Выполнение практических работ и КР по предмету Разработка АС реального времени

В течении семестра вам предстоит разработать автоматизированную систему для некоторого вида производства. Рассмотрим пример. Допустим вам необходимо по заданию разработать автоматизированную систему реального времени для производства кирпичей. Что из себя будет представлять такого эта система, скорее всего она в зависимости от поставленных целей реализовывать следующий функционал: *мониторинг производственных линий, сбор данных с датчиков IoT, управление процессами в реальном времени и предоставление интерфейсов сотрудникам и программам для анализа производительности и отслеживания работы оборудования.* Для того, чтобы ускорить процесс разработки вы можете использовать для реализации фреймворк Spring). Например, вы можете использовать следующие компоненты Spring Framework, в частности, Spring Boot, Spring WebSocket, Spring Kafka, Spring Data, Spring Security и другие модули, что позволит вам создать гибкое, масштабируемое и интерактивное решение. Рассмотрим порядок разработки АС.

Порядок разработки АС реального времени

1. **Инициация разработки, начинается с целей (ответ на вопрос ЗАЧЕМ?)**

Для начала вам нужно определились цель вашей АС - для чего система создается и затем описать требуемый функционал системы.

Например, для АС реального времени по производству кирпичей это может быть следующий функционал:

1. Автоматизировать процесс управления оборудованием: управление потоком сырья, производить контроль температуры обжига, влажности, формовочных параметров.

2. Производить мониторинг состояния оборудования и показаний датчиков в реальном времени: сбор данных с производственных линий (температура, давление, сырье).

3. Обеспечить быстродействие системы: то есть поддерживать минимальную задержку в обмене данными для оптимальной работы всех производственных процессовпроцессов.

4. Поддерживать в системе уведомления в реальном времени и работу с аналитикой: выявлять отклонения, отправлять предупреждения, рассчитывать производственные KPI.

5. Обеспечить информационную безопасность системы : ограничить доступ к системе для несанкционированных пользователей.

На этом шаге, вам нужно на основе целей выявить требования к вашей АС

2. **Архитектура системы**

Здесь нам нужно определить основные компоненты нашей будущей системы или ее архитектуру. Система будет состоять из следующих ключевых компонентов:

1. Датчики IoT и PLC (программируемые логические контроллеры) для мониторинга производственного процесса/процессов.

2. Обработчик событий в реальном времени (на основе Kafka/WebSocket) для сбора и обработки данных.

3. Серверное приложение (Backend), реализованное с использованием Spring Framework для управления процессом и аналитики.

4. Панель управления (Frontend) для визуализации данных и взаимодействия с пользователем/пользователями.

5. База/базы данных(SQL или NoSQL) для хранения временных меток, информации о процессах и аналитики.

Мы перечислили основные компоненты, которые вам необходимо поэтапно реализовать в своей работе. После определения архитектуры можно перейти к разработке АС.

3. Процесс разработки автоматизированной системы

Рассмотрим шаги по разработке, которые вам предстоит выполнить, чтобы реализовать требуемый функционал АС.

1) Шаг 1. Получение данных

Выбор данных

Вам нужны данные, которые вы будете обрабатывать, следовательно вам нужно определить какие данные вы будете собирать.

Сбор данных с датчиков в реальном времени осуществляется с оборудования, либо вам может помочь ресурс https://www.kaggle.com/datasets, там вы можете поискать датасеты с нужными вам данными, вы также можете сэмулировать нужные данные с помощью метода Монте Карло

Для того чтобы собрать данные с реального оборудования, вам понадобятся обеспечить взаимодействие с ним.

Для взаимодействия с машинами и датчиками используются протоколы, такие как MQTT, Modbus или OPC-UA. Эти данные будут поступать на промежуточный обработчик (например, Apache Kafka), а затем обрабатываться микросервисом Spring, в зависимости от выбранной вами архитектуры.

Интеграция с датчиками через Spring

Как вам может помочь Spring? Вы можете использовать модуль Spring Integration& Используйте Spring Integration или Spring Boot для обработки входящих данных.

Например, для MQTT:

```xml
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-integration</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.integration</groupId>
  <artifactId>spring-integration-mqtt</artifactId>
</dependency>
```

Настройка MQTT:

```java
@Configuration
public class MqttConfig {

  @Bean
```

```
    public MqttPahoMessageDrivenChannelAdapter mqttAdapter() {
        return new
MqttPahoMessageDrivenChannelAdapter("tcp://broker.hivemq.com:1883", "mqtt-client");
    }

    @ServiceActivator(inputChannel = "mqttInputChannel")
    public void handleMessage(String payload) {
        System.out.println("Получено сообщение от датчика: " + payload);
        // Логика обработки
    }
}
```

После того как вы определитесь с оборудованием, данными и интерфейсами для их получения, вам нужно каким-то образом их обрабатывать.

2) Шаг 2. Обработка данных в реальном времени

У вас теперь есть данные, и вы знаете откуда и как их получать. Теперь их нужно обрабатывать. Как вам может помочь Spring?

Для построения системы обработки данных в реальном времени можно использовать Spring Kafka Streams.

Apache Kafka обеспечит передачу сообщений между компонентами системы без потерь и с минимальными задержками. Каждое сообщение может содержать данные о состоянии оборудования, таких как влажность, температура или статус производственной линии.

Пример. Разработка микросервиса обработки производственных данных:

- В `pom.xml` добавьте Kafka:

```xml
<dependency>
    <groupId>org.springframework.kafka</groupId>
    <artifactId>spring-kafka</artifactId>
</dependency>
```

- Определите топики Kafka:

- `raw-data` – для необработанных данных с датчиков.
- `alerts` – для уведомлений о проблемах.
- `stats` – для метрик и аналитики.

- Пример обработки данных:

```java
@Configuration
public class KafkaProcessor {

    @Bean
    public KStream<String, SensorData> processStream(StreamsBuilder builder) {
        KStream<String, SensorData> input = builder.stream("raw-data");
```

```
        KStream<String, SensorData> filtered = input.filter(
            (key, value) -> value.getTemperature() > 100 // Условие, например, превышение температуры
        );

        filtered.to("alerts");
        return input;
    }
}
```

3) Шаг 3. Мониторинг состояния системы (WebSocket) в реальном времени

Теперь у вас есть данные, вы можете их обрабатывать, но вам нужно реализовать функцию мониторинга, причем *в реальном времени*. Как вам может помочь Spring ?

Для отображения текущего состояния производственной линии и уведомлений в интерфейсе панели управления вы можете использовать Spring WebSocket. Это позволит передавать обновления в режиме реального времени на фронтенд. Эта технология поваляет установить двухстороннюю связь клиента с сервером и передавать данные в дуплексном режиме и представляет собой протокол взаимодействия транспортного уровня (TCP). Она нам как раз инужна чтобы передавать в режиме реального времени, с помощью htpp вы этого не получите.

Пример Spring WebSocket

- *Настроить WebSocket:*

```java
@Configuration
@EnableWebSocket
public class WebSocketConfig implements WebSocketConfigurer {
    @Override
    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
        registry.addHandler(new MonitoringWebSocketHandler(), "/monitoring")
                .setAllowedOrigins("*");
    }
}
```

- *Обработчик WebSocket:*

```java
public class MonitoringWebSocketHandler extends TextWebSocketHandler {
    @Override
    public void handleTextMessage(WebSocketSession session, TextMessage message) throws Exception {
        String payload = message.getPayload();
        // Логика для отправки обновлений с датчиков
        session.sendMessage(new TextMessage("Current Status: " + payload));
    }
}
```

4) Шаг 4. Хранение данных

Производственные системы часто работают с большими объемами данных. Вы можете использовать *Time-Series базы данных* (например, InfluxDB) для временных рядов данных или *PostgreSQL/MySQL* для обработки метрик

Примеры обработки метрик:

- Для временных данных: `sensor_data` (температура, влажность, давление и т.д.).
- Для критических ошибок: `alerts`.

Как вам может помочь в этом Spring? Вы можете использовать модуль Spring Data JPA для работы с БД. (если не используете, но используете Java, то это интерфейсы JDBC см лекции и методичку по практическим)

Пример с помощью Spring Data JPA:

```java
@Entity
public class SensorData {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   private double temperature;
   private double humidity;
   private LocalDateTime timestamp;
}
@Repository
public interface SensorDataRepository extends JpaRepository<SensorData, Long> {}
```

5) Шаг 5. Безопасность и управление доступом

Для обеспечения требований по безопасности, что очень важно для производственной системы, вы можете использовать модуль Spring Security для авторизации и аутентификации пользователей системы.

Например ваша система может реализовывать следующие функции:

- Операторы могут управлять оборудованием.
- Технические специалисты получают доступ к данным мониторинга.
- Руководство может видеть статистику и выводы.

Пример настройки Spring Security:

```java
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
   @Override
   protected void configure(HttpSecurity http) throws Exception {
      http
         .authorizeRequests()
```

```
            .antMatchers("/monitoring").authenticated() // Только авторизованные
пользователи
            .anyRequest().permitAll()
        .and()
        .formLogin();
    }
}
```

Пример использования панелью управления

Данные, отправляемые через WebSocket/Kafka, могут быть визуализированы через реализацию графского интерфейса пользователя, это могут быть панели управления, дэшборды, например для отображения графиков данных из базы, уведомлений и обезличенной аналитики (напримерт для АС для производства кирпичей это может быть среднее время обжига кирпича, и так далее). Для реализации интерфейса используйте React.js или Angular.

Пример интеграции через API (REST) для исторических данных и WebSocket для реального времени:

```java
@RestController
@RequestMapping("/api")
public class MonitoringController {
    @Autowired
    private SensorDataRepository sensorDataRepository;

    @GetMapping("/sensor-data")
    public List<SensorData> getSensorData() {
        return sensorDataRepository.findAll();
    }
}
```

Заключение.

Итак, что же у нас получилось? Опишем итоговую функциональность вашей системы

- Организация сбора данных с датчиков с помощью MQTT/Kafka в реальном времени.
- Обработка данных в реальном времени (например, проверка максимальных значений, расчет среднего времени, уведомления).
- визуализация данных, полученных через WebSocket на панели пользовательского интерфейса.
- Анализ производительности для оптимизации выполняемых процессов.

- Автоматизированные действия системы на основе событий (например, сигнал об остановке оборудования при сбое).

Разработка такой системы требует тесного сотрудничества с инженерными специалистами, работающими на производственной линии.

Но это еще не все нам нужно провести тестирование.

4. Процесс тестирования

После тога как мы разработали АС , нам необходимо провести тестирование нашей системы.

Тестирование автоматизированной системы реального времени— это важный этап разработки, так как любая ошибка в системе может привести к значительным последствиям, особенно в производственных процессах (например производство тех же кирпичей). Система в реальном времени обычно состоит из согласованного взаимодействия множества компонентов, поэтому подход к тестированию должен быть системным и многоэтапным.

Опишем основные шаги, подходы и инструменты, которые можно использовать для тестирования разработанной системы (см часть 2).