**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.
2. Отображать динамику и последние данные средней температуры и средней влажности в теплице, как в табличном виде, так и в виде графиков.
3. Открыть или закрыть форточки в теплице нажатием на специальные кнопки, а также просматривать текущий статус.
4. Отображать динамику и последние данные с каждого датчика влажности почвы, как в табличном виде, так и в виде графиков.
5. Открыть или закрыть полив конкретной бороздки в теплице нажатием на специальные кнопки, а также просматривать текущий статус.
6. Открыть или закрыть общее увлажнение теплицы нажатием на специальные кнопки, а также просматривать текущий статус.

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.
4. Предусмотреть режим настройки системы и указания параметров: *T*, *H*, *Hb*.
5. Предусмотреть переход в режим экстренного управления, при котором возможно нажатие всех описанных выше кнопок даже если условия не соблюдены.

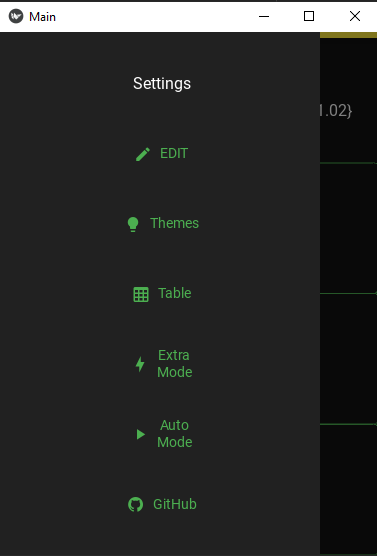
**Обоснование выбора языка программирования**

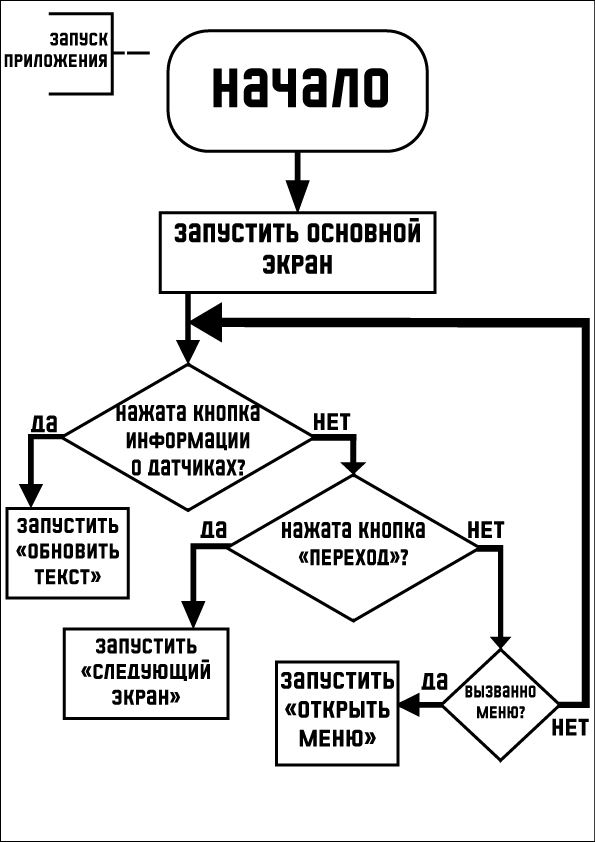
**и используемых программных средств**

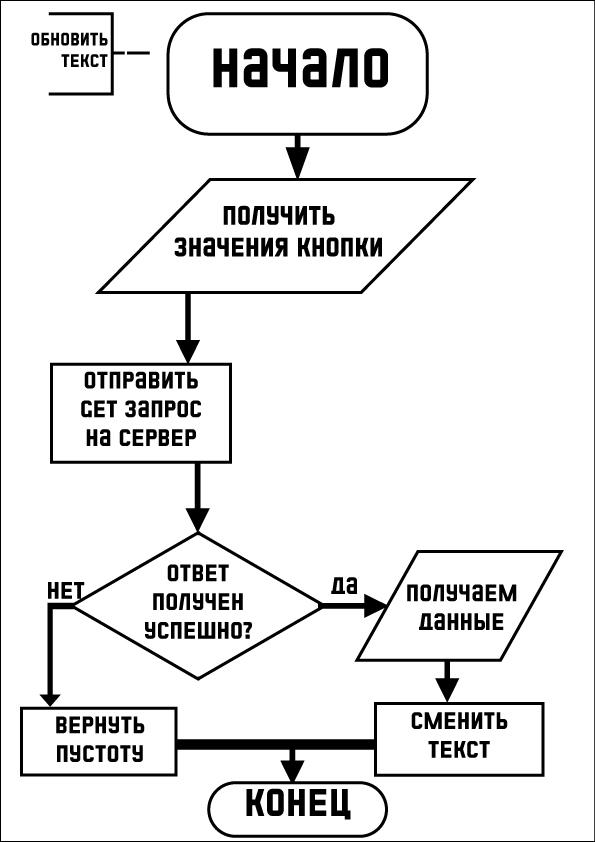
На сегодняшний день на рынке информационных технологий существует большое количество различных средств разработки приложений и веб – сайтов.

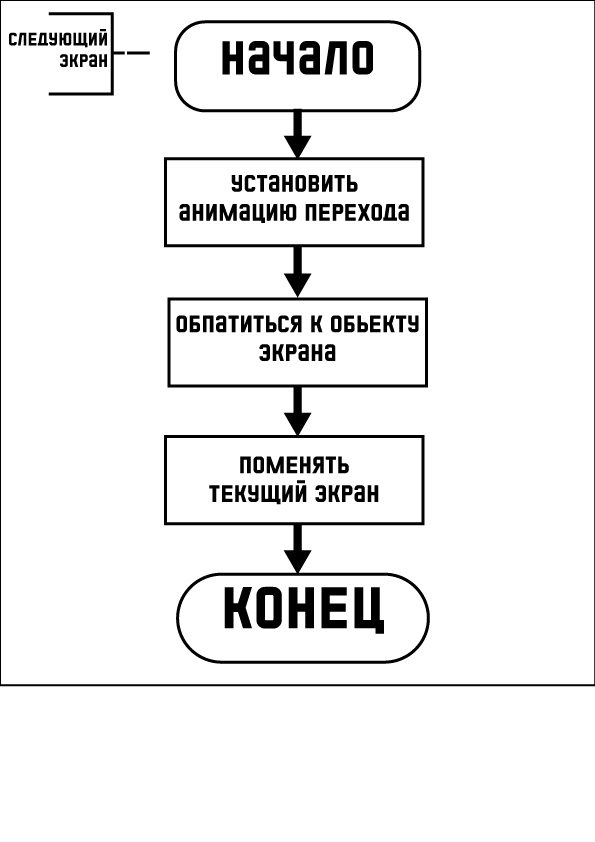
*Kivy* — это библиотека *Python*, имеющая открытый код, предназначенная для разработки кроссплатформенных GUI приложений. Она позволяет писать вам приложения с графическим интерфейсом на чистом Python, которые работают на основных платформах (Windows, Linux, MacOS, Android, IOS).

Фреймворк Kivy разрабатывается с 2011 года. С момента релиза на его основе было построено много проектов под платформу Андирод. Если сравнивать *Kivy* с другими библиотеками языка Питон по набору функций, то среди крупных конкурентов можно выделить PyQT. Все прочие библиотеки явно будут уступать по функционалу.

**

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

**Блок-схема работы основного алгоритма**

****

**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.

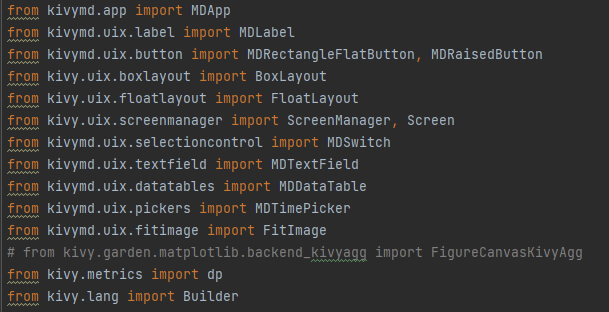
Испытание №8 – тестирование приложения, как APK файл для Android устройств. Запуск осуществлялся на платформе Android 12.0 без дополнительного программного обеспеченья. Итог – положительный. Приложение полностью работоспособно, все функции работают корректно.

Полное окончание испытаний – 18.02.2023.

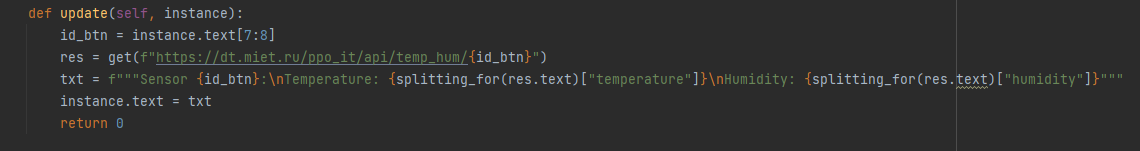
**Программный код**

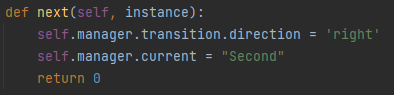
Как мы ранее уже говорили для написания основной части кода нами был задействован *python kivy*. И сейчас мы вам продемонстрируем части готового кода.

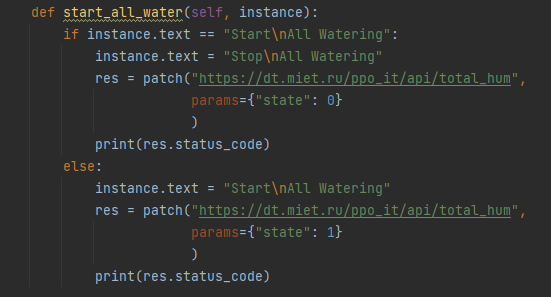
1 – Задействованные библиотеки



2 – Обновление текста по нажатию на соответствующий экземпляр кнопки



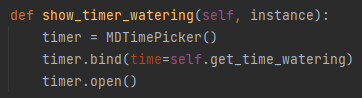
3 – Задействование других экранов путем обращения метода к менеджеру экранов

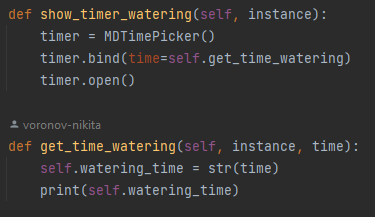
 4 - Отправляем patch запрос на старт полива всех грядок, где в качестве параметра указываем 0 – если требуется прекратить полив, 1 – если начать соответственно.

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

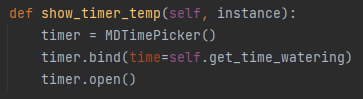


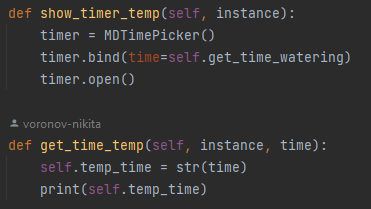
6 – Автоматический полив с использование таймера.





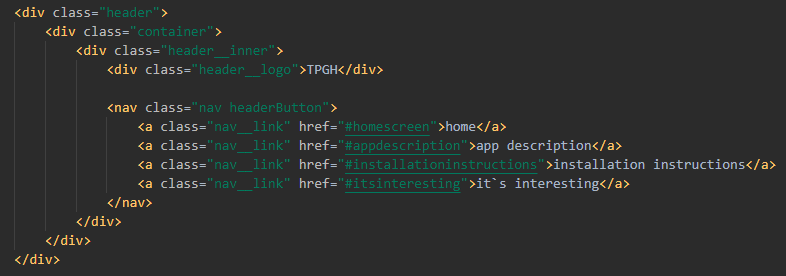
7 – Автоматический регулятор температуры с использование таймера.



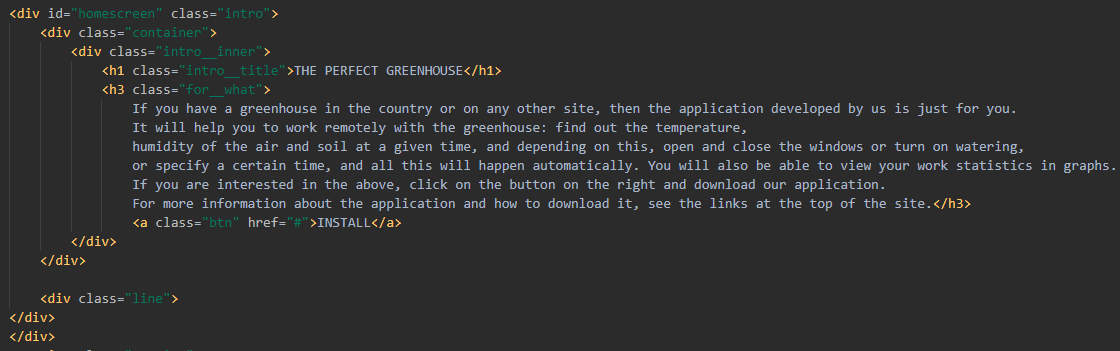


Части кода для создания сайта. При его создание мы работали с HTML и CSS:

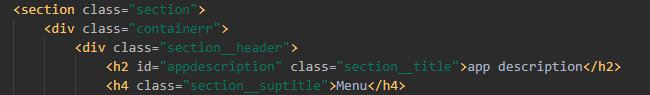
1 – Создание header сайта, в котором есть логотип и ссылки на отдельные части сайта.



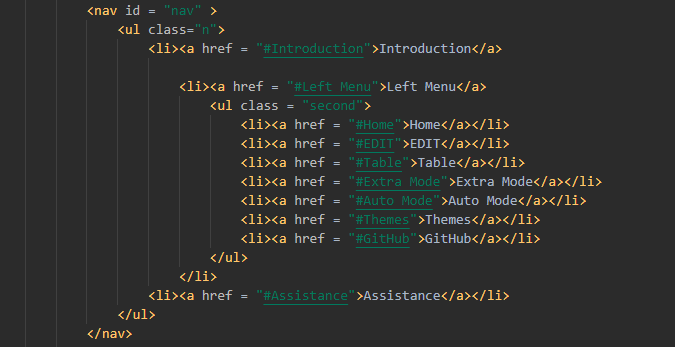
2 – Создание вступительной страницы сайта, где можно скачать приложение.



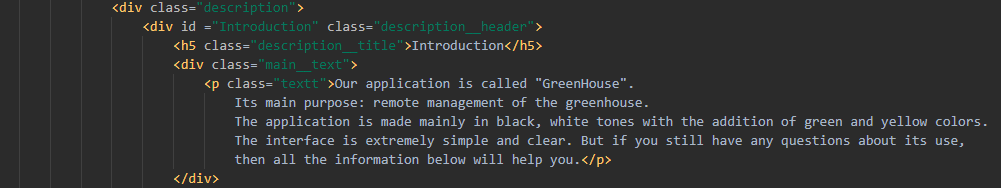
3 – Новая секция для описания приложения.



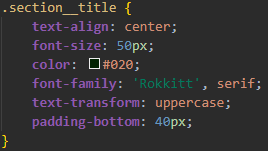
4 – Выпадающее меню для работы с данной секцией.



5 – Описание страниц приложения.



6 – Пример кода оформления сайта в CSS.



7 – Медиа запрос для адаптации сайта

