ФГБОУ «Рязанский государственный радиотехнический университет
имени В.Ф. Уткина»

РАЗРАБОТКА ПЛАТФОРМЫ ДЛЯ ТРЕНИРОВКИ В РЕШЕНИИ АЛОГРИТМИЧЕСКИХ ЗАДАЧ И АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ КАНДИДАТОВ

Подготовил:
Вашкулатов Н.А. 045

Руководитель:
Ефимов А.И.

Рязань 2024

Актуальность

Сейчас почти на всех собеседованиях в крупные компании для начинающих разработчиков обязательно присутствует решение каких-либо алгоритмических задач.

Постоянная практика в решении алгоритмических задач не только развивает умение быстро и эффективно находить оптимальные решения, но также формирует аналитическое мышление и готовность к промышленной разработке.

Так же компании могут быть заинтересованы в отборе лучших кандидатов и такая платформа может помочь в отборе.

Существующие решения

Платформа	Русский язык	Для решения только реализовать метод	Удобный интерфейс	Быстрая обратная связь	Быстрое создание соревнований	Быстрое создание задач
LeetCode	Нет	Да	Да	Да	Нет	Нет
CodeWars	Нет	Да	Нет	Да	Да	Нет
Codeforces	Да	Нет	Нет	Нет	Нет	Нет
Яндекс Контест	Да	Нет	Да	Нет	Частично	Частично
Желаемое	Да	Да	Да	Да	Да	Да

Функциональность платформы

- Выполнение кода пользователя на сервере;
- Автоматическая проверка решения пользователя при помощи набора тестовых данных и запуска кода;
- Сортировка задач по сложности и темам;
- Создание приватных и публичных задач, в решении которых пользователь может тренироваться;
- Создание приватных и публичных соревнований – контестов;
- Для решения задачи пользователю нужно лишь реализовать метод, без реализации ввода/вывода в консоль;
- Не является платформой для проведения олимпиадных соревнований.

Компиляция и выполнение кода на сервере

Для автоматизации этих процессов используется отдельное приложение и Docker

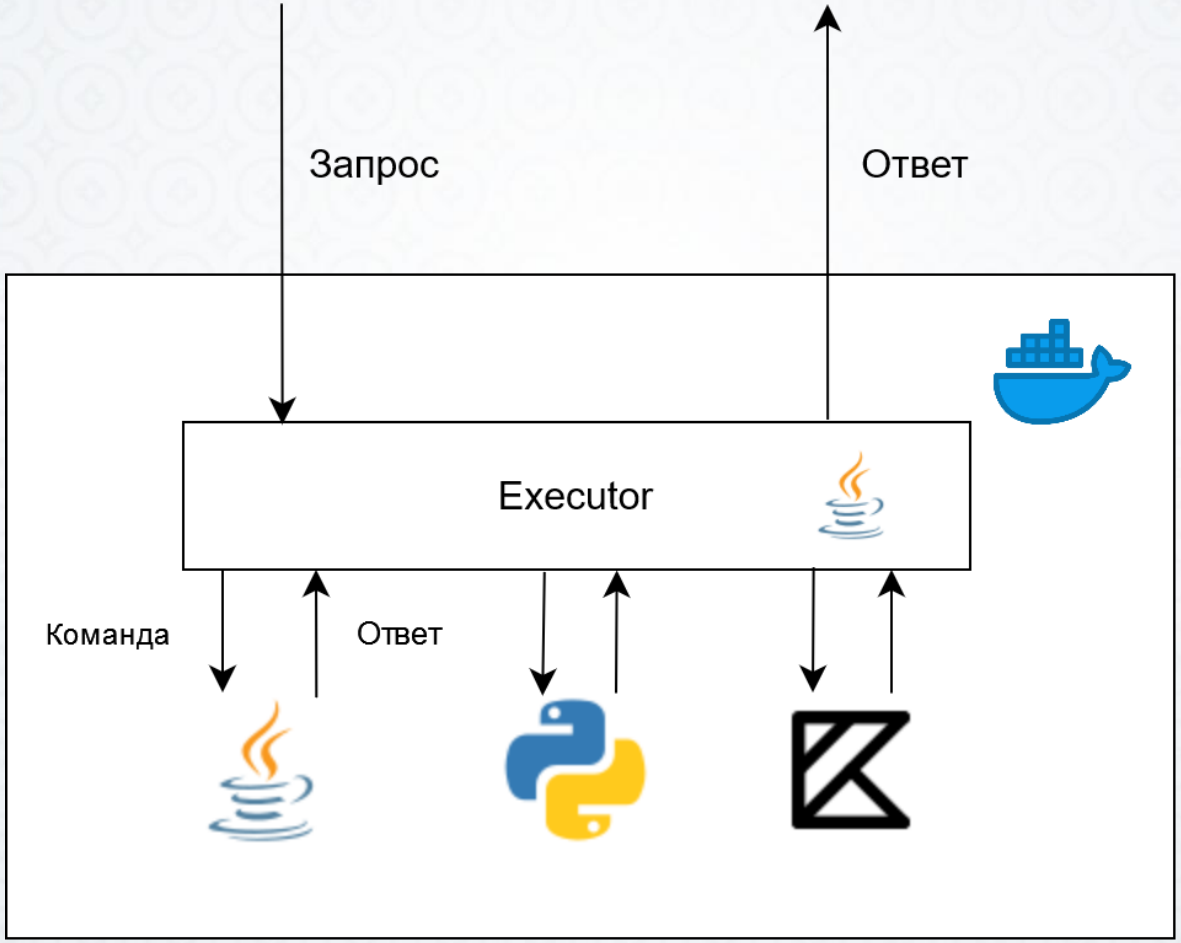
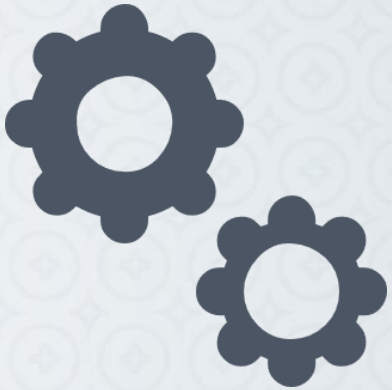


Рисунок 1 – Сервис выполнения



```
FROM openjdk:17-jdk

RUN  microdnf  install -y python3
RUN  microdnf install gcc python3-devel
RUN python3 -m pip install psutil
```

Рисунок 2 – Часть Dockerfile для образа

Структура задачи

- Типы данных для входных параметров и результата
- Тестовые данные, проверяющие правильность решения.
- Шаблон решения (рисунок 1 - 2) для каждого возможного языка

```
1 class Solution{
2     public int add(int a, int b){
3
4     }
5 }
```

Рисунок 3 - Пример шаблона

```
1 class Solution:
2     def add(s,a,b):
```

Рисунок 4 - Пример шаблона

Подготовка решения пользователя

- Платформа берет на себя ввод и вывод данных - мы не можем просто запустить код пользователя и отправить тестовые данные.
- Для чтения, измерения времени, преобразования типов входных данных и вывода результатов решения в консоль используется специальный заготовленный код – драйвер.
- Все эти усложнения нужны для того, чтобы стандартизировать формат входных и выходных данных, благодаря чему процесс тестирования не будет отличаться от ЯП к ЯП, а так же относительно просто добавить не стандартный тип.

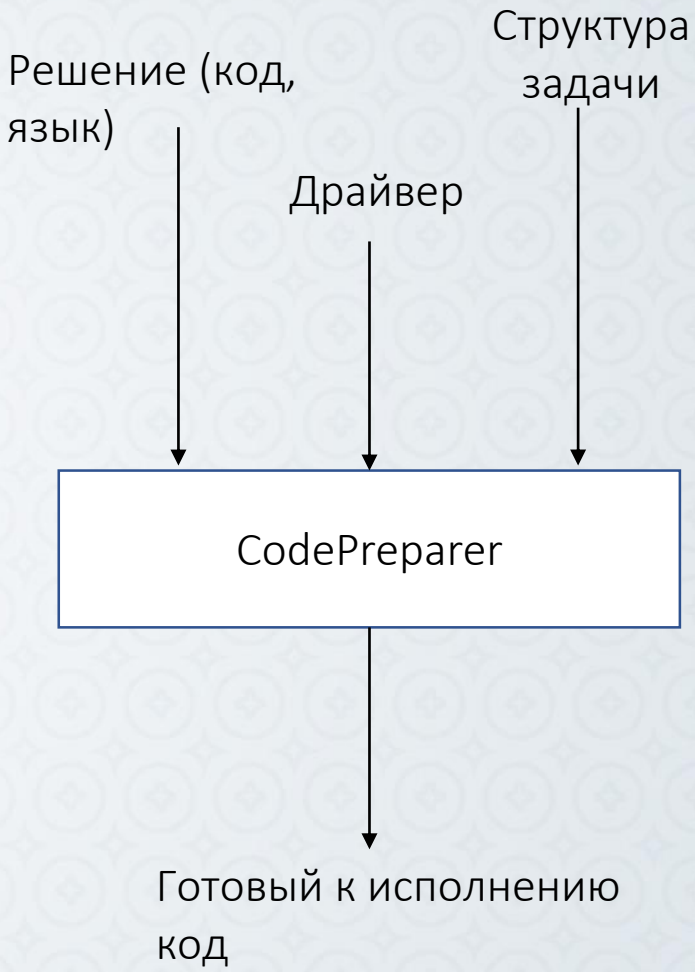


Рисунок 5 – Подготовка кода

Проверка правильности решения

- Каждая задача содержит набор тестовых данных, который проверяет правильность решений, отправляемых пользователем.
- Тест состоит из массива входных данных и соответствующего ответа.
- Может быть несколько вариантов ответов для одних входных данных.
- Входные и выходные данные это просто строки, что упрощает тестирование.

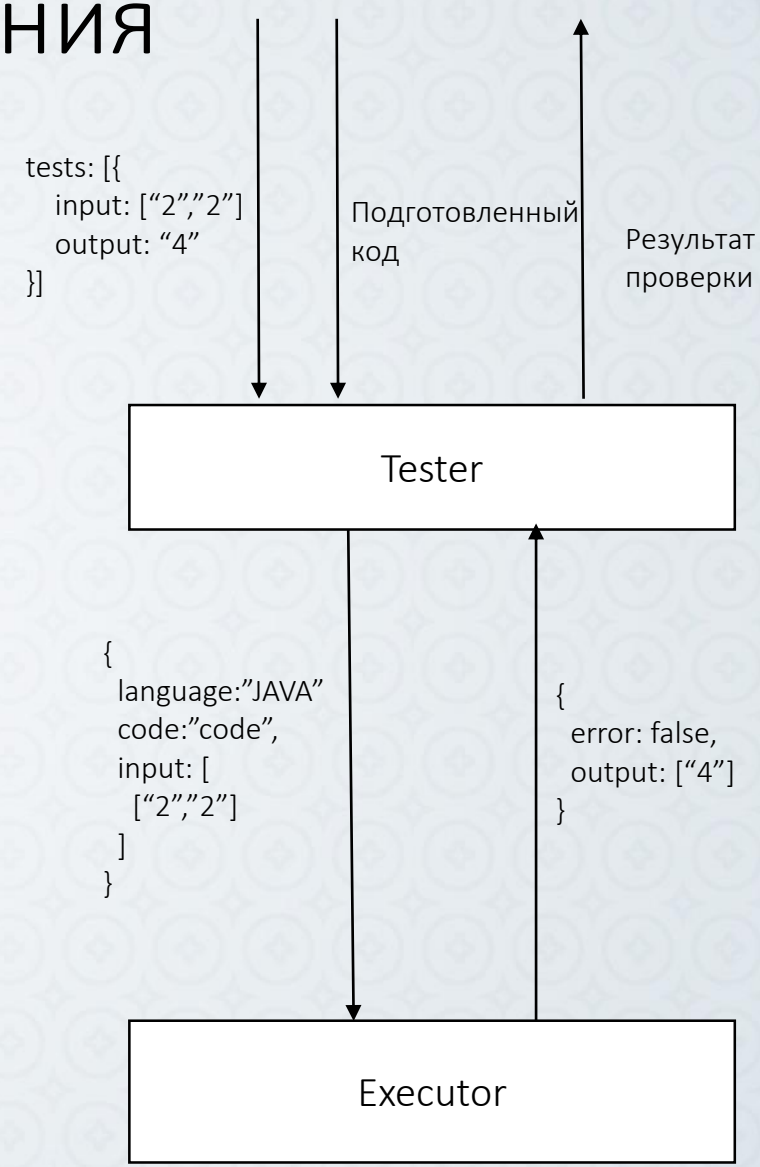


Рисунок 6 – Проверка решения

Архитектура платформы

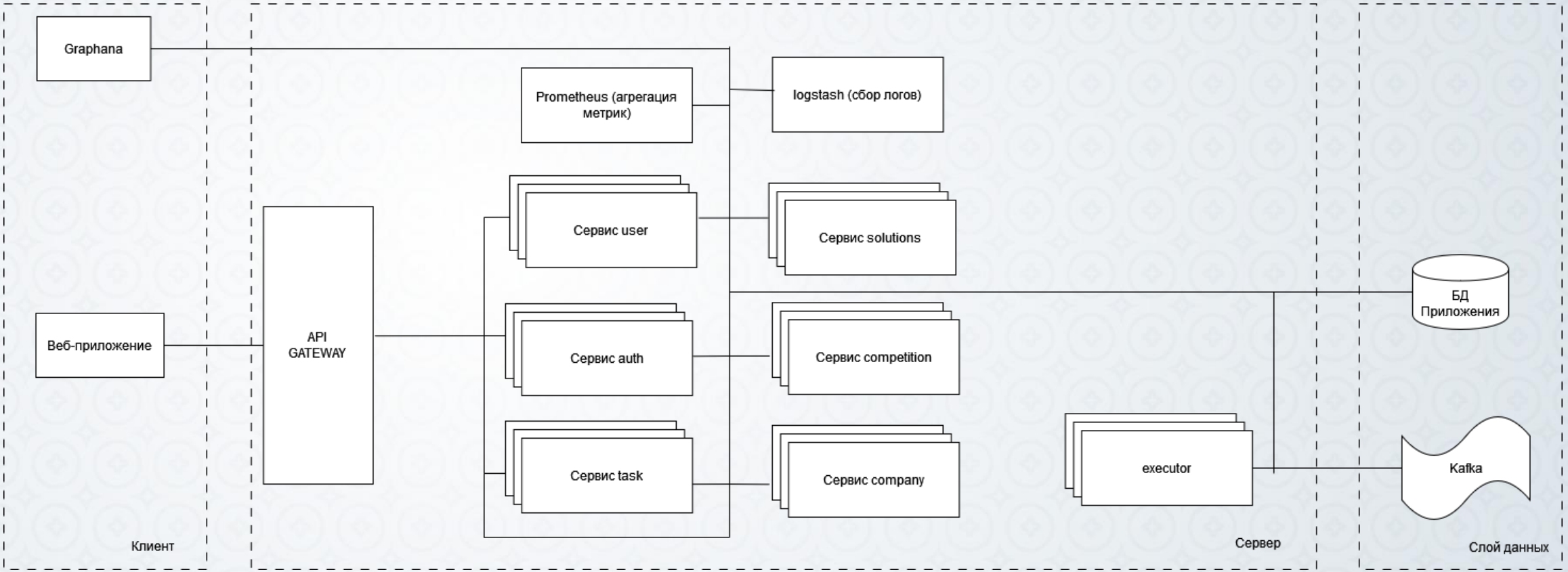


Рисунок 7 – Архитектура

Инфраструктура

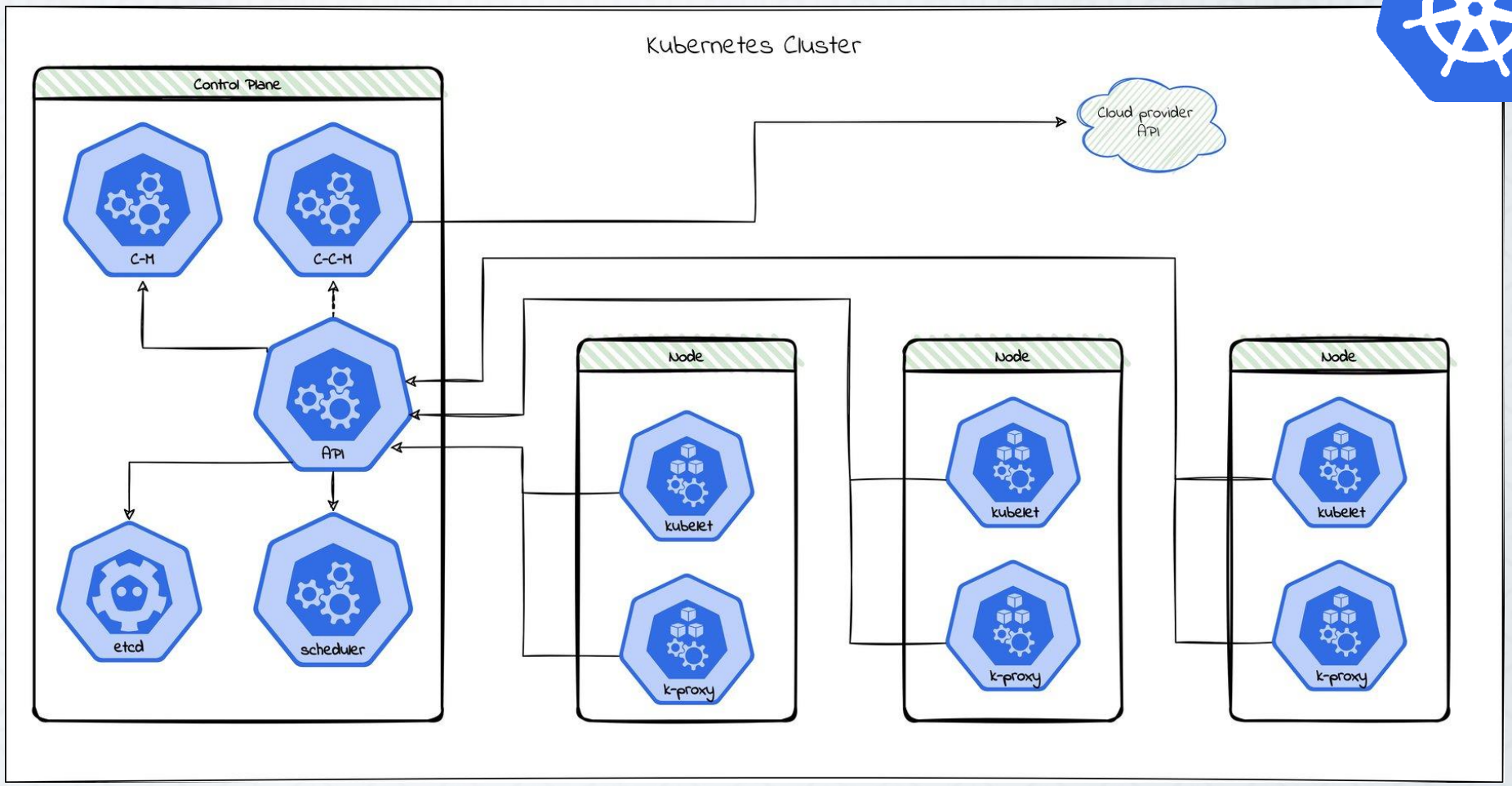


Рисунок 8 – Инфраструктура

Используемые технологии

- Kotlin
- Kora/Spring
- Kafka
- PostgreSQL или Cassandra
- Kubernetes
- Grahana
- Prometheus

Спасибо за внимание

https://codestopen.r3nny.ru/task

Процесс проверки решения

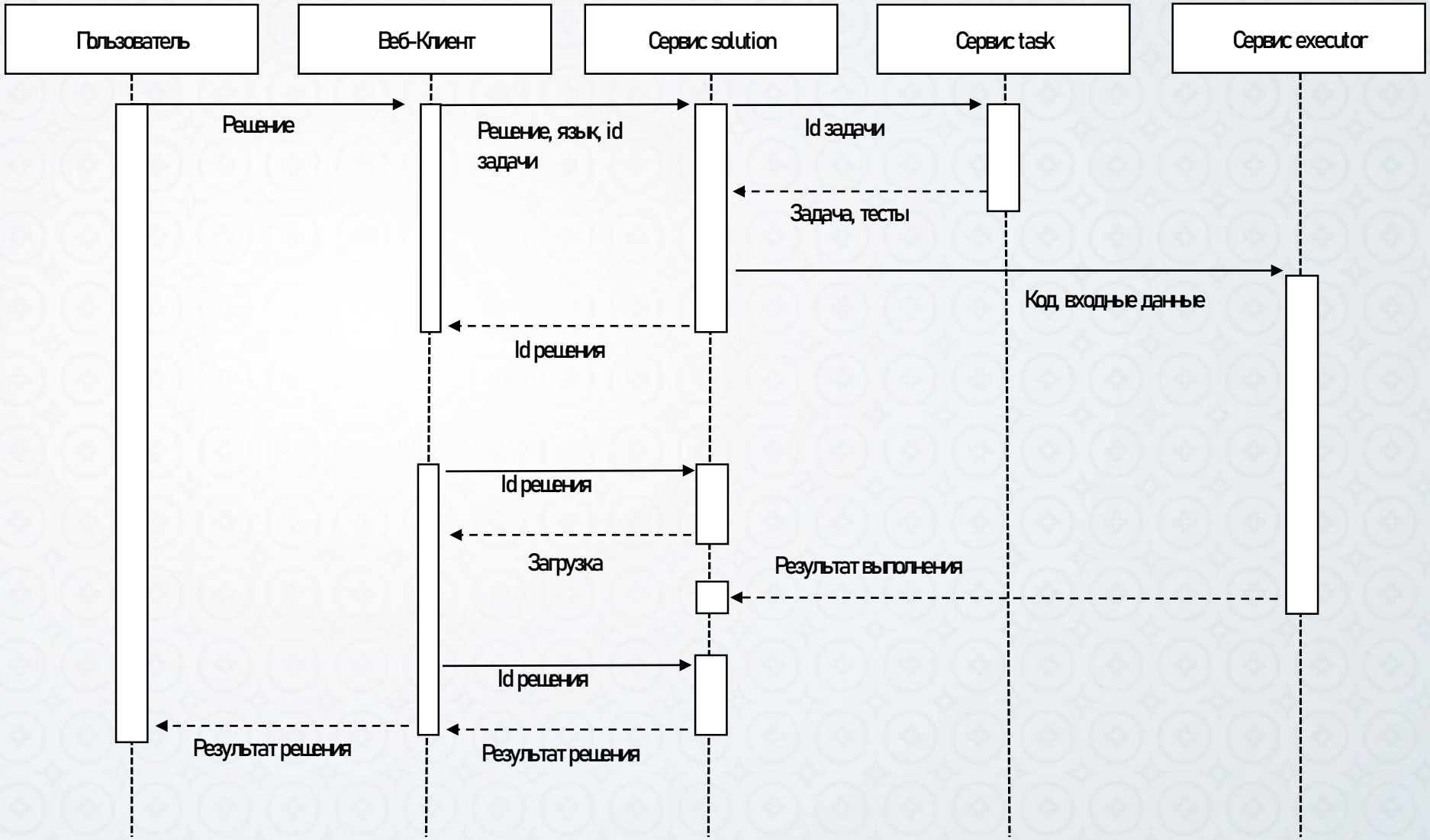


Рисунок 9 – Процесс проверки решения

Структура драйвера

1. Место куда вставляется код пользователя.
2. Функции чтения данных определенных типов из консоли.
3. Секция для чтения входных параметров (вызов функций их п. 2)
4. Начало отсчета времени
5. Вызов кода, отправленного пользователем, с передачей параметров
6. Конец отсчета времени
7. Подсчет памяти
8. Вывод результатов в консоль

```
${solution}  1)

def READ_STRING(): 2)
    return input()

def READ_INTEGER_ARR(): 2)
    input_str = input()
    if input_str == "[]":
        return []
    input_str = input_str.replace("[", "").replace("]", "").replace(" ", "")
    elements = input_str.split(",")
    arr = [int(element) for element in elements]
    return arr

if __name__ == '__main__':
${paramsInputSection} 3)

    s = Solution()
    start_time = time.time() 4)
    ret = Solution.${methodName}(s, ${paramList}) 5)
    end_time = time.time() 6)
    process = psutil.Process()
    memory_info = memory_info = process.memory_info() 7)
    used_memory_kb = memory_info.rss / (1024*1024)
    execution_time = (end_time - start_time) * 1e9
    print(ret) 8)
    print(execution_time)
    print(used_memory_kb)
```