Факультет информационных технологий
Кафедра параллельных вычислений

Выпускная квалификационная работа бакалавра

Разработка системы управления базой фрагментов кода для системы активных знаний

Направление подготовки 09.03.01 Информатика и вычислительная техника Направленность (профиль): Программная инженерия и компьютерные науки

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Соруководитель ВКР
ст. преп. каф. ПВ ФИТ
Городничев М. А

20.06.2025

Руководитель ВКР
к.т.н, доц. каф. ПВ ФИТ
Перепёлкин В. А.

Введение

Разработка прикладных параллельных программ на высокопроизводительных вычислительных системах — сложный и трудоёмкий процесс.

Однако со временем накапливается множество уже созданного программного обеспечения, и усилия по разработке постепенно заменяются усилиями по поиску и применению готовых решений. (Поиск подходящих библиотек, определение, какая из них предпочтительнее).

Чтобы использовать уже разработанную программу в новом контексте, требуется программист, – он должен разобраться, как работает программа.

Даже программист, хорошо разбирающийся в своей области, вынужден тратить усилия на изучение особенностей программ в конкретной предметной области, и при переходе от одной области к другой этот процесс приходится проходить заново.

Система активных знаний

Система активных знаний представляет собой программное решение для автоматизации процесса “стыковки” различных программ, а также для автоматического создания параллельных программ специалистом в предметной области, не обладающим навыками программирования.

С помощью этой системы человек, не умеющий писать код, может создать параллельную программу, которая выполняет то, что он описал с помощью интуитивно понятного языка описания графов.

Автоматизация создания параллельных программ позволяет сократить время разработки. Более того, при автоматизации исключается влияние человеческого фактора, что дополнительно снижает ресурсоёмкость процесса.

Проблема и актуальность

Проблема – разработка параллельных программ требует много ресурсов и, из-за влияния человеческого фактора, чем масштабнее система, тем больше ресурсов требуется.

Данная проблема является актуальной, потому, что со временем – количество написанных программ растет, что делает их поиск и переиспользование ресурсозатратнее, в то время как размер разрабатываемых систем увеличивается, что увеличивает влияние человеческого фактора на разработку.

Автоматизация процесса переиспользования написанных программ – снизит количество необходимых ресурсов необходимых для разработки программ.

Цель и задачи

Система управления базой фрагментов кода (СУБФК) – программа, разрабатываемая в составе прототипа системы активных знаний, отвечающая за хранение, а также стандартизированное переиспользование сохранённых в системе программ.

Цель работы — разработка системы управления базой фрагментов кода в составе системы LuNA.

Задачи:

1. Конкретизировать потребности системы активных знаний и требования предъявляемые к СУБФК
2. Проанализировать существующие решения, и, по возможности, адаптировать их в систему активных знаний
3. Разработать технические решения для создания СУБФК и конкретизировать разработанное приложение

Обзор существующих средств хранения и стандартизации фрагментов кода

Основные параметры оценки:

1. Возможность автоматизированного переиспользования фрагментов кода в широком смысле без участия человека
2. Возможность добавления множества метаданных к фрагментам кода
3. Расширяемость. Возможность добавления поддержки новых типов фрагментов кода без переработки ядра системы

Основные родственные работы:

1. SPIRAL
2. GitHub Copilot
3. Protégé
4. KNIME
5. Galaxy

Таблица 1.1 – Сравнение систем по потребностям системы активных знаний

Система	Авт. переиспользование	Метаданные	Общий случай
SPIRAL	-	-	-
GitHub Copilot	+- (полуавтоматическое)	-	+
Protégé	-	-	-
KNIME	+	-	+-
Galaxy	+	-	+-

Вывод

Существующих решений недостаточно для удовлетворения потребностей системы активных знаний.

На данный момент нет систем, которые бы сочетали в себе автоматическое переиспользование фрагментов кода с учётом нефункциональных свойств конкретных фрагментов в общем случае.

Исследование этой темы — целесообразно.

Базовый принцип работы системы активных знаний

Ключевая идея, которая позволяет автоматически создавать параллельные (и не только) программы, состоит в выводе решения задачи, поставленной на вычислительной модели, и в последующей подстановке соответствующих модулей (фрагментов кода).

Вычислительная модель представляет собой ориентированный двудольный конечный граф, состоящий из двух типов вершин (двух долей): операций и переменных. Дуги, входящие в операционный узел, задают входные переменные, а исходящие из него — выходные переменные.

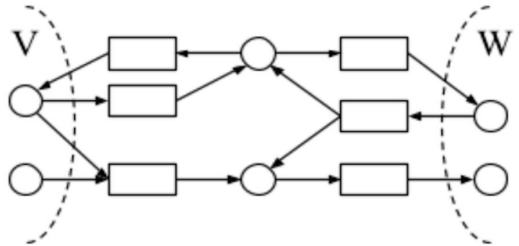


Рисунок 2.1 – Пример вычислительной модели. Круги — переменные, прямоугольники — операции

Структура графа отражает вычислительные связи между данными и преобразованиями и однозначно задает из каких переменных можно вывести другие, и какие другие переменные для этого понадобятся.

V-W задача

V-W задача - способ задания функциональных требований к решаемой задаче.

V - входные параметры задачи, W - целевые параметры задачи. При фиксации значений всех переменных из V формируется VW-задача.

Однако, для практического применения такого подхода нужно также учитывать нефункциональные свойства конкретных подставляемых операций (К примеру, одна операция может быть быстрее, но тратить больше памяти во время выполнения, в то время как другая — наоборот)

Постановка задачи

В описанном выше контексте СУБФК — это компонент, из которого достаются конкретные операции и нефункциональные свойства, подставляемые в модули перед вычислениями.

Несмотря на наличие большого количества существующих программных решений (библиотек, фрагментов кода), их повторное использование сопряжено с рядом проблем:

• Необходимость ручного анализа и адаптации существующего кода

• Отсутствие стандартизированных механизмов интеграции компонентов

• Высокий порог входа для специалистов предметной области, не обладающих квалификациями в программировании

Для решения этих проблем требуется разработать **систему управления базой фрагментов кода (СУБФК)**, которая должна обеспечить часть функциональности системы активных знаний связанной с хранением добавленных в систему фрагментов кода и стандартизацией фрагментов кода для будущего эффективного переиспользования.

Предлагаемое решение

Предлагается решить эту проблему системой плагинов.

Общая идея такая: так как в общем случае переиспользовать фрагменты кода возможности нет — уменьшить множество фрагментов кода до такого размера, когда стандартизированное переиспользование — возможно. (Множество фрагментов кода превращается в конкретный тип)

Для этого типа фрагментов кода можно написать “плагин”— встраиваемый в СУБФК код, который добавляет возможность СУБФК работать с этим множеством фрагментов кода стандартизированно.

Компонент системы активных знаний №1

Компонент системы активных знаний №2

Компонент системы активных знаний №3

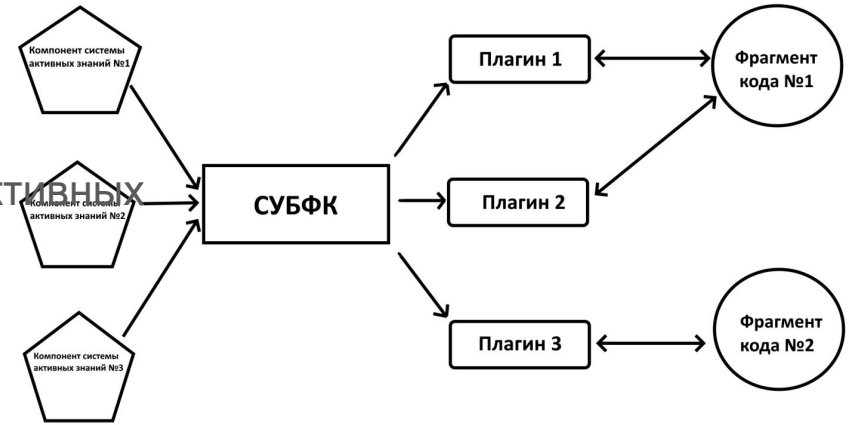


Рисунок 2.3 – Схема работы СУБФК

Стандартизация фрагментов кода

Для того чтобы автоматически переиспользовать фрагмент кода, он должен соответствовать определенному соглашению. В предлагаемом решении это достигается путем приложения к фрагменту кода некоторых метаданных в заранее определенном формате.

Эти метаданные делятся на обязательные и необязательные. Обязательные метаданные определены для каждого плагина и могут различаться. Необязательные метаданные зависят от предметной области и могут использоваться для выводов более эффективных VW-планов.

При добавлении фрагмента кода в систему нужно добавить к нему все необходимые обязательные метаданные для каждого плагина, с которым будет работать этот фрагмент кода.

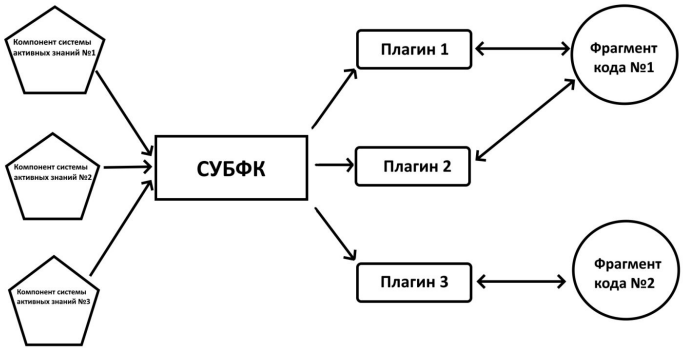


Рисунок 2.3 – Схема работы СУБФК

Модель фрагмента кода

Модель фрагмента кода состоит из двух частей — корневых файлов программы и метаданных, сохраненных придерживаясь известного соглашения для подобных метаданных.

Сам формат корневых файлов программы и метаданных не играет решающей роли, однако у корневых файлов программы должна быть возможность быть переиспользованными компетентным программистом.

В свою очередь у СУБФК должна быть возможность однозначно определить и прочитать метаданные приложенные к корневым файлам программы, а в самих метаданных должно быть:

1. Количество и типы входных и выходных аргументов программы
2. Количество и идентификаторы плагинов, с которыми может взаимодействовать этот фрагмент кода.

Анализ решения

1. Предложенный вариант СУБФК предполагает под собой автоматическое переиспользование фрагментов кода без вмешательства человека.

2. СУБФК напрямую не работает с фрагментами кода, но работает с сущностью “плагин”, поэтому информации про внутреннюю реализацию фрагмента кода у неё — нет.

3. Расширяемость системы напрямую заложена в предложенный вариант СУБФК.

4. В предлагаемом решении оставлено место для добавления дополнительных необязательных метаданных, количество которых ограничено только с технической точки зрения. Также из-за стандартизованного хранения метаданных у фрагментов кода в СУБФК могут быть реализованы механизмы валидации этих метаданных.

Недостатки предлагаемого решения

1. **Необходимость разработки плагинов**

   Рядовой пользователь, как использующий саму систему активных знаний, так и пользователь, добавляющий фрагменты кода в систему, не имеет возможности добавить новый тип фрагментов кода в систему, так как это требует продвинутых навыков программирования.

2. **Ответственность за заполнение метданных ложится на пользователя**

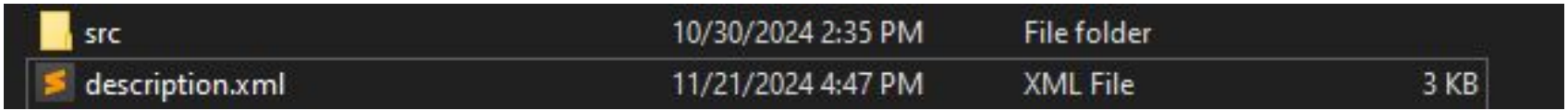
   В то время как СУБФК имеет возможность проверить наличие обязательных метаданных, возможности гарантировать правильность этих данных — нет. Это следует из того, что у СУБФК нет информации о внутренней структуре фрагмента кода. Этот факт позволяет стандартизированно переиспользовать добавленные фрагменты кода, но ограничивает систему в возможностях валидации приложенных метаданных. Если данные введены некорректно, то это может выясниться только после создания итоговой сгенерированной программы, а может и не выясниться вовсе.

Разработанный прототип СУБФК

Для реализации решений, предложенных ранее, был разработан прототип СУБФК. Система включает в себя три части — **хранилище фрагментов кода**, **хранилище плагинов** и **систему управления**. Описанное решение реализовано в виде микросервиса на языке Java с помощью фреймворка “Spring Framework”.

- **Хранилище фрагментов кода**

Хранилище фрагментов кода представляет собой определенную папку в системе, где была запущена система управления. В хранилище фрагментов кода содержатся пакеты фрагментов кода добавленные в систему. Пакет фрагментов кода — единица добавления фрагментов кода в систему (может содержать в себе несколько фрагментов кода). Пакет представляет собой папку, содержащую в себе файл “description.xml” и папку “src”. В папке “src” хранятся все файлы внутренней реализации фрагментов кода. В файле “description.xml” хранятся метаданные фрагментов кода пакета.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<code_fragments>
  <cf>
    <name>Корреляционная свертка сейсмотрасс</name>
    <description>Корреляционная свертка сейсмотрасс. В отчете для полученных коррелотрасс приводится отношение сигнал/шум
        (SNR) и эквивалентная амплитуда сигнала (A) в дискретах АЦП.Отчет о работе выводится на экран и в файл corr.log В
        отчете для полученных коррелотрасс приводится отношение сигнал/шум (SNR) и эквивалентная амплитуда сигнала (A) в
        дискретах АЦП.</description>
    <input_var_count>3</input_var_count>
    <ivars>
      <ivar>Текстовый файл со списком сейсмотрасс в формате PC-A</ivar>
      <ivar>Файл опорного сигнала в формате PC-A</ivar>
      <ivar>Интервал корреляции в секундах</ivar>
    </ivars>
    <output_var_count>2</output_var_count>
    <ovars>
      <ovar>Отчет о работе в формате .log</ovar>
      <ovar>Итоговый файл свертки сейсмотрассы</ovar>
    </ovars>
    <ivar_types>
      <ivar_type>any_path</ivar_type>
      <ivar_type>any_path</ivar_type>
      <ivar_type>int</ivar_type>
    </ivar_types>
    <ovar_types>
      <ovar_type>local_path</ovar_type>
      <ovar_type>local_path</ovar_type>
    </ovar_types>
    <local_paths>
      <local_path>corr.log</local_path>
      <local_path>_corr/out</local_path>
    </local_paths>
    <implemented_plugins>
      <plugin>ExeStartCProcedure</plugin>
    </implemented_plugins>
    <plugin_specific>
      <plugin>
        <name>ExeStartCProcedure</name>
        <exe_path>corr.out</exe_path>
      </plugin>
    </plugin_specific>
    <id>b642e476-8d79-4783-b1da-a016b0d226aa</id>
  </cf>
</code_fragments>
```

```
<implemented_plugins>
  <plugin>ExeStartCProcedure</plugin>
</implemented_plugins>
<plugin_specific>
  <plugin>
    <name>ExeStartCProcedure</name>
    <exe_path>corr.out</exe_path>
  </plugin>
</plugin_specific>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<code_fragments>
  <cf>
```

```
<id>b642e476-8d79-4783-b1da-a016b0d226aa</id>
```

Хранилище плагинов

Хранилище плагинов представляет собой определенную папку в системе, где была запущена система управления. Внутри папки хранилища плагинов хранятся плагины СУБФК в виде папок. Плагин — программа-дополнение для системы управления. Эта программа описывает как система управления должна обрабатывать определенный тип фрагментов кода. Плагин является java классом реализующим интерфейс Plugin.

```
1  public interface Plugin {
2    public List<String> getResult(String cfId, String urlToControlSystem);
3  }
```

Система управления

Система управления представляет собой Java программу, использующую Spring Framework для реализации своих микросервисных функций.

Система управления отвечает за удаление и добавление фрагментов кода, а также за обработку запросов от других компонентов системы. Для передачи сообщений используется протокол HTTP.

1. code_fragments
   - Описание: Возвращает список [id] фрагментов кода
   - Формат возвращаемых значений: json
   - Пример возвращаемого значения, см. листинг 3.4:

```
{
  "code_fragments":["b642e476-8d79-4783-b1da-a016b0d226aa"]
}

```

Листинг 3.4 – Пример возвращаемого значения запроса code_fragments

   - Пример запроса: GET /code_fragments

Испытания

Начиная с зимы 2025 года СУБФК была запущена на одном из тестовых стендов кафедры параллельных вычислений и использовалась вплоть до момента написания текста ВКР.

После небольшого периода стыковки с другими компонентами система начала работать стабильно и требовала перезагрузки только после выключения самого тестового стенда.

Тем не менее, некоторые компоненты системы ещё не были внедрены на тестовый стенд. Поэтому также были проведены дополнительные тесты на базовую функциональность системы.

В частности была проведена проверка API системы на соответствие зафиксированному режиму работы, также был разработан один плагин, который был успешно внедрен в систему, и была проверена правильность его работы с точки зрения других компонентов.

Генерация программы

Основной сценарий работы с СУБФК, — это генерация программы. Испытание заключалось в оценке возможности автоматически сгенерировать программу, которая бы выполняла задачу вычисления корреляционной свертки сейсмотрасс, используя СУБФК как источник кода. Испытание состоит из следующих шагов:

1. Использовать запрос code_fragments для того чтобы получить идентификатор нужного фрагмента кода
2. Использовать запрос pluginProcedure через plugin ExeStartCProcedure и getNeededContentFor для получения итоговой процедуры запуска и требуемых для запуска корневых файлов фрагмента кода
3. Из возвращаемого значения pluginProcedure составить итоговую процедуру C
4. Вставить эту процедуру, не меняя её, внутрь любой C программы
5. Вызвать эту процедуру в программе так, чтобы она была запущена в одной папке с разархивированным результатом getNeededContentFor
6. Запустить программу
7. Должно запуститься вычисление корреляционной свертки сейсмотрасс
8. В самой программе должна быть возможность получить доступ к результату выполнения корреляционной свертки

Выводы

Испытания прошли успешно — метаинформации и фрагмента кода, которые были получены по API, хватает для того, чтобы автоматизированно сгенерировать программу, не имея информации о внутреннем устройстве фрагмента кода — задача вычисления корреляционной свертки сейсмотрасс успешно запускается, и внутри программы можно получить результат выполнения этой задачи.

СУБФК, периодически взаимодействуя с другими компонентами, работает стабильно с течением времени.

В контексте данной работы такие результаты можно считать **успешными**.

Аппробация

Данная работа была заслушана на конференции

"Молодежные Марчуковские научные чтения 2023" (ММНЧ 2023) - научная конференция молодых ученых (КМУ ИВМиМГ СО РАН)

Заключение

В ходе представленной работы была спроектирована и разработана система управления базой фрагментов кода, также были разработаны необходимые для создания этой системы модели и форматы.

Было проведено тестирование разработанной системы. Практические испытания подтвердили, что предложенное решение представляет собой работающий прототип СУБФК, выполняющий требуемые системой активных знаний от СУБФК функции.

Итоговый разработанный прототип может быть использован в составе системы активных знаний для выполнения функции хранения и стандартизации переиспользования фрагментов кода. Теоретический результат, в свою очередь, может быть использован как основа для разработки собственного решения для СУБФК.

Дальнейшие планы

Разработанное решение имеет области, которые нуждаются в улучшении. По большей части эти улучшения касаются технической части.

В дальнейшие планы работы входит:

- Движение СУБФК в сторону распределенного хранения фрагментов кода.
- Расширение возможностей СУБФК путём добавления новых плагинов.

Факультет информационных технологий
Кафедра параллельных вычислений

Выпускная квалификационная работа бакалавра

Разработка системы управления базой фрагментов кода для системы активных знаний

Направление подготовки 09.03.01 Информатика и вычислительная техника Направленность (профиль): Программная инженерия и компьютерные науки

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Соруководитель ВКР
ст. преп. каф. ПВ ФИТ
Городничев М. А

20.06.2025

Руководитель ВКР
к.т.н, доц. каф. ПВ ФИТ
Перепёлкин В. А.

На защиту выносятся следующие положения:

- Разработана модель фрагмента кода.
- Разработан интерфейс взаимодействия с фрагментами кода.
- Разработана система управления базой фрагментов кода.
- Проведено тестирование разработанной системы управления базой фрагментов кода.

1. code_fragments
    - Описание: Возвращает список [id] фрагментов кода
2. info
    - Описание: Возвращает всю доступную метаинформацию про фрагмент кода
3. Plugins
    - Описание: Возвращает количество и список поддерживаемых плагинов фрагмента кода или список всех доступных плагинов вообще
4. getNeededContentFor
    - Описание: Возвращает все нужные корневые файлы фрагмента кода в .tar архиве в зависимости от плагина
5. pluginProcedure
    - Описание: Возвращает результат работы выбранного плагина для выбранного фрагмента кода
6. add_package
    - Описание: Добавляет пакет в систему.
7. remove_package
    - Описание: Удаляет пакет из системы

1. code_fragments

- Описание: Возвращает список [id] фрагментов кода
- Формат возвращаемых значений: json
- Пример возвращаемого значения, см. листинг 3.4:

```
{
  "code_fragments":["b642e476-8d79-4783-b1da-a016b0d226aa"]
}

```

Листинг 3.4 – Пример возвращаемого значения запроса code_fragments

- Пример запроса: GET /code_fragments

2. info

- Описание: Возвращает всю доступную метаинформацию про фрагмент кода
- Формат возвращаемых значений: json
- Пример возвращаемого значения, см. листинг 3.5:

```
{
 "package": "Convolution",
 "inputVarCount": 3,
 "outputVarCount": 2,
 "description": "Correlation convolution of seismic traces. The report for the resulting correlograms includes the signal-to-noise ratio (SNR) and equivalent signal amplitude (A) in ADC units.",
 "name": "Correlation Convolution of Seismic Traces",
 "inputVarTypes": ["any_path", "any_path", "int"],
 "inputVarDescriptions": [Text file with a list of seismic traces in PC-A format",Reference signal file in PC-A format",Correlation interval in seconds"],
 "outputVarTypes": ["local_path", "local_path"],
 "outputVarDescriptions": ["Work report in .log format","Final convolved seismic trace file"],
 "implementedPlugins": ["ExeStartCProcedure"]
}
```

Листинг 3.5 – Пример возвращаемого значения запроса info

- Дополнительные параметры: ?specific=name (вместо name поставить любой ключ из возврщаемого значения при запросе без specific, также можно комбинировать запросы specific: /info?specific=inputVarCount@outputVarCount@description@implementedPlugins)
- Пример запроса: GET /"id"/info

3. plugins

- Описание: Возвращает количество и список поддерживаемых плагинов фрагмента кода или список всех доступных плагинов вообще
- Формат возвращаемых значений: json
- Пример возвращаемого значения, см. листинг 3.6:

```
{
  "plugins":["ExeStartCProcedure", "getSO"]
}

```

Листинг 3.6 – Пример возвращаемого значения запроса plugins

- Пример запроса: GET /"id"/plugins или /plugins

4. getNeededContentFor

- Описание: Возвращает все нужные корневые файлы фрагмента кода в .tar архиве в зависимости от плагина
- Формат возвращаемых значений: .tar архив. Скачивается архив с именем ”id фрагмента кода”.tar в котором лежит папка src со всеми нужными корневыми файлами фрагмента кода
- Дополнительные параметры: ?plugin=”plugin_name”
- Пример запроса: GET /”id”/getNeededContentFor?plugin=ExeStartCProcedure

5. pluginProcedure

- Описание: Возвращает результат работы выбранного плагина для выбранного фрагмента кода
- Формат возвращаемых значений: Зависит от плагина
- Пример запроса: GET /”id”/pluginProcedure?plugin=ExeStartCProcedure

6. add_package

- Описание: Добавляет пакет в систему.
- Формат входныых значений: packageName, MultipartFile (xml, .tar)
- Пример возвращаемого значения:
- Дополнительная информация: Post запрос. В xml должен быть description файл. В файле должен быть .tar файл в котором лежит папка src с корневыми файлами фрагмента кода
- Пример запроса через curl: curl -X POST .../add_package -H ”Content-Type: multipart/form-dataF ”packageName=packageNameF ”xml=@description.xmlF ”file=@test_send.tar”

7. remove_package

- Описание: Удаляет пакет из системы
- Формат входных значений: параметр URL
- Пример запроса: DELETE /remove_package?packageName=Convolution