Санкт-Петербургский Государственный Университет
Математико-механический факультет

Кафедра Системного Программирования

Кладов Алексей Александрович

Разработка системы проверки упражнений для образовательной платформы

Дипломная работа

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Терехов А. Н.

Научный руководитель:
Луцив Д. В.

Рецензент:
Вяххи Н. И.

Санкт-Петербург
2014

SAINT-PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty

Chair of Software Engineering

Aleksei Kladov

Quiz checking system for educational engine

Graduation Thesis

Admitted for defence.
Head of the chair:
professor Andrey Terekhov

Scientific supervisor:

Reviewer:
Nikolay Vyahhi

Saint-Petersburg
2014

Оглавление

Введение

В последнее время технологии Интернет активно используются чтобы улучшить качество предоставляемого образования. Комплекс технологий и продуктов, осуществляющих такое улучшение, получил название электронного обучения.

Важно обратить внимание на отличие электронного обучения от традиционного.

Особенности электронного обучения

Во-первых, это большое количество студентов в одном потоке, как правило достигающее нескольких десятков тысяч человек[7].

Во-вторых, хотя учебные материалы уникальны для каждого курса, они, как правило, очень высокого качества — видео лекции, субтитры, аннотированые слайды, книги.

В-третьих, мотивация студентов курса зачастую не очень высокая. Это связанно с тем, что записаться на Интернет курс значительно проще, чем поступить в университет[2].

В-четвёртых, свободный формат электронного обучения позволяет учится в удобное для студента время.

История

Пожалуй, история компьютерного образования началась с началом распростронения персональных компьютеров, если не раньше, однако можно выделить следующие вехи:

Walter Lewin – профессор физики из MIT, лекции которого показывались по местному телевидению

open courseware – MIT сделал все учебные материалы доступными в Интернет.

Udacity – первый стартап, с MOOC в настоящем его понимании.

Coursera – наиболее успешная на настоящий момент платформа для online образования

Проблемы

Электронному обучению присущи некоторые специфичные проблемы. Среди них можно выделить следующие.

- Ограничены возможности проверки знаний. Так как упражнения должны проверяются автоматически, то их набор часто ограничен.

	год основания	количество студентов
Udacity	2011	1.6 млн.
Coursera	2012	7.1 млн.
edX	2012	2.1 млн.

Таблица 1: MOOC платформы

- Большая стоимость создания электронного курса. Создание курса требует значительных затрат на запись видео лекций, оформление электронного конспекта и создание набора упражнений.

- Для электронного обучения характерен значительно меньший процент студентов, заканчивающих курс[2].

Стоит заметить, что в электронном образовании лекционный материал может быть представлен очень хорошо, зачастую лучше, чем в аналогичном по содержанию классическом курсе. Так как студент сам активно взаимодействует с лекционным материалом, он может изучать его с удобной для себя скоростью. Можно сказать, что лекции хорошо "масштабируются" по количеству студентов.

При этом практическая часть обучения — решение задач и выполнение упражнений — в электронном обучении реализуется значительно хуже. Дело в том, что, в связи с большим количеством студентов, упражнения необходимо проверять без участия преподавателя. Но не для каждого типа упражнения придуман способ подобной автоматизации.

Существующие платформы

Существует большое количество платформ для MOOC. В таблице 1 приведена информация про наиболее интересные из них.

Udacity Первая платформа для MOOC.

Coursera Крупнейшая на настоящий момент платформа. Некоторые курсы Coursera засчитываются в американских университетах.

edX Платформа с открытым исходным кодом.

1. Платформа Stepic

1.1. Статус проекта

Stepic[6] это молодой проект, развивающийся в рамках компании JetBrains. Разработка проекта стартовала в 2013 году. На текущий момент в проекте занято семь человек.

1.2. Использование

На текущий момент Stepic используют 23 тысячи студентов. На платформе создано более 400 уроков, многие из которых объеденены в курсы. Из завершившихся курсов стоит отметить следующие.

Алгоритмы в биоинформатике Этот англоязычный курс проходил одновременно с соответствующим курсом на Coursera. В данном случае Stepic использовался в качестве дополнения к Coursera, как платформа для упражнений. Курс состоял из большого количества текстовых материалов и упражнений на програмирование из области алгоритмической биологии. В нём приняли участие более 10 тысяч человек со всего мира.

Алгоритмы и структуры данных Это русскоязычный, зарытый на настоящий момент курс, который использовался для предварительной оценки знаний абитуриентов Computer Science Center в Санкт-Петербурге. В нём приняли участие 500 человек из Петербурга. Курс состоит из видео лекций и различных типов упражнений. Наиболее часто использовались упражнения на программирование и упражнения со свободным ответом.

1.3. Возможности платформы

Единицей учебного материала на Stepic является урок — набор упражнений и/или теории, представленный в виде слайдов. Максимальное количество слайдов в уроке — 16 штук. Цель урока — изучить одну концепцию.

Уроки можно группировать в курсы. У курса две задачи. Первая это организация и упорядочивание уроков. Как правило уроки в курсе разбиты на несколько последовательных блоков-недель. Вторая задача это организация групп пользователей. С курсом можно связать множество проходящих его студентов, назначить в курсе разные стоимости и крайние сроки для сдачи упражнений.

Stepic ориентирован на интеграцию и сотрудничество с другими инструментами online образования. Для этого поддерживаются стандарты oEmbed и LTI. Именно с

их помощью в курсах на Coursera можно использовать Stepic в качестве платформы для упражнений.

Разнообразие типов упражнений является отличительной особенностью платформы Stepic, поэтому важна лёгкость расширения набора доступных разновидностей упражнений.

1.4. Технологи и инструмены

В качестве базы данных используется MySQL. Разработка серверной части ведётся на языке Python 3, с использованием фреймворка Django. Клиентская часть разрабатывается на CoffeeScript, с использованием Ember.js.

Также используются Celery для распределённого выполнения заданий, CodeJail в качестве основы системы изолированного исполнения кода, Sympy для символьных вычислений.

2. Постановка задачи

Целью работы является реализация системы для создания и проверки упражнений для образовательной платформы Stepic, с возможностью легко добавлять новые типы упражнений, в том числе и сторонним разработчикам.

Для достижения этой цели были сформулированы следующие задачи.

- Обеспечить возможность лёгкого расширения набора типов упражнений сторонними разработчиками (реализовать соответствующий API к платформе Stepic и фреймворк для разработки).
- Реализовать с помощью разработанного фреймворка в Stepic типы упражнений, часто встречающихся в других образовательных платформах и проверить их работу на практике.
- Реализовать возможность масштабирования и изолированного исполнения потенциально не безопасного кода упражнений.

3. Фреймворк для создания новых типов упражнений

Разработка фреймворка была начата с анализа упражнений в различных платформах.

3.1. Общий вид упражнения

В результате изучения существующих типов упражнений было выявлено, что взаимодействие пользователя с упражнением можно описать набором следующий общих шагов.

Шаг 1. Пользователь читает условие упражнения. Условие представляет собой форматированный текст. При этом в условие можно внедрять параметры упражнения, например ограничения или приемры решений для маленьких наборов данных.

Шаг 2. Пользователь нажимает на кнопку “начать решать”. После этого пользователю представлен один из вариантов входных данных. Формы представления входных данных могут сильно отличаться. Например, это может быть ссылка на файл, или набор возможных вариантов ответов. Часто бывает так, что входных данных вообще нет, то есть, при каждой попытке верным будет один и тот же ответ.

Шаг 3. Пользователь вводит свой ответ. Вид ответа также зависит от упражнения. Например это может быть слово или словосочетание на естественном языке, математическая формула, фрагмент кода на каком-нибудь языке программирования или перестановка элементов списка.

Шаг 4. Пользователь нажимает на кнопку “отправить”. В этом случае, сначала происходит первичная проверка ответа на корректность. Если ответ очевидно не корректный (например, ответ просто пустой) то, скорее всего, кнопка была нажата случайно. В этом случае пользователю предлагается ввести ответ. Если же первичная проверка ответа прошла успешно, то он отправляется на сервер, где происходит проверка.

Шаг 5. В результате этой проверки оценивается правильность ответа. Её можно оценивать по шкале от 0 до 1 и в случае необходимости переводить этот первичный в любую шкалу. Также иногда необходимо дать комментарий к ответу пользователя. Например, если пользователь совершил часто встречающуюся ошибку, то можно объяснить, почему этот вариант ответа является неверным.

Шаг 6. Пользователь видит оценку и подсказку, если она есть.

3.2. Оптимизация упражнений

Входные данные упражнения отличаются от попытки к попытке и генерируется случайным образом на сервере. При этом для некоторых типов упражнений создания входных данных может занимать существенное время. Поэтому целесообразно создавать набор входных данных заранее, и в момент начала решения мгновенно выдавать пользователю заранее заготовленный экземпляр входных данных.

В принципе, для проверки решения пользователя должно быть достаточно входных данных, ведь из них можно получить правильный ответ. Однако стоит обратить внимание на то, что проверка ответа в таком случае может быть долгой, ведь будет необходимо заново решить упражнение. Вместе с тем, проверку надо выполнять быстро, чтобы как можно быстрее обеспечить пользователю обратную связь. Таким образом, оказывается эффективным и удобным вместе с входными данными создавать ключ к ним, который позволяет быстро проверить решение пользователя.

3.3. Вызовы API

Из описанных выше шагов можно выделить следующий общий набор операций, необходимых для работы любого упражнения.

- Создание экземпляра упражнения из исходных данных.
- Создание пары входные данные/ключ.
- Отображение входных данных для пользователя.
- Получение ответа пользователя.
- Первичная проверка ответа.
- Проверка ответа пользователя при помощи ключа, оценка вещественным числом из интервала [0; 1] и генерация текстовой подсказки.

3.4. Система модулей

Для подсистемы упражнений крайне важна простота и модульность, так как упражнения для курса могут создаваться авторами курса, которые хорошо разбираются в своей предметной области, но при этом ничего не знают об устройстве платформы. Поэтому каждый тип упражнения это модуль, никак не зависящий от остальной платформы. Более того, эти модули можно использовать и без Stepic.

Коллекция существующих модулей доступна на github в публичном репозитории[9]. Репозиторий открыт для добавления новых модулей.

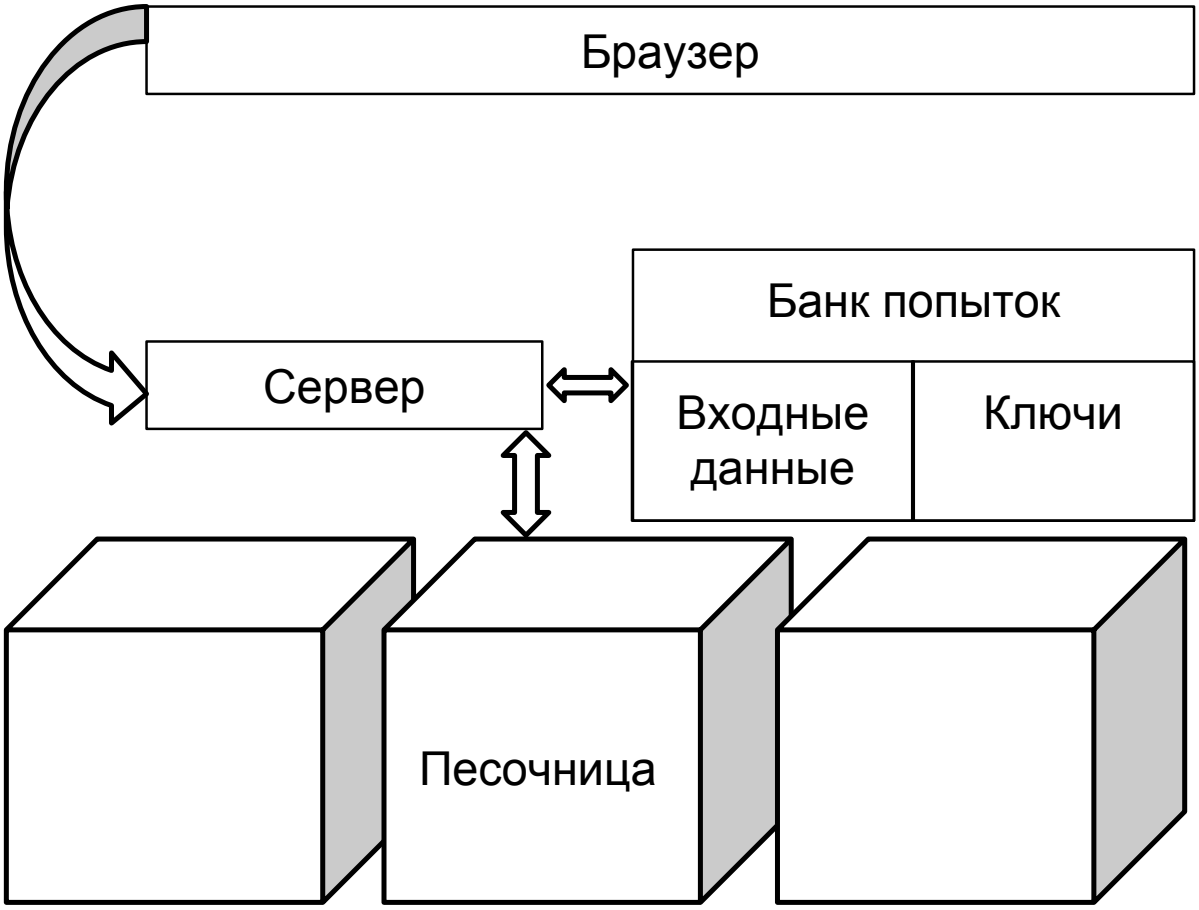

Рис. 1: Архитектура решения

3.5. Архитектура решения

Привлекательной кажется идея реализовывать упражнения целиком в клиенте, и сообщать на сервер только результаты проверки. Это позволило бы реализовывать упражнения целиком на одном языке, и обеспечило бы хороший опыт взаимодействия, так как не было бы задержек из-за коммуникации между клиентом и сервером. Однако у такого решения есть существенные недостатки. Во-первых, можно легко симулировать решение упражнения, не решая его. Во-вторых, для решения некоторых упражнений может понадобиться существенные вычислительные затраты.

Поэтому для реализации упражнений была выбрана клиент-серверная архитектура, в которой клиент отвечает за взаимодействие с пользователем, а сервер за создание входных данных и проверку ответов. При этом клиент общается с сервером при помощи ajax-запросов и поэтому для работы упражнения не требуется перезагрузка страницы.

Для ускорения упражнений используется банк попыток — набор заранее созданных пар входные данные/ключ.

Исполнение кода упражнений осуществляется в изолированных песочницах.

3.6. Детали реализации

Для сохранения данных упражнений был выбран формат JSON. Он удобен по двум причинам. Во-первых, для добавления нового типа упражнений нет необходимости менять схему базы данных, что позволяет отделить упражнения от остальной платформы. Во-вторых JSON можно использовать в качестве формата сообщений, которыми общаются клиент и сервер.

Для реализации серверной части выбран язык Python 3. Это простой и удобный язык сверхвысокого уровня, главное преимущество которого заключается в уменьшении затрат на разработку модулей [4].

Для создания серверной части разработчику модуля упражнения необходимо определить наследника класса BaseQuiz.

Также при помощи eDSL необходимо описать JSON формат для исходных данных упражнения, входных данных и ответа.

Клиентская часть реализуется на JavaScript или на CoffeeScript. Для реализации клиентской части упражнения необходимо определить функции отрисовки начальных данных для интерфейса редактирования и функции отрисовки входных данных и ответа для интерфейса решения. Также есть возможность создавать клиентскую часть упражнения в виде компоненты ember.

3.7. Инструменты для разработки

Для упрощения разработки модулей упражнений был разработан ряд инструментов.

Сервер для разработки, который позволяет запускать упражнения без Stepic. Сервер автоматически перезагружается при изменении исходного кода. С его помощью можно проверить как серверную, так и клиентскую части модуля упражнения.

Шаблон упражнения, при помощи которого можно мгновенно начать разрабатывать свой модуль.

Документация, опубликована на readthedocs[8].

4. Реализованные типы упражнений

С помощью фреймворка в Stepic были разработны следующие типы упражнений:

- Choice
- Code
- Dataset
- Free Answer
- Math
- Number
- Sorting
- String

4.1. Примеры конкретных упражнений

Рассмотрим возможности фреймворка на примере некоторых упражнений.

Choice просит студента выбрать среди предложенных вариантов ответа правильные. У этого квиза есть много опций – размер выборки, её случайность, количество правильных ответов в выборке. Это наиболее часто встречающийся в различных платформах тип квиза из-за своей простоты. Однако у него есть свои недостатки – например, студент может просто угадывать ответы. И студент не может придумать своё, по настоящему оригинальное решение

Dataset предлагает скачать текстовые входные данные и требует текстового ответа. Этот тип упражнений подходить для некоторых упражнений на программирование. Он хорош тем, что позволяет использовать любой инструмент для решения. Но у этого типа есть и недостатки. Размер входных данных не может быть большим, чтобы их удобно было скачивать через Интернет. Сложно проконтролировать эффективность пользовательского решения. Студенту необходимо иметь нужные компиляторы на своей машине.

Code дополняет dataset. В этом упражнении студент должен послать код для решения задачи, которы затем будет исполнен на сервере, с замерами времени и памяти. Это удобно для проверки эффективности решения, однако необходимо решать задачу на одном из поддерживаемых языков программирования(Java, Python, C++, Octave).

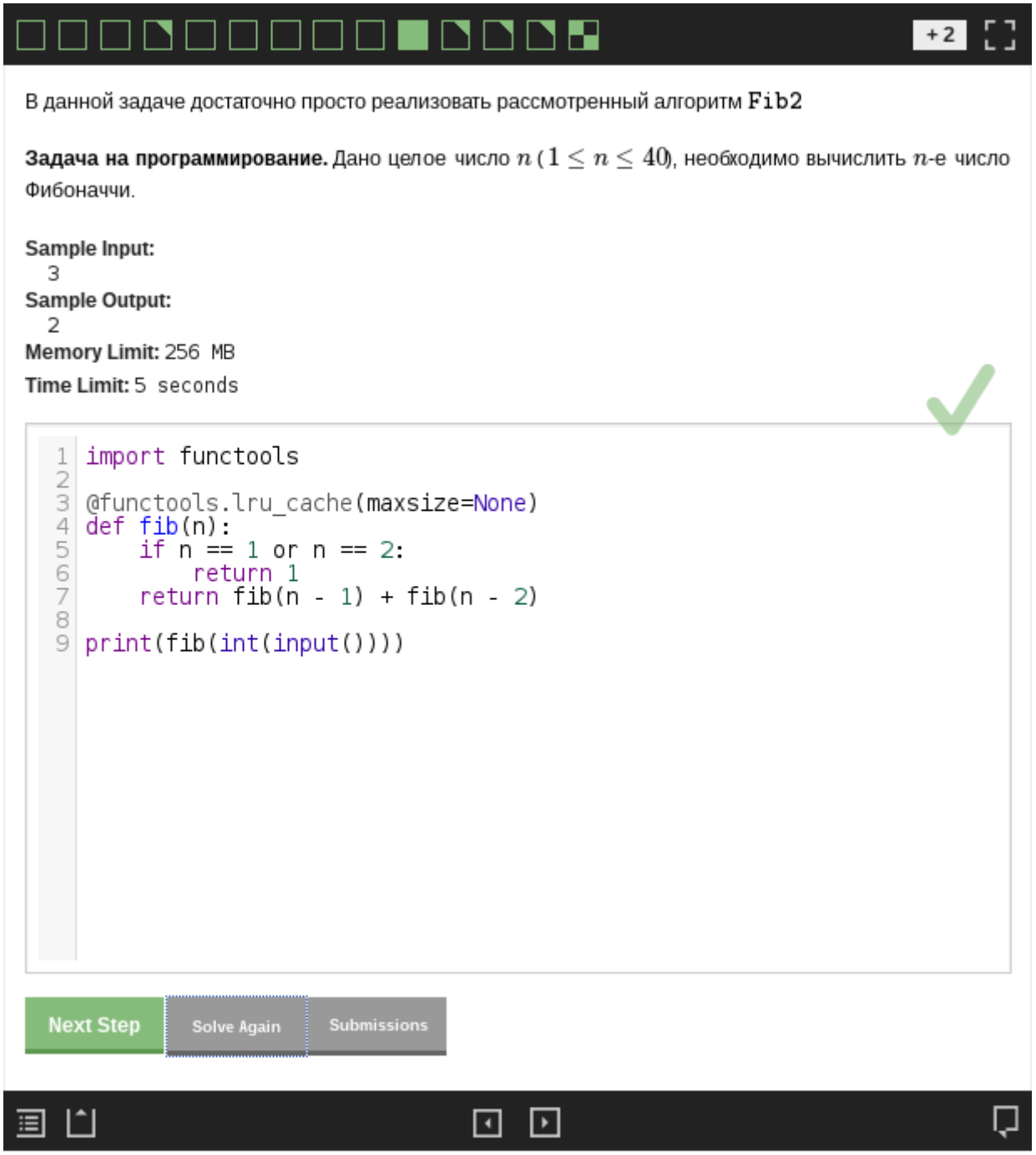

Рис. 2: Пример упражнения (Code)

4.2. Использование упражнений на практике

Созданные упражнения были успешно использованы на практике, при этом в разных курсах использовались разные типы упражнений.

В курсе "Алгоритмы в биоинформатике" большую часть упражнений составляли dataset упражнения. Этот тип упражнений оказался наиболее удобен, так как позволяет делать задачи про обработку больших объёмов данных, что характерно для биоинформатики, использую при этом любой язык программирования.

В курсе "Алгоритмы и структуры данных" большую часть упражнений составляли code и free answer упражнения. Code упражнения оказались удобны, так как позволяют ограничить решения по времени и памяти, что необходимо для проверки эффективности реализации алгоритмов. Free answer упражнения использовались для проверки теоретических задач.

4.3. Пример использования фреймворка сторонним разработчиком

С использованием API сторонним разработчиком был создан новый тип упражнения (admin) для курса по системногу администрированию Linux.

Это упражнение использует сторонний сервис, позволяющий конфигурировать учебный Linux-сервер с использованием командной строки.

Задача упражнения — выполнить настройку определённого ПО на сервере. Для проверки выполнение задания надо определить набор тестов, который должен пройти Linux-сервер после конфигурации.

Модуль упражнения admin общается со сторонним сервером по протоколу HTTP. Интересно, что при разработки API модулей упражнений возможность коммуникации со сторонними сервисами не рассматривалась, тем не менее API оказалось достаточно удобным и для этого случая.

Таким образом с помощью фреймворка можно создавать самые разнообразные типы упражнений.

5. Изолированное исполнение кода упражнений

Для многих важных типов упражнений необходимо исполнять потенциально не безопастный код. Из упражнений, реализованных в Stepic, такими являются следующие.

Dataset Генерация входных данных и проверка ответа происходит при помощи кода на языке Python, который вводит автор упражнения.

Code Также как и в предыдущем случае необходимо выполнять Python код автора упражнения. Но для проверки ответа также необходимо запустить код студента, который может быть не только на Python, но и на другом языке программирования, например на Java или Haskell.

Math В этом упражнении производится сравнение математических формул при помощи библиотеки SymPy. Библиотека SymPy может быть не безопасна, так как она использует eval.

String Это упражнение позволяет задавать для проверки регулярное выражение. Во многих современных языках программирования проверка строки на соответствие регулярному выражению занимает экспоненциальное время, что может привести к DOS атаке.

Таким образом возникает необходимость обеспечить изолированное и ограниченное по ресурсам исполнение кода. Код может быть написан на языке Python, или на каком-нибудь компилируемом или интерпретируемом языке программирования.

5.1. Изоляция

За основу подсистемы безопасного исполнения года был взят CodeJail. CodeJail основывается на AppArmor — безопасном[1] и удобном для использования[5] средстве мандатного управления доступом. Также CodeJail предоставляет интерфейс для использования из Python.

Использовать CodeJail без модификаций не удалось, так как он не удовлетворяет следующим требованиям.

- Совместимость с Python 3. CodeJail написан на языке Python 2 и не работает с версией языка, используемой в Stepic.
- Поддержка компелируемых языков программирования. CodeJail позволяет запустить код пользователя при помощи любого интерпретатора, однако не позволяет его предварительно скомпилировать.

количество процессов	время выполнения
1	120 с
2	30 с
4	23 с
8	1 с

Таблица 2: Эффективность масштабирования

- Наличие сообщений о превышении допустимых ограничений на ресурсы. Если программа завершается из-за превышения ресурса времени или памяти, то об этом узнать нельзя.

Для решения этих проблем CodeJail был существенно расширен. Портирован на Python 3, добавлена поддержка компилируемых языков программирования, улучшены сообщения об ошибках.

Также были созданы профили AppArmor для Java, Python, Octave и компилируемых языков программирования, таких как C++ и Haskell.

5.2. Масштабирование

Важным требованием системы проверки упражнений является масштабируемость. В каждом курсе участвует большое количество студентов, многие из которых сдают упражнения. Быстрое получение результата приводит к тому, что для сдачи упражнения совершается много попыток. Более того, нагрузка сильно не равномерная — в последний возможный день сдачи упражнения можно наблюдать большой пик попыток[7].

В качестве инструмента для масштабирования была выбрана библиотека Celery, так как она позволяет масштабироваться эффективно[3].

Эффективность решения была проверенно экспериментально. В эксперименте измерялось время создания 50 попыток для упражнения типа code в зависимости от количества потоков-обработчиков. Результаты измерений приведены в таблице .

Заключение

В результате проделанной работы все поставленные цели были достигнуты.

Был разработан фреймворк для создания новых типов упражнений в виде подключаемых модулей. Фреймворк был документирован. Для облегчения разработки был создан небольшой сервер для тестирования модулей.

С использованием фреймвока были разработыны следующие типы упражнений: choice, code, dataset, free answer, math, number, sorting, string. Эти типы упражнений были успешно использованны в курсах “Алгоритмы в биоинформатике”, “Алгоритмы и структуры данных” и других. Тип упражнения admin был создан сторонним разработчиком. Существующие модули были опубликованы на репозитории на GitHub.

На основе CodeJail была создана система безопасного исполнения кода с возможностью ограничивать и измерять затраченные ресурсы. При помощи Celery достигнуто масшабируемое и распредлённое исполнение кода упражнений.

Список литературы

[1] Bauer Mick. Paranoid Penguin: An Introduction to Novell AppArmor // Linux J. — 2006. — Vol. 2006, no. 148. — P. 13–. — http://dl.acm.org/citation.cfm?id=1149826.1149839.

[2] Clow Doug. MOOCs and the Funnel of Participation // Proceedings of the Third International Conference on Learning Analytics and Knowledge. — LAK '13. — New York, NY, USA : ACM, 2013. — P. 185–189. — http://doi.acm.org/10.1145/2460296.2460332.

[3] Lunacek M., Braden J., Hauser T. The scaling of many-task computing approaches in python on cluster supercomputers // Cluster Computing (CLUSTER), 2013 IEEE International Conference on. — 2013. — Sept. — P. 1–8.

[4] Prechelt Lutz. An Empirical Comparison of Seven Programming Languages // Computer. — 2000. — Vol. 33, no. 10. — P. 23–29. — http://dx.doi.org/10.1109/2.876288.

[5] Schreuders Z. Cliffe, McGill Tanya, Payne Christian. Empowering End Users to Confine Their Own Applications: The Results of a Usability Study Comparing SELinux, AppArmor, and FBAC-LSM // ACM Trans. Inf. Syst. Secur. — 2011. — Vol. 14, no. 2. — P. 19:1–19:28. — http://doi.acm.org/10.1145/2019599.2019604.

[6] Stepic, an educational engine. — https://stepic.org.

[7] Studying learning in the worldwide classroom: Research into edx's first mooc / Lori Breslow, David E Pritchard, Jennifer DeBoer et al. // Research & Practice in Assessment. — 2013. — Vol. 8. — P. 13–25.

[8] Документация API. — https://readthedocs.org/projects/stepic-plugins.

[9] Репозиторий с упражнениями. — https://github.com/StepicOrg/stepic-plugins.