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

Трудно представить современными мир без Интернета. Он используется в разных сферах жизни с целью повышения удобства социальныхм взаимодействий между людьми, улучшения качества услуг и расширения возможностей каждого отдельно взятого человека. Не стала исключением и сфера образования, электронная версия которой даёт пользователем возможность слушать лекции, выполнять упражнения, читать конспекты, не присутствуя физически в лекционной аудитории. Электронные образовательные ресурсы получили такое большое распространение, так как они позволяют получать качественное образование в удобном месте и в свободное время, причём с максимально комфортной для себя скоростью. Данный формат обучения получил название MOOC (Massive open online course) — массовый открытый online-курс.

Стоит обратить внимание на отличия электронного обучения от традиционного. Во-первых, из-за большей доступности и удобства количество слушателей курса может достигать нескольких десятков тысяч человек, что существенно больше, чем в обычном академическом потоке в университете, поэтому зачастую уделить индивидуальное внимание каждому студенту не предоставляется возможным [12].

Во-вторых, различие проявляется в подготовке и изменении учебного материала. В университете преподаватель может изменять курс в зависимости от успеваемости, предпочтений и количества студентов после начала курса. В электронной версии изменять лекции заметно сложнее, так как это потребует записи новых видеоуроков, подготовки новых упражнений и, вероятно, изменения формата лекций. В связи с этим, чаще всего курсы сразу готовятся высокого качества и уникальными для каждого преподавателя и предмета. Большое внимание уделяется улучшению восприятия информации: визуализация, анимация, хорошая дикция преподавателя, электронные конспекты, статьи, субтитры, видеоуроки. Все курсы очень тщательно прорабатываются, поэтому потенциально электронное образование сможет стать хорошим дополнением традиционного, так как в сети уже сейчас находится огромное количество качественного учебного материала.

В-третьих, студенты электронного курса склонны переоценивать свои силы. Кроме того, пропадает мотивация из-за отсутствия понятия «академическая неуспеваемость» как такового. Также на online-курс проще записаться, чем поступить в университет, что порождает больший спрос. Из-за отсутствия последствий непрохождения курса до конца и личного общения с преподавателем пропадает и мотивация студентов курса [2].

В-четвёртых, свободный формат позволяет студенту заниматься в удобное для него время. Студент может скачивать лекции на мобильный телефон и смотреть их в любом месте, выполнять и проверять результаты выполнения домашних заданий вне

урока, общаться со своими одногруппниками [6].

Если углубиться в историю, то можно проследить, что online-образование начало развиваться параллельно с распространением персональных компьютеров. Возможность получать знания удалённо и в новом формате стала главным фактором в популяризации MOOK. Одним из первых примеров дистанционных лекций стали уроки Уолтера Левина, профессора физики из Массачусетского Технологического Университета (MIT), в 1982 году. Он же в потом использовал для чтения лекций Skype. Затем MIT опубликовал большую часть своих учебных материалов и запустил проект OpenCourseWare (OCW). С помощью OCW стали распространяться многие материалы. Но OCW не является MOOK в современном понимании этого слова. MOOK — полноценный курс, состоящий не только из лекций, но и из системы сдачи и проверки упражнений и прочих необходимых инструментов. Первым Интернет-ресурсом, предоставляющим MOOK в полной мере, стал стартап Udacity, запущенный, в свою очередь, с целью расширения программы демократизации образования Стенфордского Университета. В начале студентам предлагалось всего два курса: написание поисковых машин и программирование беспилотных автомобилей. С тех пор формат курса мало изменился: студент может записаться на любое количество курсов до истечение сроков первого домашнего задания, по окончании курса и при условии выполнения необходимого для зачёта минимума упражнений студент получает электронный сертификат с подписью преподавателя. С 2012 года появилась возможность сдачи очного экзамена.

В настоящее время самым успешным и популярным ресурсом, предоставляющим MOOK-курсы является Coursera [3].

Стоит также заметить, что, несмотря на все плюсы, online-образованию присущ ряд специфичных проблем:

- ограниченные возможности проверки знаний,
- большая стоимость создания курса,
- отношение количества студентов, закончивших курс, к количеству студентов, записавшихся на него, как правило, очень маленькое.

Специфика online-обучения

Как было сказано выше, у электронного обучения есть ряд проблем, влияющих на его реализацию.

Из-за того, что упражнения должны проверяться автоматически, так как в случае десятков тысяч студентов вручную проверять работы не представляется возможным, теряется многообразие видов используемых упражнений. Например, вопросы со свободным ответом проверять затруднительно из-за индивидуальных особенностей

	Год основания	Количество студентов
Udacity	2011	1.6 млн.
Coursera	2012	7.1 млн.
edX	2012	2.1 млн.

Таблица 1: Год основания и число пользователей MOOK платформ на Май 2014 года

формулирования и выражения мыслей, пусть даже и формальных (в случае математических доказательств). Не малой проблемой так же является большая стоимость создания курса: запись видеолекций требует специального оборудования, большого количества времени на съёмку и обработку; подготовка электронного конспекта, инструкций для прохождения курса, написание субтитров, презентаций и т.п. Всё требует большого времени и вложений. Немаловажной частью подготовки является разработка системы упражнений.

Как мы уже обращали внимание, большая часть студентов, записавшихся на курс его не заканчивает [2], что можно считать проблемой, так как становится не до конца понятно, как с этой проблемой бороться, и стоит ли.

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 [13].

Coursera Крупнейшая на настоящий момент платформа. Некоторые курсы Coursera засчитываются в американских университетах [3].

edX Основан на Open edX — платформе с открытым исходным кодом [14].

1. Платформа Stepic

Платформа Stepic [11] это молодой проект, развивающийся в рамках компании JetBrains, разработка которой стартовала в 2013 году. На текущий момент работу над проектом ведут семь человек. Сейчас её используют 23 тысячи студентов. На платформе создано более 400 уроков, многие из которых объединены в курсы. Из завершившихся курсов стоит отметить следующие:

Алгоритмы в биоинформатике: англоязычный курс, проходивший единовременно с соответствующим курсом в проекте Coursera. В данном случае платформа Stepic использовалась как дополнения к Coursera в качестве подсистемы для упражнений. Курс состоял из большого количества текстовых материалов и задач на программирование в области алгоритмической биологии. В курсе приняли участие более 10 тысяч человек со всего мира.

Алгоритмы и структуры данных: русскоязычный курс, не доступный без приглашения в настоящий момент, использовавшийся для предварительной оценки знаний абитуриентов Computer Science Center в Санкт-Петербурге. В нём приняли участие 500 человек из России. Курс состоит из видео лекций и различных типов упражнений. Наиболее часто использовались упражнения на программирование и упражнения со свободным ответом.

1.1. Возможности платформы

Единицей учебного материала в платформе Stepic является урок. Урок — это набор упражнений и/или теории, представленный в виде слайдов. Максимальное количество слайдов, которое можно использовать в уроке 16 штук. Цель каждого урока — изучить одну тему.

Уроки можно группировать в тематические курсы. Основными задачами курса являются:

- Организация и упорядочивание уроков.

  Как правило курс состоит из блоков уроков. Блоки могут формироваться по определённой теме. Например, в курсе по алгоритмам и структурам данных удобно группировать уроки, посвящённые алгоритмам решающие схожие задачи: алгоритмы сортировок, алгоритмы поиска подстрок в строке и т.п. Так же уроки можно разбить на блоки, среднее выполнение заданий из которых примерно равно. Так можно создать блок с уроками на неделю, месяц и т.д.

- Организация групп пользователей

Неотделимо от курса можно рассматривать множество проходящих его студентов. Курс можно запускать несколько раз, и платформа должна уметь различать студентов, проходивших его в первый раз и во второй. Кроме того, различными должны быть и рейтинговые таблицы студентов, проходящих курс. Помимо хранения информации о студентах надо устанавливать временные сроки, дающиеся на выполнение заданий.

Платформа Stepic ориентирована на интеграцию и сотрудничество с другими инструментами online образования [10]. Для этого платформа реализует ряд API, такие как:

- oEmbed: необходимо для встраивания почти на любую веб-страницу.
- LTI: с помощью него есть возможность быть встроенным в Coursera.

  Так было в курсе по биоинформатике, когда наша платформа использовалась в качестве подсистемы проверки упражнений.

Отличительной особенностью платформы Stepic является разнообразие предоставляемых типов упражнений (на данный момент более 8. Для сравнения, на Coursera их около 4). В связи с этим очень важна возможность и лёгкость расширения набора доступных разновидностей упражнений.

1.2. Технологи и инструмены

Server side Серверная часть разрабатывается на языке Python 3. Основу составляет веб-фреймворк Django. При этом Stepic представляет собой rich web application, поэтому сервер только предоставляет REST API [4], при помощи Django REST framework, а вся логика представлнения описана в клиентской части.

Для обеспечения масштабируемости была использована библиотека Celery, которая позволяет распределять выполнение заданий по многим процессам, не обязательно в рамках одной машины.

Для нужд подсистемы упражнений также были использованы библиоека Sympy для символьных вычислений и AppArmor для обеспечения изоляции кода.

Client side Клиентская часть разрабатывается на языке CoffeeScript. Как уже упоминалось, именно на клиентскую часть ложится вся забота о взаимодействии с пользователем — генерация html страниц, обработка ввода, отправка запросов к серверу. Поэтому для неё был выбран client side фреймвок Ember.js.

Платформа располагается на Amazon AWS.

2. Постановка задачи

Целью данной работы является реализация системы для создания и проверки упражнений в образовательной платформе Stepic, с возможностью лёгкого добавления новых типов 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 платформе Stepic типы упражнений, часто встречающиеся 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, таких как Rosalind [8], Coursera [3], edX [14] и Udacity [13].

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 текст определённого формата. При этом в условие можно внедрять параметры упражнения, например ограничения или примеры решений для маленьких наборов данных.

Шаг 2. Пользователь нажимает на кнопку «начать решать». После этого ему 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 список вариантов.

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 балл 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 платформы Stepic.

Коллекция реализаций существующих модулей доступна на GitHub в публичном репозитории [16]. Репозиторий открыт для добавления новых модулей.

3.5. Архитектура решения

Привлекательной кажется идея реализовывать упражнения целиком на стороне клиента, 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. В-третьих, не все задачи можно решить в браузере. Поэтому для реализации упражнений была выбрана клиент-серверная архитектура, в которой клиентская часть отвечает только 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 потенциально не безопасного кода упражнений осуществляется в изолированных песочницах.

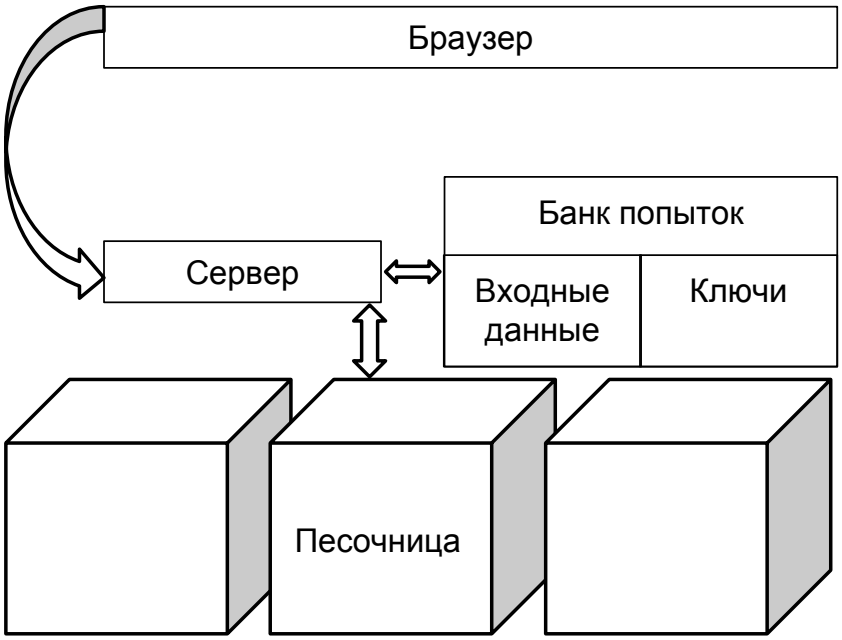

Рис. 1: Архитектура решения

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 необходи-

мости 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 высокого уровня, главное преимущество которого заключается в уменьшении затрат на разработку модулей [7].

Для создания серверной части разработчику модуля упражнения необходимо определить на Python 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.js.

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, на основе которого можно быстро начать разрабатывать свой модуль.

Документация для разработчиков [15].

4. Реализованные типы упражнений

С помощью фреймворка в Stepic были разработаны следующие типы упражнений:

Choice — выбор ответа из списка предложенных вариантов.

Code — написание программы для эффективного решения задачи.

Dataset — машинная обработка входных данных.

Free Answer — свободный ответ.

Math — проверка символьной формулы.

Number — числовой ответ.

Sorting — упорядочивание предложенного набора.

String — проверка ответа при помощи регулярного выражения.

4.1. Примеры конкретных упражнений

Рассмотрим возможности фреймворка на примере некоторых упражнений.

Choice просит студента выбрать среди предложенных вариантов ответа правильные. У этого упражнения есть много опций: размер выборки, её случайность, количество правильных ответов в выборке. Это наиболее часто встречающийся в различных платформах тип упражнения 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 подходит для некоторых упражнений на программирование. Он хорош тем, что позволяет использовать любой инструмент для решения. Но у этого типа упражнения 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 среду разработки на своём компьютере.

Code: дополняет Dataset. В этом упражнении студент должен предоставить в качестве решения код на каком-то конкретном языке программирования, который 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, Haskell).

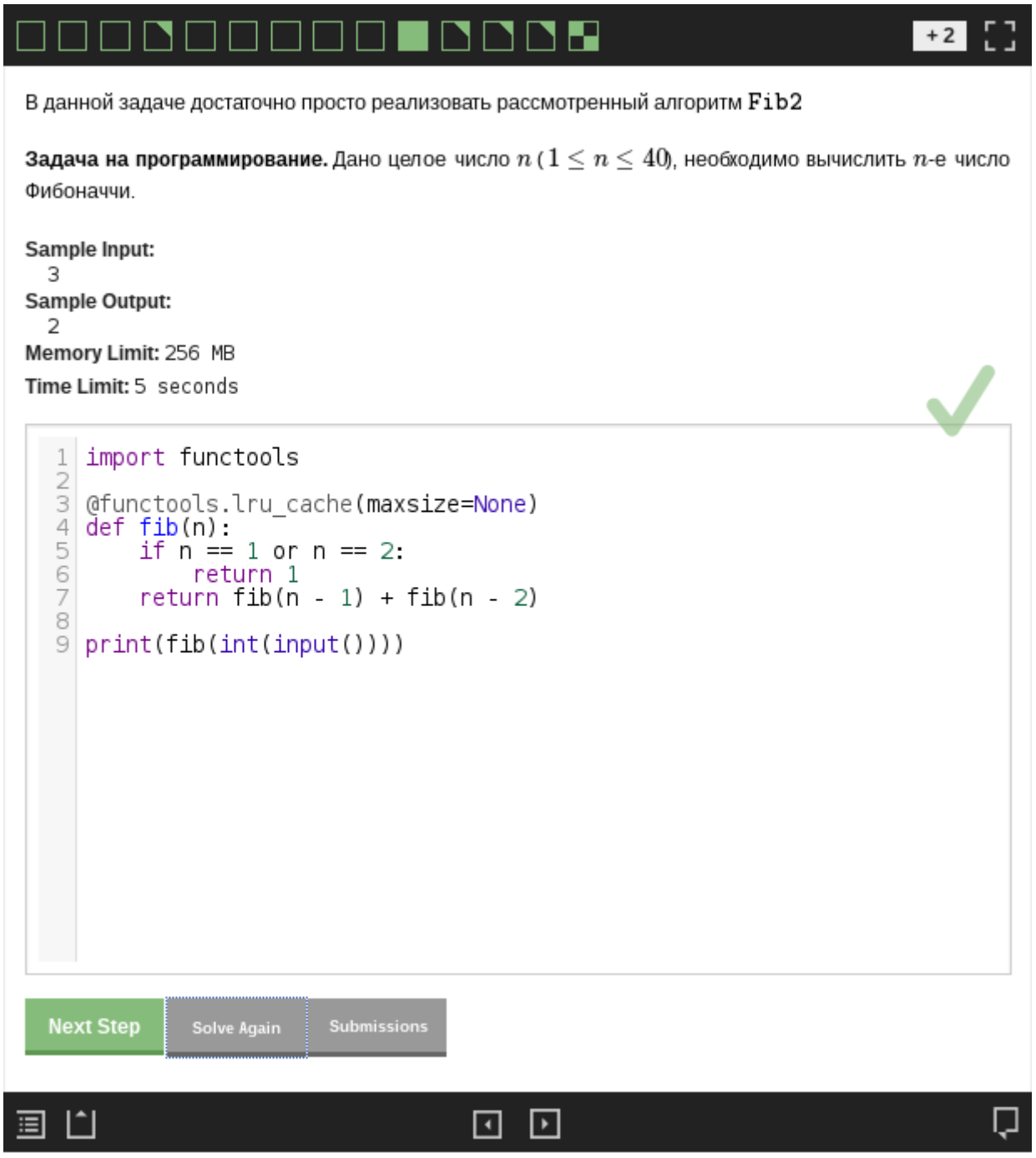

Рис. 2: Пример упражнения типа Code

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 упражнения типа Dataset. Этот тип упражнений оказался наиболее удобен, так как позволяет создавать в качестве заданий задачи на обработку больших объёмов данных, что характерно для биоинформатики, используя при этом любой язык программирования.

В курсе «Алгоритмы и структуры данных» большую часть упражнений составляли упражнения типа Code и Free Answer. Упражнения типа Code оказались удобны, так как позволяют строго ограничить решения по времени и памяти, что необходимо для проверки эффективности реализации алгоритмов. Упражнения типа Free Answer использовались для теоретических задач, где свободный ответ содержал доказательства или идею алгоритма.

4.3. Пример использования фреймворка сторонним разработчиком

С использованием API сторонним разработчиком был создан новый тип упражнения Admin для курса по системному администрированию Linux.

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

Задача упражнения — выполнить настройку определённого ПО на сервере. Для проверки 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 разработке 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 другом 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 [9] 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 компилируемых языков программирования. CodeJail позволяет запустить код пользователя при помощи любого интерпретатора, однако не позволяет его предварительно скомпилировать.

- 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 [12].

В качестве инструмента для масштабирования была выбрана реализация распределённой очереди задач Celery, так как она позволяет масштабироваться эффективно [5].

Эффективность решения была проверенно экспериментально. В эксперименте измерялось время создания 60 попыток для упражнения типа Code в зависимости от количества потоков-обработчиков. Результаты измерений приведены в таблице 2.

количество процессов	время выполнения
1	37 с
2	20 с
4	8 с

Таблица 2: Эффективность масштабирования

Заключение

В результате проделанной работы все поставленные цели были достигнуты.

Был разработан фреймворк для создания новых типов упражнений в виде подключаемых модулей. В состав фреймворка также входят шаблон для начала разработки, набор примеров, демонстрирующих возможности API, сервер для локального запуска и тестирования упражнений. С подробной документацией фреймворка можно ознакомится в [15].

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. Специальный тип упражнения Admin для курса по системному администрированию Linux был создан сторонним разработчиком. Был опубликован репозиторий со всеми существующими модулями [16].

На основе CodeJail и AppArmor была создана система безопасного исполнения кода с возможностью ограничивать и измерять затраченные ресурсы. Эта система была использована при создании упражнений Code, Dataset, Math и String. 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] Coursera. — https://www.coursera.org/about/.

[4] Fielding Roy T, Taylor Richard N. Principled design of the modern Web architecture // ACM Transactions on Internet Technology (TOIT). — 2002. — Vol. 2, no. 2. — P. 115–150.

[5] 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.

[6] Mak Sui, Williams Roy, Mackness Jenny. Blogs and forums as communication and learning tools in a MOOC // Networked Learing Conference / University of Lancaster. — 2010. — P. 275–285.

[7] 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.

[8] Rosalind. — http://rosalind.info/about/.

[9] 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.

[10] Service-oriented e-learning platforms: From monolithic systems to flexible services / Declan Dagger, Alexander O'Connor, Seamus Lawless et al. // Internet Computing, IEEE. — 2007. — Vol. 11, no. 3. — P. 28–35.

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

[12] 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.

[13] Udacity. — https://www.udacity.com/.

[14] edX. — https://www.edx.org.

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

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