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:

Научный руководитель:

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 программ, осуществляющих такое улучшение, получил название электронного обучения[http://www.eurodl.org/ma

Особенности электронного обучения

Электронное обучение отличается некоторыми особенностями.

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

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.

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

online образованию присущи и некоторые специфичные проблемы. Среди них можно выделить следующие.

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

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

- Drop out. Для online образования характерен значительно меньший процент заканчивающих курс.

Стоит заметить, что в электронном образовании лекционный материал может быть представлен очень хорошо, зачастую лучше, чем в похожем классическом обучении. 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. Но при этом практическая часть обучения – решение задач и выполнение упражнений – в электронном обучении реализуется значительно хуже, так как нет возможности проверки заданий студентов учителем.

Существующие платформы

*** Udacity 2011 год 1.6 млн *** Coursera 2012 год 7.1 млн *** edX 2012 год 2.1 млн MOOC.org

1. Платформа Stepic

1.1. Статус проекта

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

1.2. Использование

На текущий момент Stepic используют 23 тысячи студентов. На платформе создано 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, он состоял из большого количества текстовых материалов и упражнений на програмирование из области алгоритмической биологии. В этом курсе приняли участие более 10 тысяч человек со всего мира.

"Алгоритмы и Структуры Данных"

Это 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 на русском языке. Для курса использовались разные типы упражнений, но наиболее часто упражнения на программирование.

1.3. Возможности платформы

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

Единицей учебного материала на Stepic является урок – набор упражнений и/или теории представленный в виде сладов, общим количеством не превосходящий 16 штук. Уроки можно объединять в курсы.

Курс позволяет создавать секции уроков, назначать разным упражнениям разные стоимости, создавать коллективы студентов и преподавателей, устанавливать предельные сроки сдачи упражнений.

Теория может быть представленна в виде html слайдов, или в виде видео лекций.

Необходимость расширять возможности автоматически проверяемых 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, flask тоже зачем-то используется.

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 проверить на практике его удобство.
- Реализовать с помощью разработанного API 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. API для создания новых типов упражнений

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.

Входные данные 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.

3.1. Вызовы API

В результате анализа различных типов упражнений был выявлен следующий общий набор операций, необходимых для создания 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.

3.2. Система Модулей

Для подсистемы упражнений крайне важна модульность, так как упражнения для курса могут создаваться авторами курса, которые хорошо разбираются в своей предметной области, но при этом 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. Чтобы добавить новый модуль, автору необходимо сделать pull request в этот репозиторий.

3.3. Архитектура Решения

Упражнения работают по модели клиент-сервер.

Серверная часть обеспечивает создание входных данных и проверку ответов. Теоретически, её можно было бы осуществлять и на клиенте, но в таком случае было бы просто подделать сдачу упражнения.

Клиентская часть работает в браузере и обеспечивает богатое взаимодействие с пользователем. Клиент общается с серевером при помощи ajax запросов и упражнения работают без перезагрузги страницы.

3.4. Детали Реализации

Все данные упражнения сохраняются в формате JSON. Коммуникация между клиентом и сервером также происходит в формате JSON. Для описания схемы JSON сообщений реализован eDSL в Python.

Серверная часть модуля должна быть написана на Python. Клиентская часть может быть написана на Javascript или CoffeScript, с необязательным использованием ember.

3.5. Сервер Для Разработки

Для упрощения разработки модулей упражнений был написан сервер, который позволяет запускать упражнения без Stepic. Сервер автоматически перезагружается при изменении исходного кода упражнений.

4. Реализованные типы упражнений

С помощью API в Stepic были разработны следующие типы упражнений:

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

Примеры Конкретных Упражнений

Рассмотрим некоторые из них более подробно.

Choice просит студента выбрать среди предложенных вариантов ответа правильные. У этого квиза есть много опций – размер выборки, её рандомизация, 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 проконтроллировать 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 проврки эффективности решения, однако необходимо решать задачу на одном из поддерживаемых языков программирования(Java, Python, C++, Octave).

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.

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 – в этом упражнении входные данные генерируются при помощи кода на питоне, который вводит пользователь.

code – в этом типе упражнений ответом является код программы на Python, C++, java, haskell или octave.

math – в этом упржнении необходимо вычислять произвольное математическое выражение, также записанное в синтаксисе Python

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

5.1. Изоляция

Для изоляции кода был существенно расширен codejail. Добавлена поддержка различных языков программирования, улучшены сообщения об ошибках, созданы профили apparmor для java, octave, haskell и C++.

5.2. Масштабирования

celery, nuff said

Заключение

В результате проделанной работы все поставленные цели были достигнуты.

Были разработаны 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.

Был разработан API для создания новых типов упражнений в виде подключаемых модулей. API был документирован. Для облегчения разработки бы создан небольшой сервер для тестирования модулей. С использованием API был создан первый сторонний модуль упражнения. Существующие модули были опубликованы на репозитории на GitHub.

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