# Техническое задание на разработку системы мониторинга данных с виртуальных датчиков

## 1. Введение

Создать комплексное приложение для мониторинга данных, поступающих с "виртуальных датчиков", используя архитектуру из 4 контейнеров Docker.

## 2. Функциональные требования

## 2.1. API на ASP.NET Core

Реализовать API с методами:  
- `POST /api/upload-xml` – принимает XML-файл, валидирует его по XSD-схеме и сохраняет в базе.  
- `POST /api/data` – принимает данные от эмулятора датчиков и сохраняет их в базе.  
- `GET /api/data` – возвращает данные за заданный период времени.  
- `GET /api/sensors/summary` – возвращает агрегированные данные (среднее, максимум, минимум) за указанный период.  
- Документировать API с использованием Swagger/OpenAPI.

## 2.2. Сервис эмуляции данных

- Реализовать сервис генерации данных с "виртуальных датчиков":  
 - Создать 3 датчика (ID: 1, 2, 3), которые каждую секунду отправляют случайное значение в диапазоне (0–100).  
 - Обеспечить отправку данных в базу через API.

## 2.3. База данных

- Создать БД для хранения данных, полученных с сервиса эмуляции.  
- Использовать PostgreSQL/MSSQL.

## 2.4. Фронтенд (React + TypeScript)

Реализовать веб-интерфейс со следующими функциями:  
- Отображение данных в виде таблицы с фильтром по диапазону времени.  
- Отображение данных на графике (временной график с линейной интерполяцией).  
- Автоматическое обновление данных каждые 5 секунд.  
- UI/UX оформление (CSS/Material-UI/Ant Design).  
- Кнопка для выгрузки данных из БД в XML-файл и отправка его в API для валидации.  
- Обработка XML согласно XSD-схеме и отображение статуса валидации на фронтенде.

## 3. Архитектура и развертывание

Проект должен состоять из следующих Docker-контейнеров:  
1. Сервис эмуляции данных  
2. База данных  
3. API  
4. Фронтенд

## 4. Дополнительные требования

- Код должен быть размещен в публичном репозитории (GitHub/GitLab).  
- Возможность локального запуска без Docker (инструкции в `README.md`).  
- Документация должна содержать:  
 - Описание архитектуры приложения.  
 - Инструкции по запуску (Docker и без Docker).  
 - Примеры работы API (curl/Postman).

## 5. Критерии оценки

1. \*\*Функциональность:\*\*  
 - Корректная работа API.  
 - Работоспособность фронтенда и обновление данных в реальном времени.  
2. \*\*Качество кода:\*\*  
 - Использование современных практик (TypeScript, React hooks, DI в ASP.NET).  
3. \*\*Инфраструктура:\*\*  
 - Четкая организация Docker-контейнеров.  
4. \*\*Документация:\*\*  
 - Полное и понятное описание проекта и его развертывания.