Инженер по разработке ПО для базовых станций (Python)

https://careers.yadro.com/impulse/#inzhener-po-razrabotke-po-dlya-bazovyh-stanczij-python

Оглавление

Общая информация

Базовая станция – сложный аппаратно-программная система по приему и передаче радиосигналов, способная работать со множеством мобильных устройств одновременно. Аппаратная часть состоит в том числе из антенны и одного или нескольких радиомодулей, связанных между собой физически. *Программная часть* состоит из большого количества сервисов, работающих вместе на одном «железе», при этом каждый из них отвечает за конкретный функционал. Координирующим узлом в программной части выступает *система управления базовой станцией*.

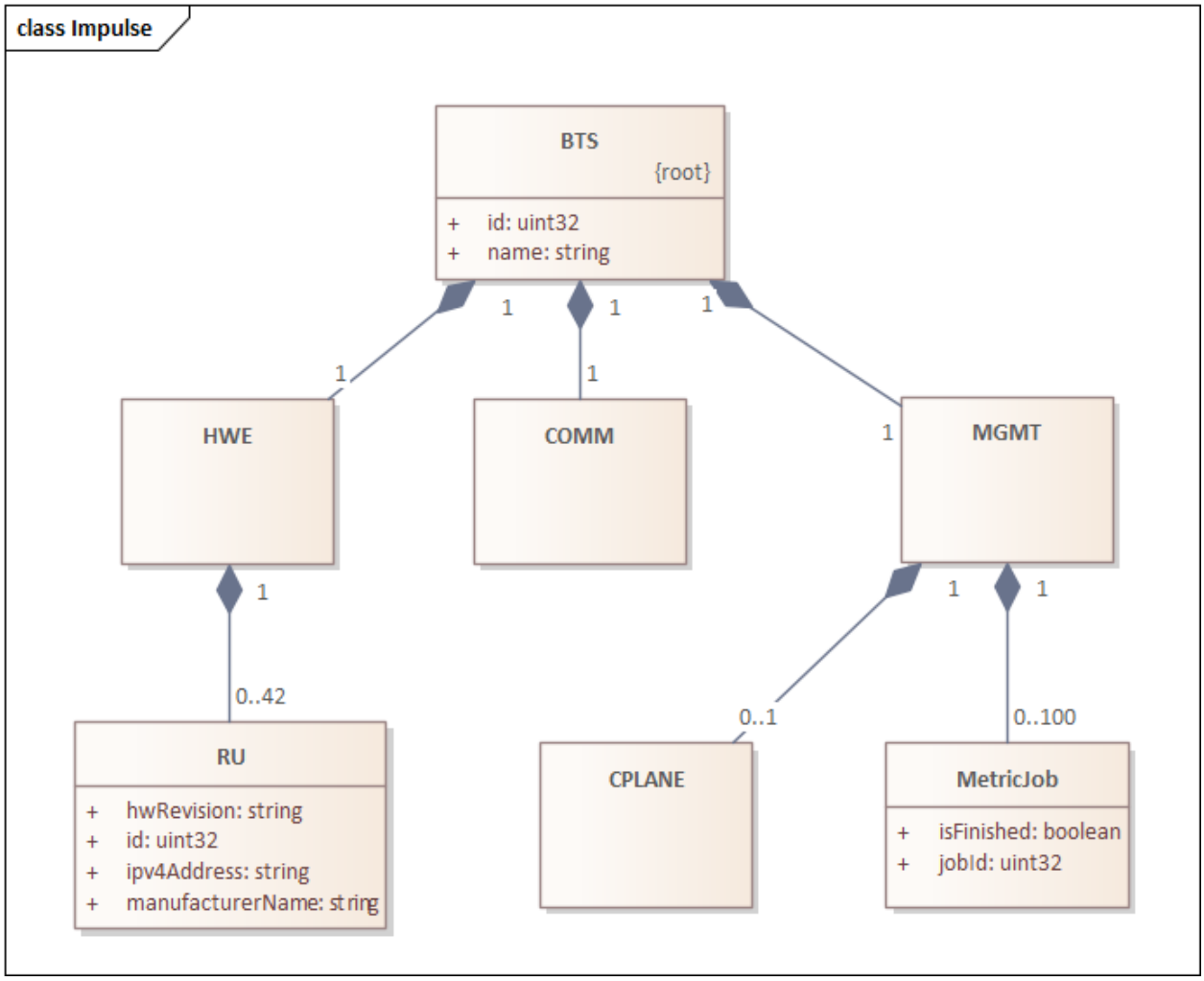
Важной частью системы управления базовой станции является *информационная модель*, которая представляет собой формальную UML-диаграмму с описанием иерархической структуры *классов и их атрибутов*. Данная модель разрабатывается в специализированных программах, которые предоставляют удобный интерфейс для одновременной совместной работы над моделями, хранение их в БД, разграничение доступа. В процессе эволюции системы в модель регулярно вносятся изменения (появляются новые классы, изменяются атрибуты и т.д.).

Модель в виде рисунка понятна и привычна для человека, но для компьютерной обработки удобнее другой формат, более приближенный к машинному, поэтому модель может быть переведена в *XML-файл*, однозначно соответствующий UML-диаграмме.

Полученный XML проверяется на непротиворечивость и отсутствие ошибок (при разработке всегда может сыграть человеческий фактор) – этот процесс мы называем *валидацией*. Если ошибок нет, запускается *генерация* определенных частей исходного кода – *артефактов* – по сути, интерфейсов для программ всех компонент базовой станции (к ним относятся файлы конфигурации, YANG-файлы, исходники на C++ и Go, Proto-файлы и т.д.). Одновременный переход всех компонент на новую версию артефактов позволяет избежать ошибок несовместимости версий и является одной из первоочередных задач системы управления базовой станцией.

Информационная модель

Для тестового задания предлагается некоторая упрощенная модель, из которой убраны лишние связи и атрибуты, но тем не менее она использует те же принципы, что и основная ее версия:



XML-файл модели состоит из двух логических частей – описание классов и описание связей между ними. Каждый класс содержит описание атрибутов (если они есть), каждый атрибут имеет тип. Связи между классами – агрегации. Для каждой из них указано два имени – target (класс, включающий в себя другой класс) и source (класс, включенный в другой класс). Также в отношениях агрегации указаны мощности (multiplicity) – может быть как одно число, так и диапазон [min..max].

Файл: **impulse_test_input.xml**

Задание

Требуется написать программу на Python, которая на основе входного XML-файла (**impulse_test_input.xml**) генерирует следующие виды выходных файлов (артефактов):

1. Файл **config.xml** представляет собой пример внутренней конфигурации базовой станции, соответствующий представленной модели.
2. Файл **meta.json** содержит мета-информацию о классах и их атрибутах. Фронтенд использует данный файл для корректного отображения дерева объектов в интерфейсе пользователя (UI).

Ожидаемую структуру выходных файлов можно посмотреть в приложенных файлах-примерах.

Требования к решению

1. Ожидаем присланный по почте архив названный (в латинице) **<ВашеИмя>_<Ваша_фамилия>.zip** с исходным кодом и папкой **out**, в которой лежат 2 сгенерированных вашей программой выходных файла – **config.xml** и **meta.json**.
2. Исполнение программы должно осуществляться запуском файла **main.py** без параметров из корня архива. Сгенерированные файлы при этом должны сохраняться в папке **out**.
3. Скрипты корректно работают на Python версии **3.11**
4. Можно использовать библиотеки, входящие в стандартный комплект поставки Python

На что еще стоит обратить внимание

При оценке решений будут учитываться следующие моменты:

1. Структурированность и читаемость кода
2. Эффективность используемых алгоритмов
3. Объектно-ориентированный подход
4. Расширяемость и масштабируемость представленного решения
5. Разумное использование шаблонов (паттернов) проектирования

Удачи!

Приложение 1. Входной файл impulse_test_input.xml

```
<?xml version="1.0" ?>
<XMI xmi.version="1.1" xmlns:UML="omg.org/UML1.3" timestamp="2024-05-11
12:34:56">
    <Class name="BTS" isRoot="true" documentation="Base Transmitter Station. This
is the only root class">
        <Attribute name="id" type="uint32" />
        <Attribute name="name" type="string" />
    </Class>
    <Class name="MGMT" isRoot="false" documentation="Management related">
    </Class>
    <Class name="COMM" isRoot="false" documentation="Communication services">
    </Class>
    <Class name="MetricJob" isRoot="false" documentation="Perfomance metric job">
        <Attribute name="isFinished" type="boolean" />
        <Attribute name="jobId" type="uint32" />
    </Class>
    <Class name="CPLANE" isRoot="false" documentation="Perfomance metric job">
    </Class>
    <Class name="RU" isRoot="false" documentation="Radio Unit hardware element">
        <Attribute name="hwRevision" type="string" />
        <Attribute name="id" type="uint32" />
        <Attribute name="ipv4Address" type="string" />
        <Attribute name="manufacturerName" type="string" />
    </Class>
    <Class name="HWE" isRoot="false" documentation="Hardware equipment">
    </Class>
    <Aggregation source="MGMT" target="BTS" sourceMultiplicity="1"
targetMultiplicity="1" />
    <Aggregation source="HWE" target="BTS" sourceMultiplicity="1"
targetMultiplicity="1" />
    <Aggregation source="COMM" target="BTS" sourceMultiplicity="1"
targetMultiplicity="1" />
    <Aggregation source="MetricJob" target="MGMT" sourceMultiplicity="0..100"
targetMultiplicity="1" />
    <Aggregation source="CPLANE" target="MGMT" sourceMultiplicity="0..1"
targetMultiplicity="1" />
    <Aggregation source="RU" target="HWE" sourceMultiplicity="0..42"
targetMultiplicity="1" />
</XMI>
```

Приложение 2. Пример выходного файла config.xml

```
<BTS>
    <id>uint32</id>
    <name>string</name>
    <MGMT>
        <MetricJob>
            <isFinished>boolean</isFinished>
            <jobId>uint32</jobId>
        </MetricJob>
        <CPLANE>
        </CPLANE>
    </MGMT>
    <HWE>
        <RU>
            <hwRevision>string</hwRevision>
            <id>uint32</id>
            <ipv4Address>string</ipv4Address>
            <manufacturerName>string</manufacturerName>
        </RU>
    </HWE>
    <COMM>
    </COMM>
</BTS>
```

Приложение 3. Пример выходного файла meta.json


```
[
    {
        "class": "MetricJob",
        "documentation": "Perfomance metric job",
        "isRoot": false,
        "max": "100",
        "min": "0",
        "parameters": [
            {
                "name": "isFinished",
                "type": "boolean"
            },
            {
                "name": "jobId",
                "type": "uint32"
            }
        ]
    },
    {
        "class": "CPLANE",
        "documentation": "Perfomance metric job",
        "isRoot": false,
        "max": "1",
        "min": "0",
        "parameters": []
    },
    {
        "class": "MGMT",
        "documentation": "Management related",
        "isRoot": false,
        "max": "1",
        "min": "1",
        "parameters": [
            {
                "name": "MetricJob",
                "type": "class"
            },
            {
                "name": "CPLANE",
                "type": "class"
            }
        ]
    },
    {
        "class": "RU",
        "documentation": "Radio Unit hardware element",
        "isRoot": false,
        "max": "42",
        "min": "0",
```

```
        "parameters": [
            {
                "name": "hwRevision",
                "type": "string"
            },
            {
                "name": "id",
                "type": "uint32"
            },
            {
                "name": "ipv4Address",
                "type": "string"
            },
            {
                "name": "manufacturerName",
                "type": "string"
            }
        ]
    },
    {
        "class": "HWE",
        "documentation": "Hardware equipment",
        "isRoot": false,
        "max": "1",
        "min": "1",
        "parameters": [
            {
                "name": "RU",
                "type": "class"
            }
        ]
    },
    {
        "class": "COMM",
        "documentation": "Communication services",
        "isRoot": false,
        "max": "1",
        "min": "1",
        "parameters": []
    },
    {
        "class": "BTS",
        "documentation": "Base Transmitter Station. This is the only root class",
        "isRoot": true,
        "parameters": [
            {
                "name": "id",
                "type": "uint32"
            },
            {
                "name": "name",
```

```
                "type": "string"
            },
            {
                "name": "MGMT",
                "type": "class"
            },
            {
                "name": "HWE",
                "type": "class"
            },
            {
                "name": "COMM",
                "type": "class"
            }
        ]
    }
]
```