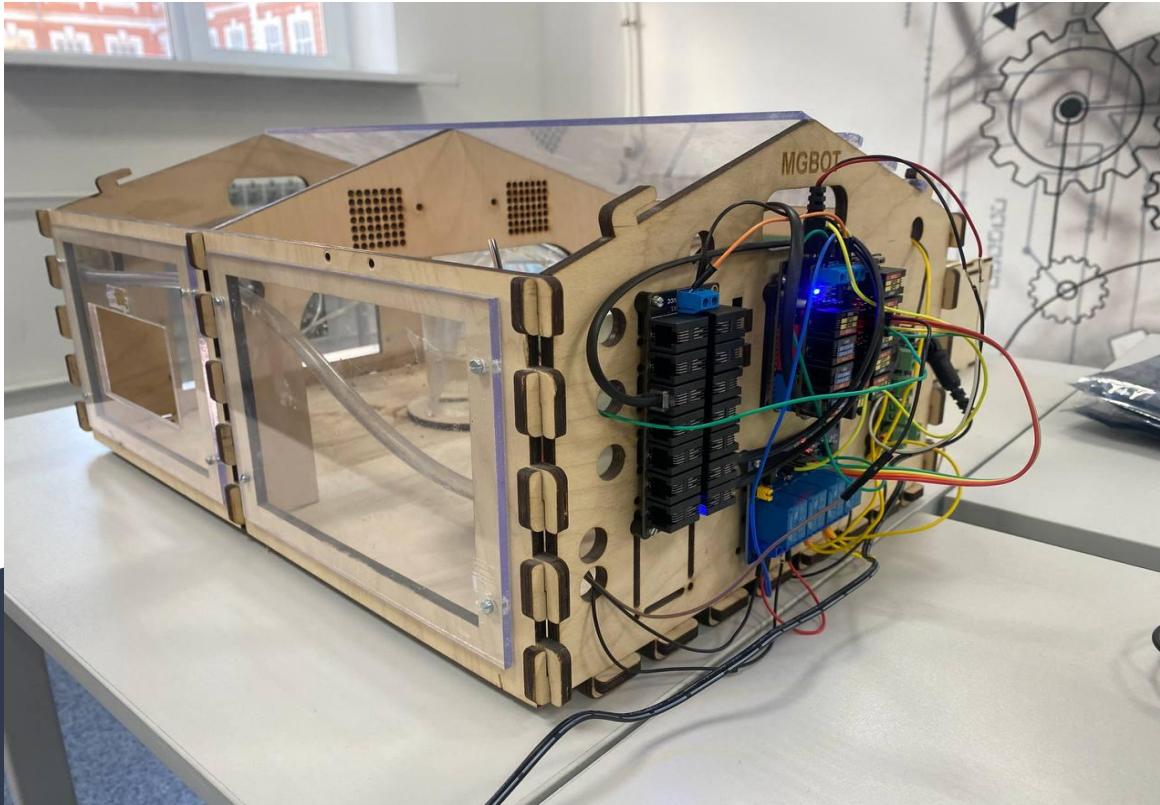


# Создание умной теплицы на базе ESP32



# Существующая проблема:

**Каждый год в России:**

**20-30%**

воды используемой для полива, тратится не эффективно

**до 30 %**

урожая теряется при не эффективном поливе

**10-20%**

от себестоимости продукции занимает зарплата рабочего персонала.

**до 20%**

урожая может теряться из-за человеческого фактора

**Проблема:**

- Экономия водных ресурсов;
- Повышение урожайности;

- Снижение себестоимости продукции за счет автоматизации производства;

**Зaintересованные отрасли:**



# Описание кейса:

**Задание:** Разработать техническое решение, для СитиФермы, на базе оборудования компании MGBOT.

## Пожелания к разработке:

- Возможность поддержания оптимальных условий выращивания растения;
- Мониторинг не менее трех параметров среды;
- Наличие архива измеряемых параметров;
- Возможность управления исполнительными механизмами;
- Унификация устройства под выращивание различных растений.

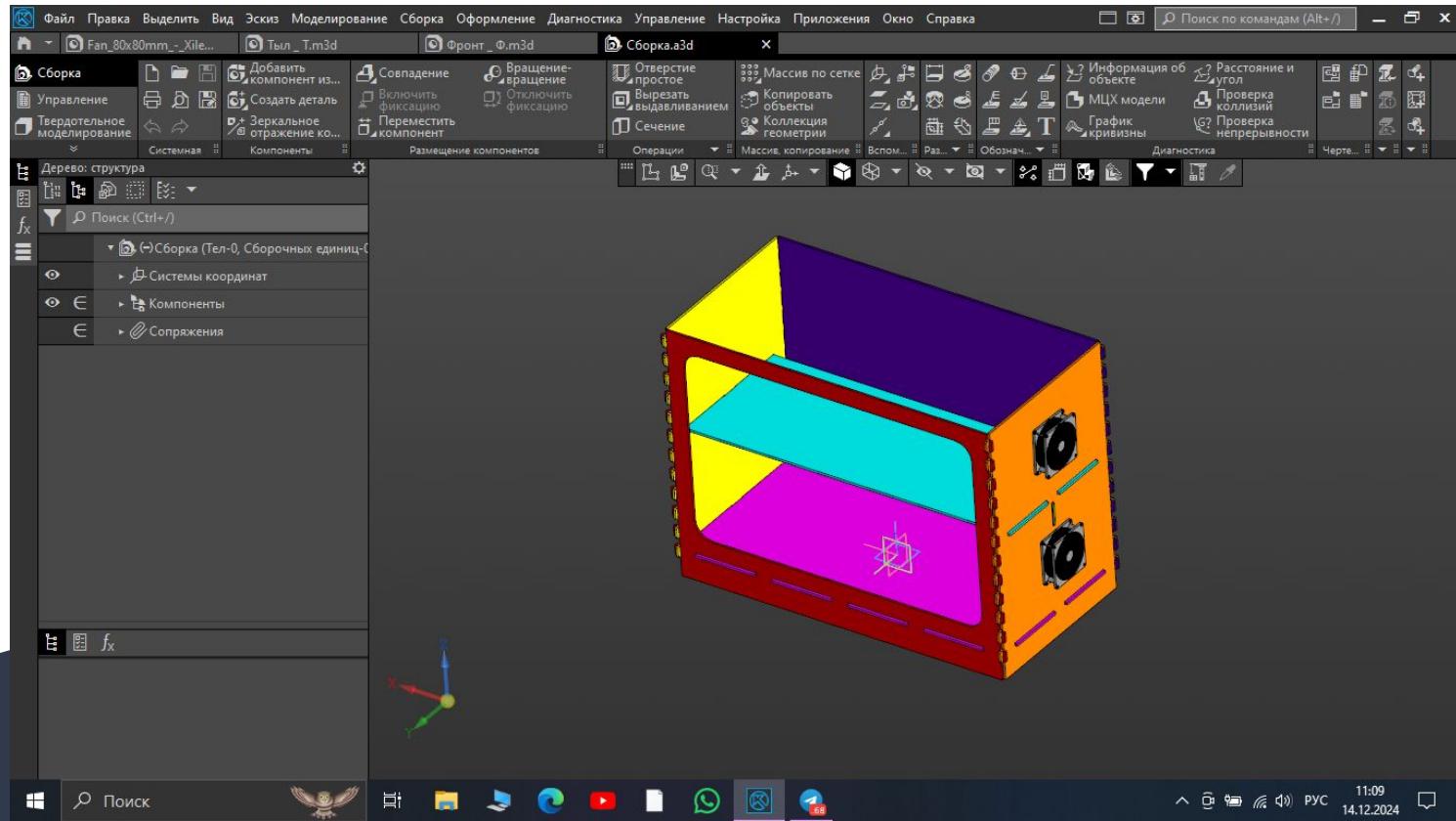
# Используемое оборудование:

- ESP32
- модуль 4 реле
- датчик темпла
- датчик света
- датчик влажности

# Команда проекта

- Анохин Никита: Java разработчик
- Бердников Даниил: Дизайнер
- Васильев Максим: Технический специалист

# 3D модель корпуса теплицы:



# Реализация Веб интерфейса программы:

- HTML
- CSS
- JAVASCRIPT
- THYMELEAF

# Безопасность. Страница авторизации администратора

**Авторизация**

Login

Password

ВХОД

**Авторизация**

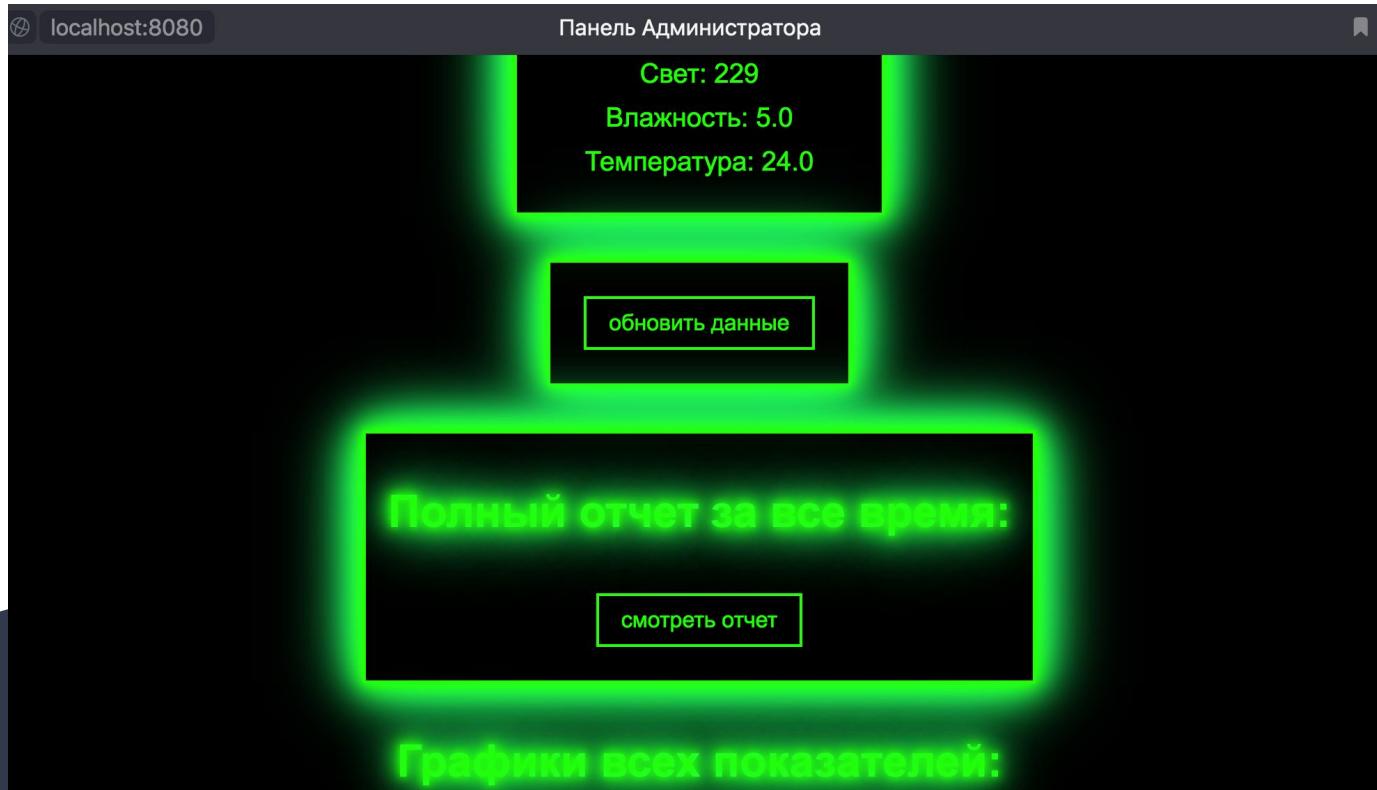
Login

Password

ВХОД

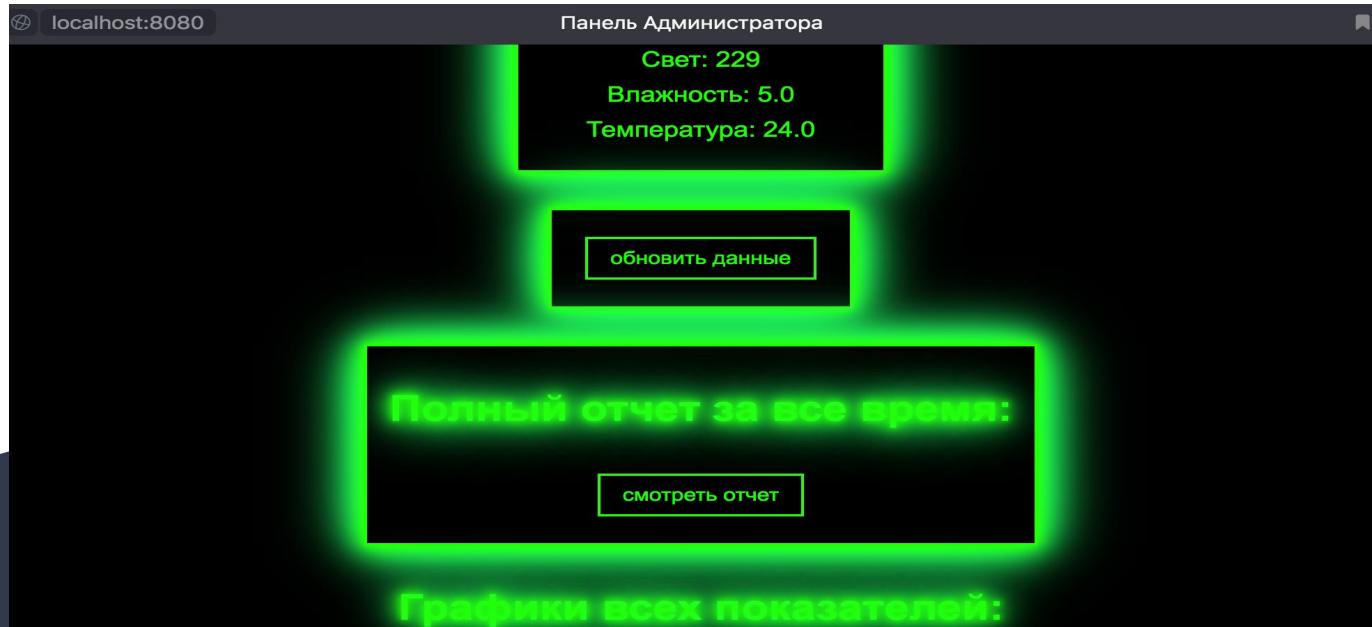
# Главная страница:



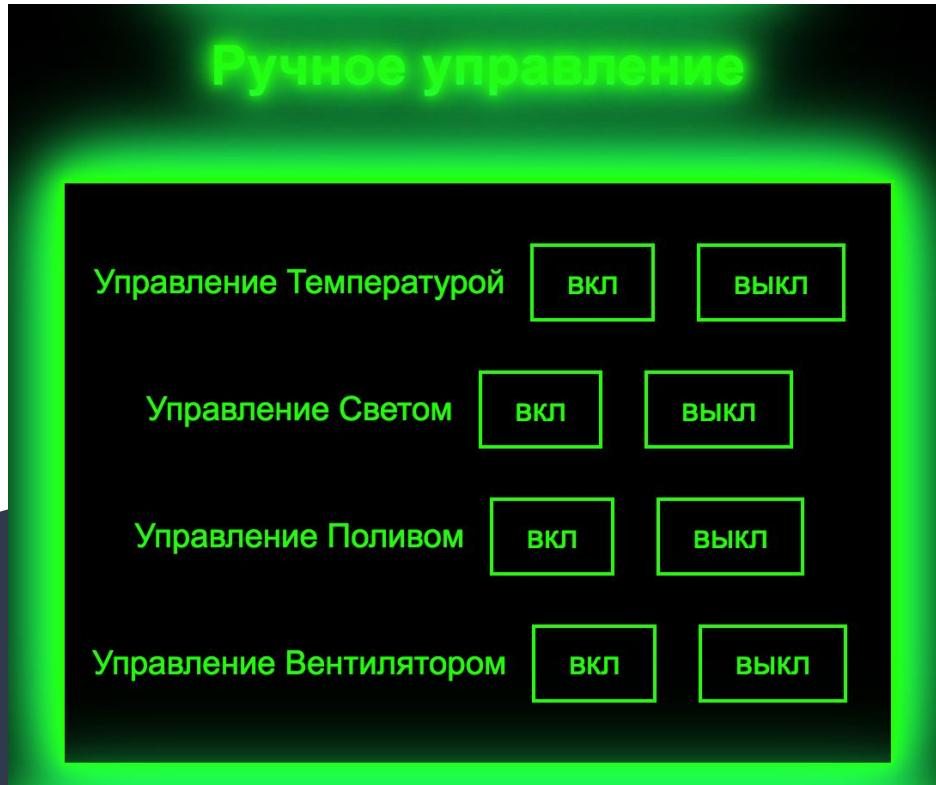
# Панель включает в себя:

- Обновление данных датчиков
- Ручное управление
- Полный отчет
- Запуск тг бота
- Вывод графиков данных
- Умное управление
- Авто статистика

# Вид панели. Данные датчиков. Обновление в реальном времени. Отчет



# Ручное управление в реальном времени:



# Управление графиками:

Графики всех показателей:

Освещение:

1 график

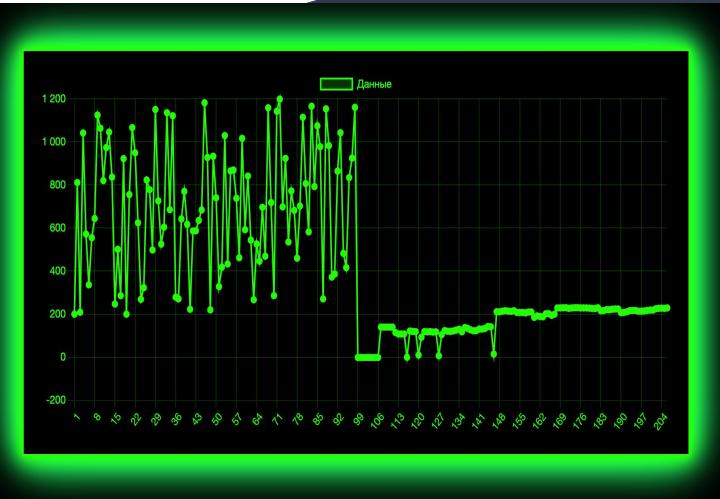
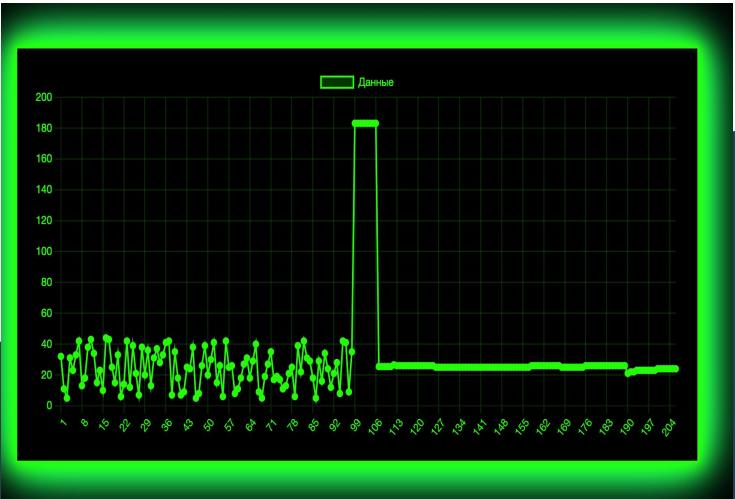
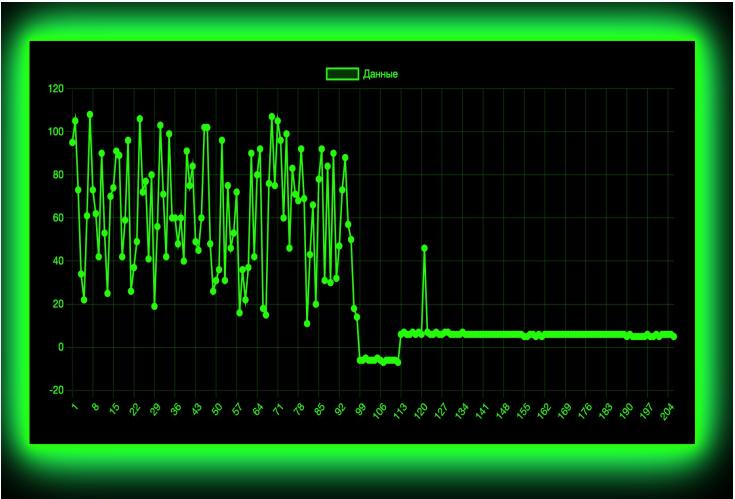
Влажность почвы:

2 график

Температура:

3 график

# Пример работы графиков:



# Настройка умного управления:

## Настройка умного управления

Оптимальный свет:  Время работы светильника:   
Оптимальная температура:  Время работы обогрева:   
Оптимальная влажность:  Время работы полива:

## Настройка умного управления

Оптимальный свет:  Время работы светильника:   
Оптимальная температура:  Время работы обогрева:   
Оптимальная влажность:  Время работы полива:

# Настройка автоматической статистики:

## Настройка умного управления

Интервал сбора информации:

3600

запуск умной статистики

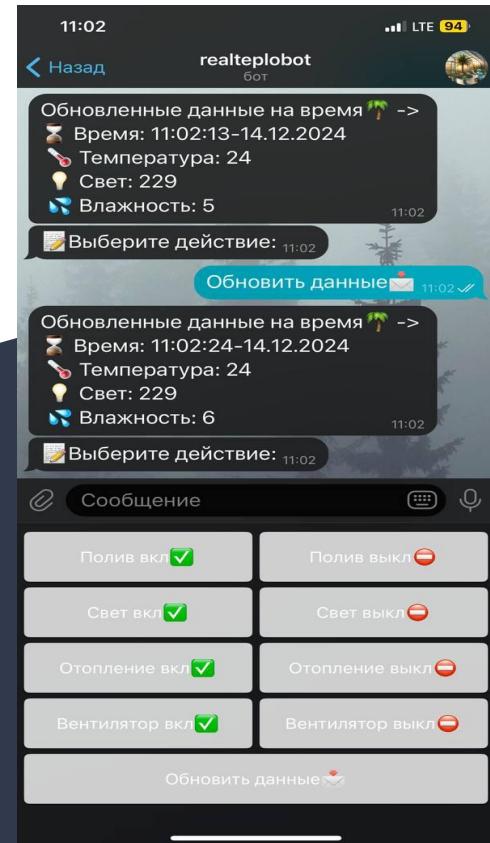
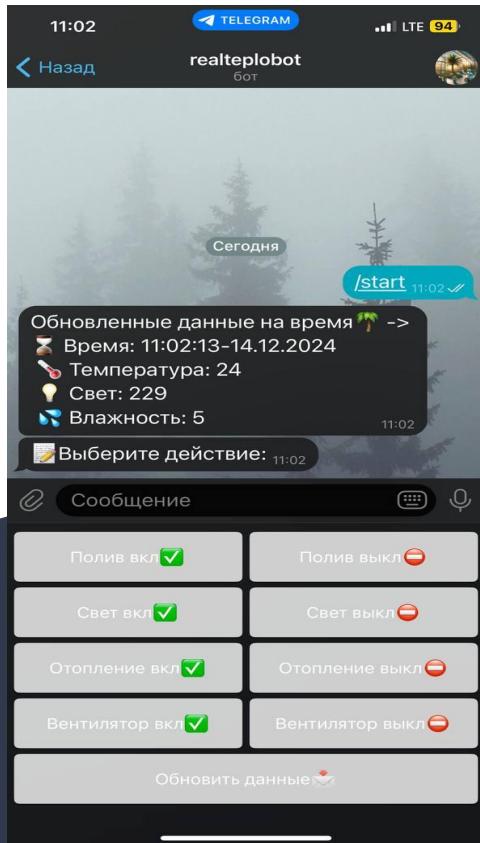
# Консольное логирование:

The screenshot shows a Java IDE interface with the following details:

- Project:** MeteoStarSpring
- Run:** MeteoStarSpringApplication
- Logs:** Console output showing application startup, Hibernate queries, and sensor data.
- IDE Features:** Project browser, DB browser, Favorites, Structure, Event Log, Terminal, Messages, and Build tabs.
- Build Status:** Compilation completed successfully in 3 s 499 ms (3 minutes ago).

```
2024-12-14 10:57:26.552 INFO 14704 --- [ restartedMain] org.hibernate.dialect.Dialect      : HHH000400: Using dialect: org.hibernate.dialect.PostgreSQLDialect
2024-12-14 10:57:27.467 INFO 14704 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator   : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformImpl]
2024-12-14 10:57:27.476 INFO 14704 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean: Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-12-14 10:57:27.936 WARN 14704 --- [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration: spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during a view's lifecycle. Add @QueryHints(QueryHint.DISABLE_OPEN_IN_VIEW_CHECK) or spring.jpa.open-in-view=false to the query to disable it.
2024-12-14 10:57:28.252 INFO 14704 --- [ restartedMain] o.s.b.a.w.s.WelcomePageHandlerMapping: Adding welcome page template: index
2024-12-14 10:57:28.505 INFO 14704 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer: LiveReload server is running on port 35729
2024-12-14 10:57:28.554 INFO 14704 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer: Tomcat started on ports(s): 8080 (http) with context path ''
2024-12-14 10:57:28.573 INFO 14704 --- [ restartedMain] c.e.M.MeteoStarSpringApplication: Started MeteoStarSpringApplication in 6.326 seconds (JVM running for 6.326 seconds)
2024-12-14 10:57:33.524 INFO 14704 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]: Initializing Spring DispatcherServlet 'dispatcherServlet'
2024-12-14 10:57:33.524 INFO 14704 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet: Initializing Servlet 'dispatcherServlet'
2024-12-14 10:57:33.527 INFO 14704 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet: Completed initialization in 2 ms
Response Code: 200
JSON Response: {"light":229,"temperature":24,"humidity":5}
GET request worked
Hibernate: select nextval ('hibernate_sequence')
Hibernate: insert into post (data, svet, temp, vlashnost, id) values (?, ?, ?, ?, ?)
Hibernate: select post0_.id as id1_0_, post0_.data as data2_0_, post0_.svet as svet3_0_, post0_.temp as temp4_0_, post0_.vlashnost as vlashnos5_0_ from post post0_
Hibernate: select post0_.id as id1_0_, post0_.data as data2_0_, post0_.svet as svet3_0_, post0_.temp as temp4_0_, post0_.vlashnost as vlashnos5_0_ from post post0_
Hibernate: select post0_.id as id1_0_, post0_.data as data2_0_, post0_.svet as svet3_0_, post0_.temp as temp4_0_, post0_.vlashnost as vlashnos5_0_ from post post0_
Hibernate: select post0_.id as id1_0_, post0_.data as data2_0_, post0_.svet as svet3_0_, post0_.temp as temp4_0_, post0_.vlashnost as vlashnos5_0_ from post post0_
Hibernate: select post0_.id as id1_0_, post0_.data as data2_0_, post0_.svet as svet3_0_, post0_.temp as temp4_0_, post0_.vlashnost as vlashnos5_0_ from post post0_
Hibernate: select post0_.id as id1_0_, post0_.data as data2_0_, post0_.svet as svet3_0_, post0_.temp as temp4_0_, post0_.vlashnost as vlashnos5_0_ from post post0_
Hibernate: select post0_.id as id1_0_, post0_.data as data2_0_, post0_.svet as svet3_0_, post0_.temp as temp4_0_, post0_.vlashnost as vlashnos5_0_ from post post0_
Проверяем теллицу..
Response Code: 200
JSON Response: {"light":226,"temperature":24,"humidity":5}
GET request worked
Response Code: 200
JSON Response: {"light":226,"temperature":24,"humidity":5}
GET request worked
Hibernate: select nextval ('hibernate_sequence')
Включаем реле - 4 на 500000 секунд!
Включаем реле - 2 на 5000 секунд!
Hibernate: insert into post (data, svet, temp, vlashnost, id) values (?, ?, ?, ?, ?)
Response Code: 200
Relay 2 turned ON
Response Code: 200
Relay 4 turned ON
Response Code: 200
Relay 2 turned OFF
Выключаем реле - 2
```

# Интерфейс телеграм бота:



# Программная масштабируемость проекта:

- дополнительная логика
- дополнительные датчики
- изменяемость подключаемой аппаратуры

# Интеграция в VR и AR



# Вывод:

Разработка умной теплицы является перспективным направлением в сельском хозяйстве, позволяющим повысить эффективность производства, снизить затраты и улучшить качество продукции. Использование современных технологий IoT делает процесс управления теплицами более удобным, надежным и экономически выгодным.