## Сравнение существующих аналогов

**LeetCode**  
Одна из самых известных и популярных платформ для решения задач на алгоритмы и структуры данных. Содержит теоретические материалы, но не позволяет создавать свои задачи.  
Имеет возможность протестировать программу на своих тестовых данных.  
Тестовые данные указываются в текстовом формате.   
Входные параметры будут десериализованы и поданы в функцию, которая производит вычисления.  
**CodeWars**  
Не содержит теоретических материалов, зато позволяет создавать свои задачи.  
Особенность сервиса - тестирование программ через автотесты.   
То есть автор задачи кодом пишет проверку пользовательского решения.  
**Информатикс**  
Сервис на русском языке, но интерфейс устаревший и недружелюбный.  
Нет встроенного редактора кода. Код приходить писать и отлаживать локально. Решение загружается как файл с кодом.   
Имеет теоретические материалы, но навигация затруднена и приходится искать задачи по номерам.   
Собственные тестовые данные указать нельзя.   
Решение проверяется через консольный ввод/вывод.   
Добавить задачу на сайт может не каждый. Как-то можно через заявку, но я этим не занимался.  
**HackerRank**  
Теоретических материалов нет, но позволяет создавать свои задачи.  
Ориентирован на алгоритмические задачи.  
Проверяет решения по консольному вводу/выводу.  
Позволяет написать код на python, java или c++ для оценки решений по баллам.

**Таблица**

Останавливаться на ней хочу, потому что все, что тут написано, уже было ранее мной озвучено при рассмотрении каждой платформы. Хочу только обратить внимание, что моя образовательная платформа выглядит крайне выгодно, если сравнивать с другими платформами по этим очень удобные для меня критериям.

**Совместное редактирование кода**Для реализации я буду использовать типа CRDT. Слева схема реализации по протоколу WebRTC. Протокол попытается организовать прямое соединение между двумя браузерами для минимизации задержки и нагрузки на сервер, иначе протокол откатится до использования сервера в качестве посредника. Такой случай указан справа. Можно сразу реализовать так – просто сервер посередине, который соединяет клиентов между собой.

## Выполнение недоверенного кода

Пользователи отправляют свой код с решением мне на сервер, и чтобы проверить их решение, мне нужно его запустить у себя на образовательной платформе.

Любой пользовательский код, отправляемый на платформу, является недоверенным   
и может являться вредоносным.  
  
Я нашел и изучил 2 статьи по тому, как применяется контейнеризация программ в международных олимпиадах по информатике.   
Olympiads in Informatics, 2012, Vol. 6, 100–109 © 2012 Vilnius University  
A New Contest Sandbox  
Olympiads in Informatics, 2021 Vol. 15, 37–52 © 2021 IOI, Vilnius University  
Security of Grading Systems  
  
В первой статье авторы описали свой вариант контейнеризации основанный на linux control group и namespaces.  
Те же механизмы используются в системе контейнеризации docker, которую использую я, поэтому я применил требования из статей к докеру.  
Во второй статье описывается безопасность системы оценки задач.   
Там указывается несколько аспектов безопасности, на которые нужно обратить внимание.  
Основное это всяческое ограничение возможностей для недоверенного кода.  
**Начнем с ограничений файловой системы.**  
Докер по умолчанию создает изолированную файловую системы для каждого контейнера.  
**Ограничиваем размер файловой системы.**  
`--storage-opt size=120G` поддерживается на btrfs, overlay2, windowsfilter, zfs.  
**Можно ограничить количество операций ввода-вывода.**   
`device-read-IOPS device-write-IOPS`   
Но если фс расположена в оперативной памяти, то смысла особого нет.   
**Ограничение памяти и процессора.**  
Даже если задание по программированию не предусматривает ограничение памяти,   
то все равно память нужно ограничить, чтобы не мешать другим контейнерам и хостовым процессам. По процессору то же самое.   
**Ограничение потоков и процессов.**  
Чрезмерное число процессов может создать перегрузку для планировщика задач, поэтому рекомендуется ограничивать количество процессов и файлов.  
**Отправка сигналов и межпроцессное взаимодействие**. Это уже сделано докером.   
**Сетевое взаимодействие** - не указано в статье. `--network=none`