсуждать какой продукт у нас должен получиться по итогу. Так, например, когда мы говорили по поводу приложения, нами было заострено внимание на техническом задание, где было множество пунктов для выполнения.*

1. Отображать динамику и последние данные с каждого датчика температуры и влажности, как в табличном виде, так и в виде графиков. Данные формировать в рамках текущей рабочей сессии. Период демонстрации определяется участником.

По этой причине в приложение был добавлен раздел ‘Home’, где расположены кнопки для просмотра температуры и влажности воздуха с датчиков, а также, нажав на ‘soil info’, мы просматриваем данные о влажности почвы.

1. Отображать динамику и последние данные средней температуры и средней влажности почвы и в теплице, как в табличном виде, так и в виде графиков.

Показатели средней температуры и влажности почвы и в теплице были выше описаны, а для отображения этих данных мы предусмотрели таблицу в разделе “Table”.

1. Открыть или закрыть полив конкретной бороздки, форточки, общее увлажнение в теплице в теплице нажатием на специальные кнопки, а также просматривать текущий статус.

Для реализации этого пункта мы разработали страницу, зайти на которую можно через “Home”/ “DO”.

Также мы предусмотрели выполнение следующих условий:

1. Если средняя температура в теплице больше, чем *T*градусов, кнопка открытия форточек активна, в противном случае нажатие невозможно.
2. Если средняя влажность в теплице меньше, чем *H %*, кнопка открытия общей системы увлажнения активна, в противном случае нажатие невозможно.
3. Если средняя влажность почвы в конкретной бороздке теплицы меньше, чем *Hb%*, кнопка открытия системы полива активна, в противном случае нажатие невозможно.

Выполняя эти пункты в приложение, мы задаем условия, при котором, если температура, влажность и средняя влажность в борозде будет достигать какого-то значения, то форточки, системы увлажнения, системы полива будут неактивны. В этом нам помогает раздел “Edit” в нашем приложении.

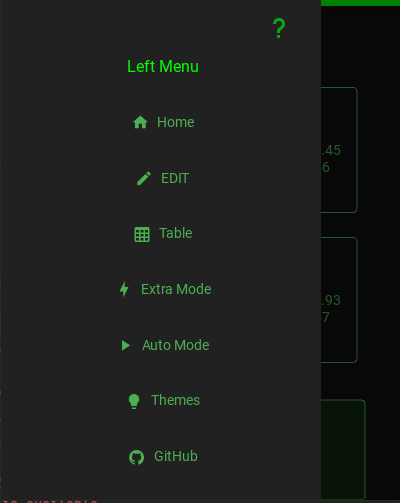
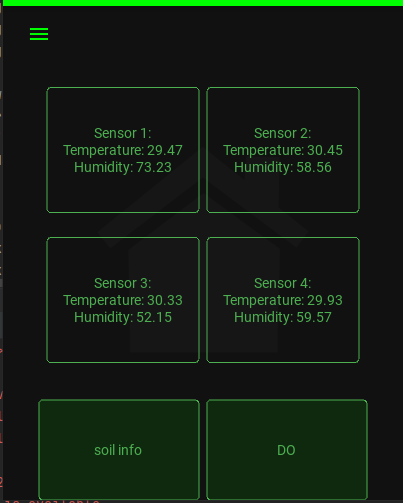
1. Предусмотреть переход в режим экстренного управления, при котором возможно нажатие всех описанных выше кнопок даже если условия не соблюдены.

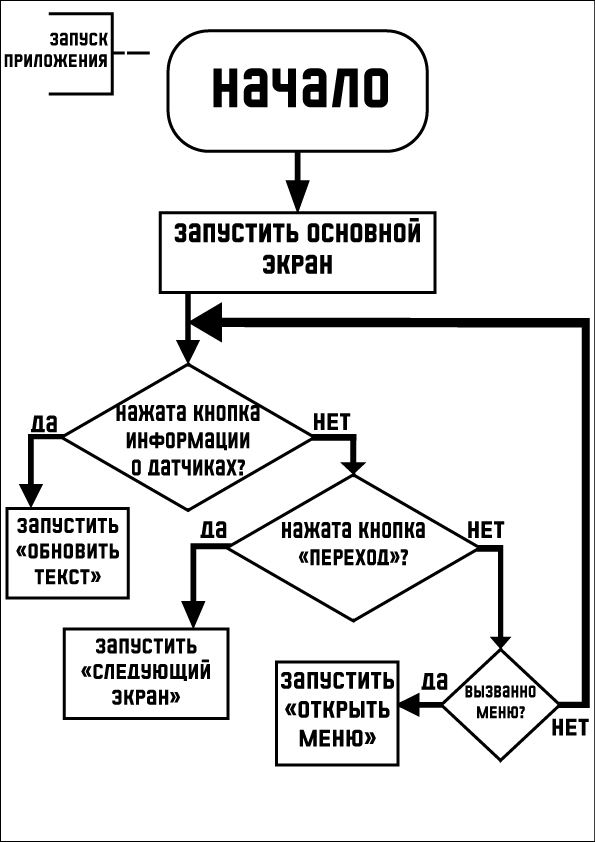
В ‘Extra Mode’ не имеет значение, при каких ограничениях нельзя открывать форточку или систему полива, ведь вы берете управление в свои руки. Важно, что в этом режиме можно, как навредить, так и сделать лучше.

**Обоснование выбора языка программирования и используемых программных средств.**

При первичном выборе языка программирования мы отталкивались от того, кто на чем умеет реализовывать различного рода функции. Выбор пал на Python. Python – весьма простой в освоение и понимании язык. Конечно, он не подходит для создания больших, оптимизированных проектов, но со своей задачей справляется отлично. К тому же Python находится на первых местах по популярности среди других языков программирования.

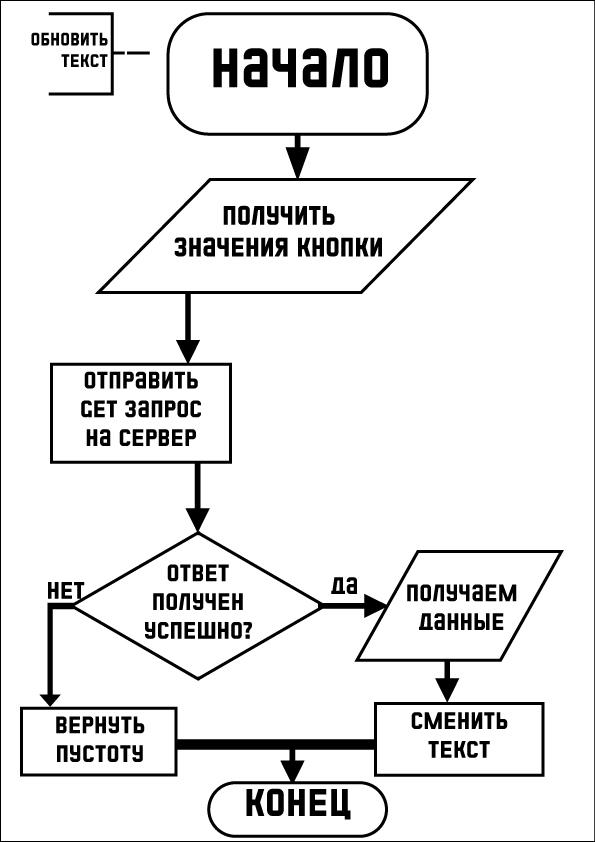
В качестве основной библиотеки для создания приложений (будь то веб, мобильное или десктопное) можно выделить 3 основные: Flask, Kivy, PyQt. Делать десктопное приложение под наши задачи не особо эффективно, а работать только в веб не совсем удобно для обычного пользователя. Поэтому варианты Flask и PyQt отпадают. Остается Kivy. У него очень большое количество преимуществ: адаптивность, что является главным плюсом данной библиотеки; программировать можно на чистом Python без дополнительных языков программирования. Так же хочется отметить невероятное количество уже готовых решений большинства вопросов. Приложения на Kivy получаются красивыми, имеют не такой большой размер и хорошо контактирует с другими библиотеками. Среди самых известных примеров приложений написанных на Kivy можно выделить: Guess The Quote – угадай цитату, 2048 – игра по складыванию одинаковых чисел, Say This – приложение преобразование речи в текст и обратно, и другое.

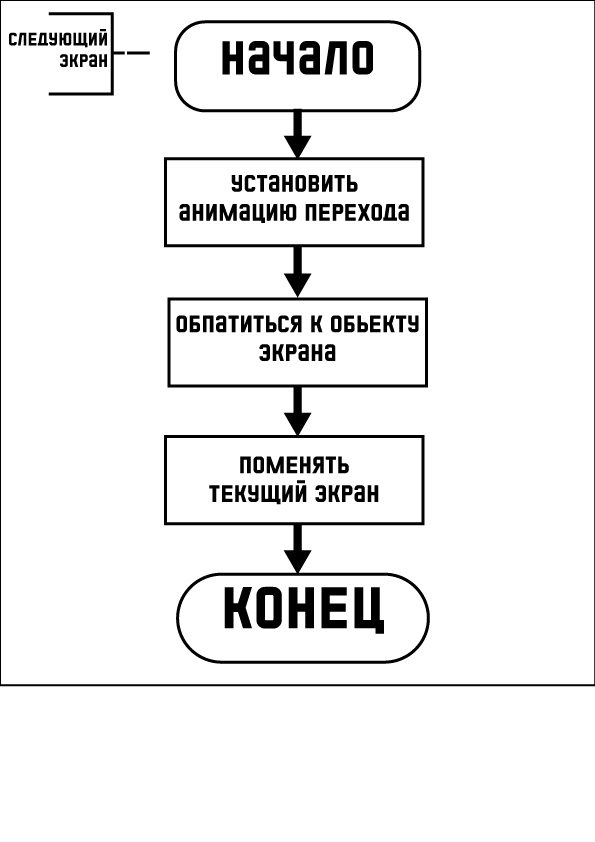


**Структурная и функциональная схемы программного** **продукта**

“Запуск приложения” срабатывает только при открытии приложения, конечно, можно вернуться в это же окно повторно, но это будет уже вызов другой функции.

**Блок-схема работы основного алгоритма**

****

****

Важно понимать, что это не все функции приложения, а лишь небольшая часть, где описано самое главное. Весь основной алгоритм функций – это есть вся программа. Функция “следующий экран” работает благодаря классу *ScreenManager* из Kivy, который отвечает за смену экранов и их отображение в приложении. А функция “обновить текст” работает с помощью json и requests путем отправки и обработки запросов сервера, но для того, чтобы показать текст используется класс MDLabel, отвечающий за показ текста и изменении его составляющей (цвет, размер, сам текст, цвет линии вокруг блока текста и другое).

**Описание проведенных испытаний в соответствии с регламентом кейса.**

Начало испытаний – 09.02.2023

*Испытание №1* – проверка работоспособности кнопок, экранов, заголовков и переходов в приложении. Запуск осуществился на платформе Windows 10 с использованием Python: Kivy. Итог – положительный отклик всех компонентов основного и второстепенного меню;

*Испытание №2* – проверка корректности получения ответа от сервера, путем отправки get и patch запросов, преобразование полученных данных в первичный текст. Запуск осуществился на платформе Windows 10 с использованием Python: requests, Kivy. Итог- положительный ответ с get запросами, данные обработались корректно, patch запросы выдали ошибку 404, данные не обработаны. Возможная ошибка – ошибка в некорректном адресе доставки.

*Испытание №3* – проверка корректности «левого меню», попытка отправить patch запросы другим способом, проверка работоспособности нового интерфейса программы. Первый запуск сайта. Запуск осуществился на платформе Windows 10 с использованием Python: requests, kivy, json, webbrowser; HTML и CSS. Итог – положительный результат в работе «левого меню» - все кнопки работают стабильно. Положительный ответ от сервера при отправке patch запроса (статус код 200), как мы и предполагали ошибка, была в ссылке запроса. Большая часть нового интерфейса работает, есть ошибки в цвете текста и иконках. Сайт запускается, вся информация доступна.

*Испытание №4* – тестирование дополнительных функций приложения: таблица, режим ввода данных, режим экстренных действий, открытие репозитория при нажатии на соответствующие кнопки; Тестирование новых функций сайта. Запуск осуществился на платформе Windows 10 с использованием Python: requests, kivy, json, webbrowser; HTML и CSS. Итог – ошибка в правильности заполнения таблицы, скорее всего запросы не успевают обрабатываться; ошибка в сохранении данных, скорее всего это неправильно подобранная переменная; положительный ответ кнопок и переключателей в режиме экстренных действий и положительный ответ при нажатии на кнопку открытия репозитория. Плохая читабельность текста на сайте, требуется подобрать шрифт и цвет. Исправлена ошибка с некорректным интерфейсом программы.

*Испытание №5* – тестирование новых функций приложения: режим автоматических действий (время, переключатели), возможность смены темы приложения на светлую и обратно. Тестирование читабельности сайта. Запуск осуществился на платформе Windows 10 с использованием Python: requests, kivy, json, webbrowser; HTML и CSS. Итог – корректная работа режима автоматических действий (переключатели, текст), но ошибка в работе циферблата (некорректное постановка времени), скорее всего это из-за неверного определения формата времени). Возможность смены темы приложения работает корректно, однако имеются недочеты в читабельности текста на светлом фоне. Читабельность сайта хорошая, понятная. Исправлена ошибка с сохранением данных путем объявления глобальных переменных, исправление заполнения таблицы путем распаковки полученных данных в python словарь и сортировки значений по кортежам.

*Испытание №6* – тестирование всех функций приложения на мобильном устройстве, тестирование большого количества новых функций сайта. Запуск осуществлялся на платформе Windows 10 с использованием HTML и CSS, Android 12.0 с использованием андроид эмулятора Linux – Termux. Итог – большая часть новых функций сайта работает корректно, есть недочеты с переходами. Испытание приложение – удачное, однако присутствуют крайне неприятные недочеты: слабая оптимизация приложения, приложение не адаптировано под разные форматы экранов. Исправления в цветах теста для большей читабельности и разделения интерфейса на классы важности.

*Испытание №7* – тестирование сайта на хостинг-сервере, тестирование 1 этапа оптимизации приложения. Запуск осуществлялся на платформе Windows 10 с использованием HTML, CSS, JS и хост-сервера; Android 12.0 с использованием андроид эмулятора Linux – Termux. В оптимизацию вошли: сокращение количества импортов библиотек и написание собственных, более оптимизированных функций; сокращение количества файлов путем объединения их в один. Для большего «оживления» в качестве заднего плана были добавлены изображения, характеризующие данный экран. Итог – благодаря оптимизации было сокращено 10 % общих ресурсов приложения. После открытия сайта на смартфоне был сделан вывод о необходимости адаптивности сайта под мобильные устройства.

*Испытание №8* – тестирование приложения, как APK файл для Android устройств. Запуск осуществлялся на платформе Android 12.0 без дополнительного программного обеспеченья. Итог – положительный. Приложение полностью работоспособно, все функции работают корректно. Был добавлен второй основной экран, срабатывающий при различных ошибках.

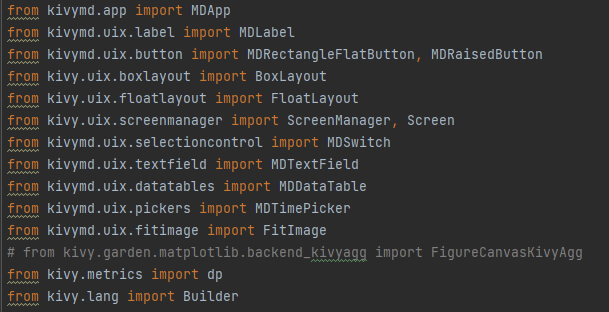
*Испытание №9* – финальное тестирование приложения и сайта. Итог – все функции приложения и сайта работают корректно, ошибки не выявлены, результат положительный.

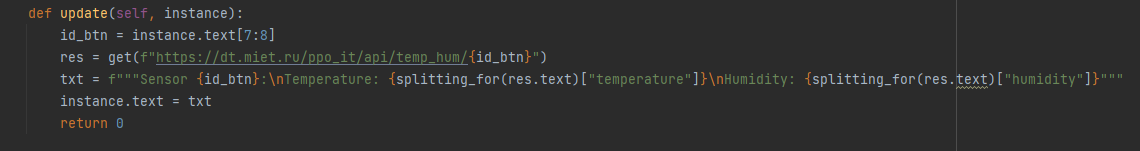
Полное окончание испытаний – 18.02.2023.

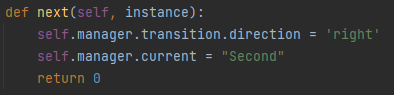
**Программный код**

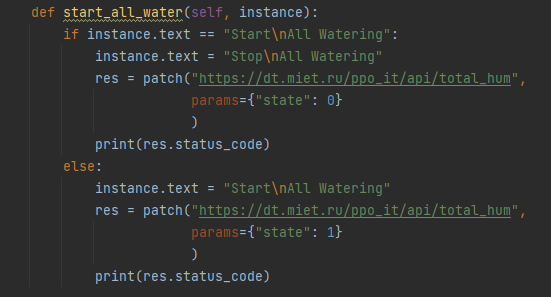
Как мы ранее уже говорили для написания основной части кода нами был задействован *python kivy*. И сейчас мы вам продемонстрируем части готового кода.

1 – Задействованные библиотеки



2 – Обновление текста по нажатию на соответствующий экземпляр кнопки

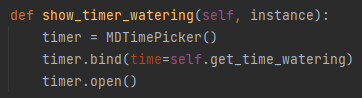
3 – Задействование других экранов путем обращения метода к менеджеру экранов

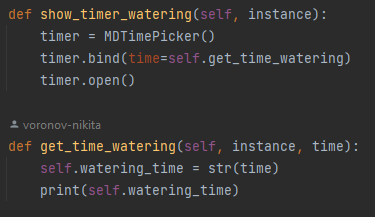
 4 - Отправляем patch запрос на старт полива всех грядок, где в качестве параметра указываем 0 – если требуется прекратить полив, 1 – если начать соответственно.

5 - Отправляем patch запрос на открытие/закрытие форточки, где параметр “state”:1 – открывает форточку, а параметр “state”:0 – закрывает соответственно.

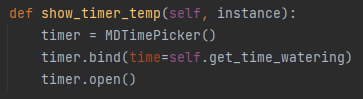


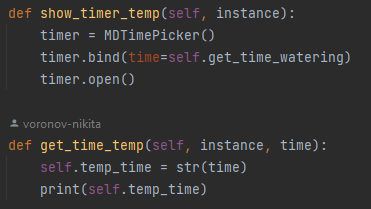
6 – Автоматический полив с использование таймера.





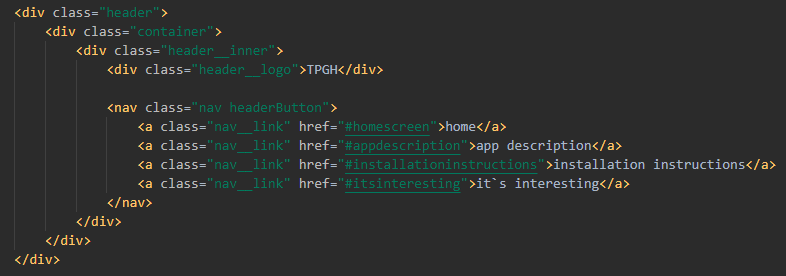
7 – Автоматический регулятор температуры с использование таймера.



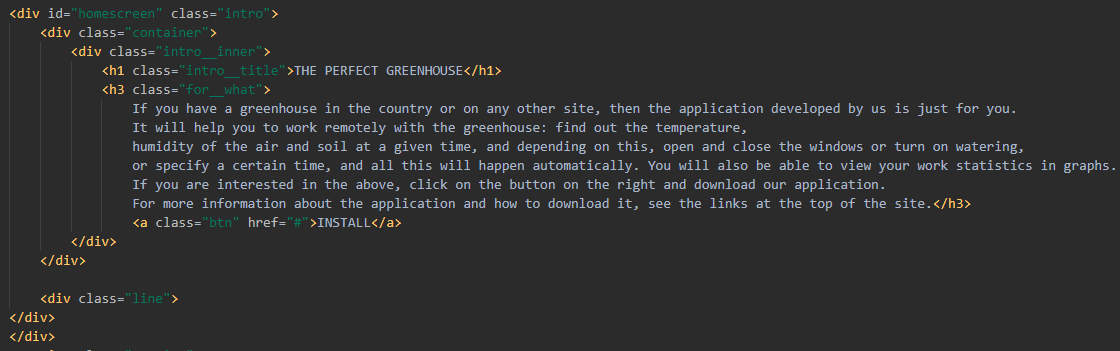


Части кода для создания сайта. При его создании мы работали с HTML и CSS:

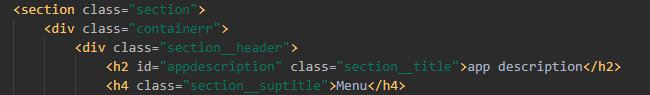
1 – Создание header сайта, в котором есть логотип и ссылки на отдельные части сайта.



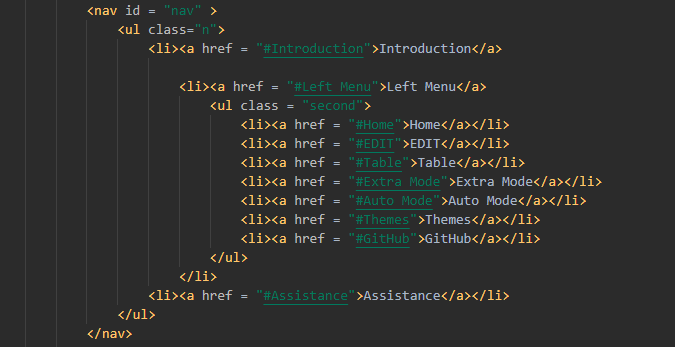
2 – Создание вступительной страницы сайта, где можно скачать приложение.



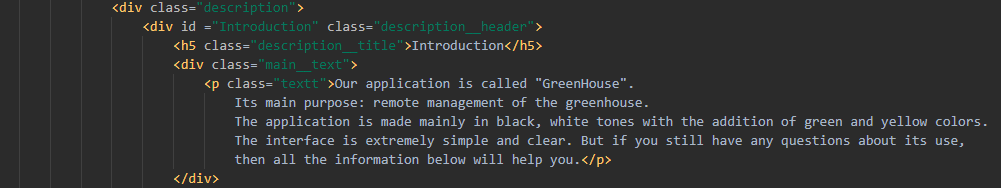
3 – Новая секция для описания приложения.



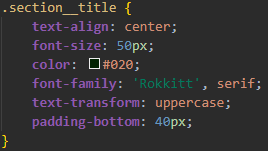
4 – Выпадающее меню для работы с данной секцией.



5 – Описание страниц приложения.



6 – Пример кода оформления сайта в CSS.



7 – Медиа запрос для адаптации сайта

