**УДК 004.42**

**РАЗРАБОТКА**

**ПЛАТФОРМЫ ДЛЯ ТРЕНИРОВКИ РЕШЕНИЯ АЛГОРИТМИЧЕСКИХ ЗАДАЧ**

***Вашкулатов Н.А.***

***Рязанский государственный радиотехнический университет  
имени В.Ф. Уткина***

В данной работе разработана платформа для тренировки решения алгоритмических задач на нескольких языках с взаимодействием через WEB-интерфейс.

***Ключевые слова:*** *WEB, Java, Spring Framework, алгоритмы, задачи, Kafka, Python, языки программирования.*

**Введение**

Современное информационное общество сталкивается с постоянным ростом потребности в квалифицированных IT-специалистах. С увеличением числа кандидатов, стремящихся занять вакансии в области программирования, выявление наилучших талантов и их эффективная оценка становятся важнейшими задачами для компаний.

Сейчас на большинстве собеседований обязательно присутствуют вопросы про различные алгоритмы или решение задач прямо во время собеседования. В связи с этим большинству начинающих программистов необходимо постоянно усовершенствовать свои навыки в решении классических алгоритмических задач. Для этой задачи идеально подходит WEB платформа на которой пользователь сможет находить задачи, писать код и проверять решения с любого устройства с WEB браузером, без установки компиляторов, сред разработки и других программ.

В данной работе разрабатывается платформа с WEB-интерфейсом на которой можно писать код, отправлять решение на сервер, проверять ошибки компиляции, выполнения и тестирования.

**Разработка платформы для тренировки решения алгоритмических задач**

Основными сущностями на нашей платформе являются: пользователь, решение, задача, тест, пример данных алгоритмов. Получим следующую ER диаграмму (рисунок 1).

Рисунок 1 – ER диграмма

Из ER диаграммы составим схему базы данных с необходимыми таблицами для связи и технической составляющей. Как видно из схемы базы данных реляционное отображение задачи является весьма непростым и в связи с этим в дальнейшем задача и тесты будут вынесены в документно-ориентированную базу данных. Так же в ранних версиях был убран пользователь.

Распределим операции над сущностями в раздельные сервисы, добавим дополнительные технические приложения, получим приблизительную архитектуру нашей платформы (рисунок 3).

Сервис task отвечает за работу над задачами такими как:

* добавление или удаление задач
* получение информации о задаче
* добавление или удаление тестов

Сервис solution отвечает за:

* формирование данных для отправки в сервис решения
* сохранение или обновление попыток
* формирование данных о попытке для пользователя

Сервис executor отвечает за:

* выполнение и оценку правильности оценки пользователя

Сервис user отвечает за работу с данными польщователя.

API шлюз служит точкой входа в серверную часть платформы и перенаправляет запросы между различными экземплярами сервисов, то есть выполняет роль балансировщика нагрузки. А также шлюз проверяет данные пользователя при доступе к защищенным методам сервисов.

Стек ELK необходим для централизованного сбора, хранения и поиска по логам и другим машинным данным производимых платформой.

Для взаимодействия с сервисом, который выполняет решение задач используется Apache Kafka.

Eureka используется для обнаружения и хранения адресов экземпляров сервисов платформы. Это позволяет запускать множество экземпляров сервисов на любых машинах по всему миру, увеличивая производительность платформы.

Рассмотрим более детально процесс отправки решения задачи (рисунок - 4).

Очень важной составляющей процесса решения задачи является «Драйвер». Драйвер – это основа для решения любой задачи. Он содержит в себе код, который выполняет счет времени выполнения и занимаемой памяти. Так же драйвер отвечает за чтение данных с консоли, преобразование их к нужному типу, вызов метода, который реализовал пользователь с передачей нужных параметров и вывод необходимых данных в консоль, для того, чтобы сервис выполнения прочитал их и сравнил с тестами. Для каждого языка пишется свой драйвер. Для каждой задачи так же составляется свой драйвер на основе типов данных, используемых в решаемой задачи и их количества. Пример драйвера на языке java:

Для выполнения в промышленной среде каждый сервис собирается в образ Docker при помощи GitHub Actions и публикуется в реестре контейнеров GitHub. После чего на машине создается специальный файл, в котором описаны какие образы использовать, а также различные параметры и настройки сетевых взаимодействий.

**Заключение**

Таким образом, была разработана основа платформы для тренировки решения алгоритмических задач с WEB-интерфейсом. Исходных код доступен на GitHub. Платформа доступна по ссылке

**Список используемых источников**

1. Алгоритмы и жизнеритмы на языке ДРАКОН. Разработка алгоритмов. Безошибочные алгоритмы. — М., 2019. — 374 с. — Иллюстраций:165
2. The Human Revolution in Understanding Programs. — 2011. — 33 с.
3. QT 6 Разработка оконных приложений на С++. — СПб.: БХВ-Петербург, 2022. — 512 с.: ил. — (Профессиональное программирование)
4. QT 6 Разработка оконных приложений на С++. — СПб.: БХВ-Петербург, 2022. — 512 с.: ил. — (Профессиональное программирование)
5. QT 6 Разработка оконных приложений на С++. — СПб.: БХВ-Петербург, 2022. — 512 с.: ил. — (Профессиональное программирование)