Информационная система управления тепличными комплексами с применением элементов нечёткой логики

Д. Е. Хныкин, email: hnykin_d_e@sc.vsu.ru
Н. К. Самойлов, email: nk.samoylov@gmail.com

Федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный университет»

Аннотация. *В данной работе рассматривается информационная система управления тепличными комплексами с применением элементов прогнозирования, на основе нечёткой логики. Принцип работы связан с взаимодействием аппаратной, серверной и клиентской части с использованием интерфейса RESTful API.*

Ключевые слова: *Информационная система, тепличный комплекс, теплица, автоматизация процессов, микроконтроллер, информационная система управления тепличными комплексами с применением элементов нечёткой логики.*

Введение

Подавляющее большинство тепличных комплексов не оснащены логикой, с помощью которой можно было бы управлять автоматикой теплиц дистанционно, без физического участия оператора. Таким образом, разработка информационной системы, обеспечивающей оснащение тепличного комплекса устройством для сбора данных их передачу, обработку, анализ и управления в следствии автоматикой теплиц представляет собой актуальную задачу. Устройство (управляющая плата и набор периферийных устройств) должно быть выполнено в компактном корпусе, с учетом особенностей места расположения его в теплицах, т.е. должно быть устойчиво, как минимум к брызгам воды и условиям повышенной влажности, что соответствует международному стандарту степени защиты (International Protection) IP54 (ГОСТ 14254-2015). [1] При этом необходимо предусмотреть своевременный сбор и обработку информации с периферийных устройств с целью прогнозирования климата внутри теплиц, в том числе и для экономии ресурсов.

1. Цель исследования

Разработка функционирующего макета устройства, которое обеспечило бы сбор, обработку, передачу данных и управление теплицами в тепличном комплексе.

2. Принцип работы информационной системы

Принцип работы информационной системы основывается на реализации и взаимодействии трех ключевых частей:

- Аппаратная часть: выполненное на печатной плате устройство, управление которым осуществляет контроллер на базе процессора с RISC-архитектурой, и его обвязка в виде периферийных датчиков, для сбора информации
- Серверная часть: набор микросервисов реализованный на базе Spring Framework для обработки информации.
- Клиентская часть: клиент для управления системой, в виде Android-приложения, для просмотра состояний и управления теплицами.

Печатаная плата для аппаратной части информационной системы разработана самостоятельно, разведена в среде автоматизации проектирования электроники включающая в себя редактор принципиальных схем – EasyEDA, выполнена на фольгированном стеклотекстолите, места под компоненты помечены с помощью шелкографии, как показано на рис. 1.

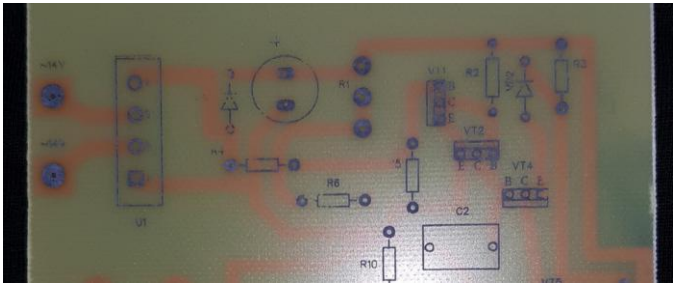

Рис. 1. Шелкография на печатной плате

Серверная часть приложения написана на языке Java с использованием Spring Boot Framework, включая технологии Spring Security и Spring Data. [2-3] Внутренняя архитектура приложения состоит из трех уровней: слоя контроллеров, слоя сервисов и слоя репозиториев.

Слой контроллеров обеспечивает взаимодействие сервера с клиентом с помощью REST API, как показано на рис. 2. Задача

контроллера – получить HTTP-запрос от клиента, передать его в слой сервиса и после того, как будет произведена необходимая логика с данными запроса, контроллер формирует и возвращает клиенту HTTP-ответ. Слой сервисов отвечает за всю бизнес-логику приложения. В них реализована вся функциональность, используемая приложением для манипуляции данными, как приходящих из запросов клиента, так и из базы данных. [4] На рис. 3 показана схема архитектуры приложения.

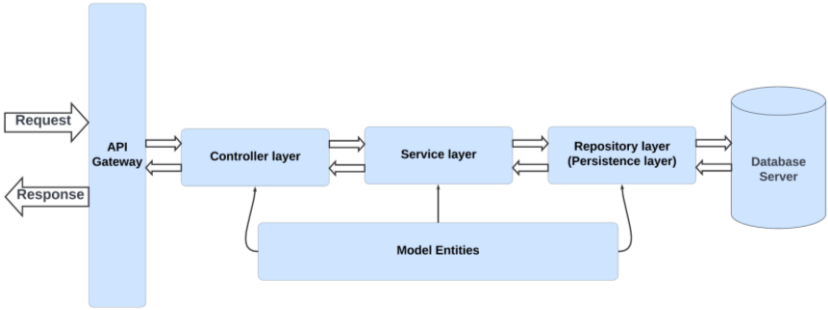

Рис. 2. Схема внутренней REST API архитектуры приложения

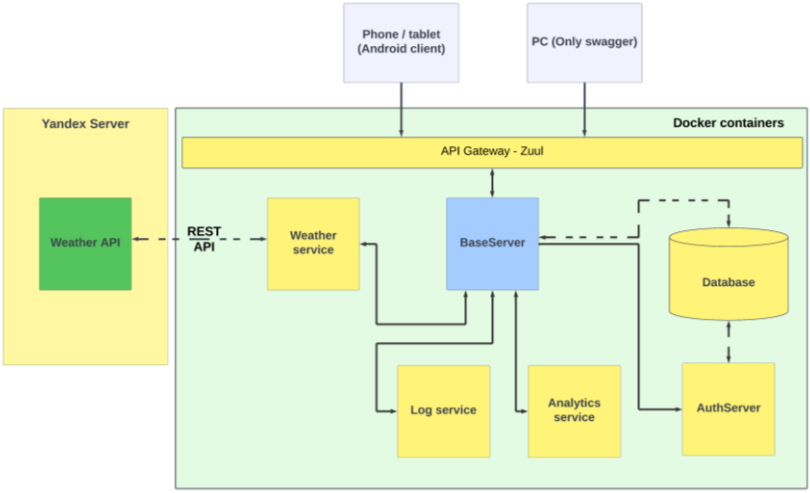

Рис. 3. Схема архитектуры приложения

Описание компонентов:

- Weather API – api от компании Яндекс, для получения информации о погоде;
- Weather Server – внутренний сервис для работы с данными о погоде;
- Auth Server – решение на основе сервера Keycloack;
- API Gateway – основанный на JVM маршрутизатор и серверный балансировщик нагрузки. Необходим для получения единой точки входа для сервисов;
- database – в качестве базы данных будет использоваться PostgreSQL;

каждый микросервис и/или компонент имеет свою схему для минимальной связности.

Клиентская часть информационной системы, выполнена в виде android-приложения.

Вариант использования информационной системы. На комплекс теплиц, или на единичных теплицах, а так же на открытых участках почвы, с засеянными на них культурами, за которыми требуется уход, размещается одно управляющее устройство, выполненное в рамках данной работы, которое имеет корпус, защищенный по стандарту IP54, во избежание попадания влаги на электронные компоненты. И опциональный набор дополнительных устройств, таких как: насос, ультрафиолетовая лампа, увлажнители воздуха и т.д. Управляющее устройство имеет набор датчиков (конфигурируется в зависимости от места применения) и доступ к сети интернет. Устройство получает данные с датчиков, которые обрабатываются микроконтроллером и отправляются на общий сервер, где происходит их анализ и обработка. [5-6]

Таким образом, на сервере осуществляется конвергенция двух потоков информации:

- Поток информации с датчиков теплицы
- Поток информации с серверов сервиса Яндекс.Погода.

Анализ и взаимодействие с информацией этих двух потоков и элементов нечеткой логики на стороне сервера и позволяет прогнозировать и выстраивать план работы внешних устройств, для создания благоприятной среды внутри теплиц.

Для управление системой предусмотрена клиентская часть в виде android-приложения для конечного пользователя. В приложении реализован полный цикл использования от регистрации нового пользователя и регистрации им конечного устройства, до просмотра статистки и управления устройством и подключенных к нему дополнительных устройств. Несмотря на наличия автоматического

режима работы системы, пользователь вправе вручную изменять состояния устройств.

3. Особенности работы с API Яндекс.Погода

Для получения информации о погоде на сервер яндекса нужно сначала получить в личном кабинете разработчика получить уникальный ключ и отправить следующий GET-запрос, представленный в листинге 1, параметр с ключом для заголовка представлен в листинге 2. Параметры запроса к API Яндекс.Погода представлены в табл. 1.

Листинг 1

Формат запроса к API Яндекс.Погода

```
GET https://api.weather.yandex.ru/v2/forecast?
```

Таблица 1

Параметры запроса к API Яндекс.Погода

Поле	Описание	Тип
lat	Широта в градусах. Обязательное поле.	double
lon	Долгота в градусах. Обязательное поле.	double
lang	Сочетания языка и страны, для которых будут возвращены данные погодных формулировок.	String
limit	Количество дней в прогнозе, включая текущий.	integer
hours	Для каждого из дней ответ будет содержать прогноз погоды по часам.	boolean
extra	Расширенная информация об осадках.	boolean

Листинг 2

Строка с ключом в заголовок запроса

```
X-Yandex-API-Key: <значение ключа>
```

Этот запрос позволяет узнать фактическое значение и прогноз погоды для указанного населенного пункта.

Далее, на серверной стороне необходимо описать класс, который отлавливает и совмещает ответы от api на сущности проекта, с необходимыми полями, чтобы далее работать, как с обычными Java-объектами.

Ответ на запрос возвращается в формате JSON. Информация, которая содержится в ответе, представлена в табл. 2.

Таблица 2

Формат ответа API Яндекс.Погода

Поле	Описание	Формат
now	Время сервера в формате Unixtime.	Число
now_dt	Время сервера в UTC.	Строка
info	Объект информации о населенном пункте.	Объект
fact	Объект фактической информации о погоде.	Объект
forecasts	Объект прогнозной информации о погоде.	Объект

4. Конструкционные особенности устройства

Аппаратная часть информационной системы состоит из следующих частей:

- Печатной платы, на который размещен микроконтроллер и необходимая для него обвязка из электронных компонентов, набор реле, для управления нагрузкой и их обвязка, а так же компоненты для реализации питания платы и необходимые разъемы для подключения периферийных устройств (датчиков и модулей).
- Датчика температуры, давления, влажности и освещенности.
- Блока питания.

Обвязка микроконтроллера подразумевает использование wi-fi модуля, который необходим для передачи данных на сервер. Для этого, соответственно, устройству необходимо предоставить доступ к сети интернет.

Сигнал с датчиков приходит в микроконтроллер, который записывает информацию о показаниях и отправляет её на сервер.

Заключение

Разработана информационная система, которая обеспечивает сбор, обработку, передачу данных для управления климатом теплиц в тепличном комплексе или участками, которые требуют ухода и поддержания благоприятной среды почвы. Первый этап развития системы представлен на GitLab. [7] Перспективы дальнейшей разработки связаны с созданием системы интегрированной в тепличный комплекс, повышением отказоустойчивости всей системы, повышением качества системы для конечного пользователя.

Список литературы

1. Межгосударственный стандарт ГОСТ 14254-2015 (МЭК 60529-2013) «Степени защиты, обеспечиваемые оболочками (код IP)» [Текст] : ГОСТ 14254-2015 от 10 декабря 2015г. № 48-2015. – Взамен ГОСТ 14254-96 (МЭК-529-89); введ. 01.03.2017г. – Москва: Стандартинформ, 2016. – 39 с.
2. Раджпут. Д. Spring. Все паттерны проектирования / Д. Раджпут. – Санкт-Петербург : Питер, 2019. – 320 с.
3. Карнелл, Д., Санчес Иллари. У. Микросервисы Spring в действии / Д. Карнелл, У. Санчес Иллари, пер. с англ. А. Н. Киселева. – Москва : ДМК Пресс, 2022. – 490 с.
4. Ahmed, I., Smith, G., Pirozzi E. PostgreSQL 10 High Performance / I. Ahmed, G. Smith, E. Pirozzi. – Birmingham : Packt Publishing, 2018. – 508 с.
5. Водолазов, А. Микроконтроллеры для систем автоматики. Учебное пособие / А. Водолазов. – Вологда : Инфра-Инженерия, 2022. – 168 с.
6. Петин, В. Новые возможности Arduino, ESP, Raspberry Pi в проектах IoT / В. Петин. – Санкт-Петербург : БХВ-Петербург, 2022. – 319 с.
7. Программный код серверной и клиентской части информационной системы [Электронный ресурс] : GitLab-проект – Режим доступа: https://gitlab.com/khnykin-greenhouse