Список проектов

1. Робот, проверяющий формат отчётов

Отчеты студентами присылаются согласно этим правилам:
http://se.moevm.info/doku.php/start:report_submis

Необходимо реализовать email-робота (smtp-клиента) для проверки соответствия формату, учета времени отправки, фильтрации, хранения и ведения статистики входящих e-mail.

Ожидаемый результат: smtp-клиент на языке Python.

2. Генератор отчета по курсовой работе по содержимому репозитория на Github.com

Требуется реализовать инструмент, который позволяет:

- сформировать из вики-страниц и заранее известного шаблона отчет в формате doc
- сформировать раздел Приложение, который включает разделы «замечания по ходу выполнения работы» - дискуссия из пулл-реквестов в репозитории и «Исходный код».

Ожидаемый результат: консольный скрипт на языке Python.

3. Тренажер публичных выступлений

Цель: сделать веб-приложение тренажер, позволяющий докладчику объективно измерить

- скорость речи (сколько слов в секунду он произносит)
- четкость речи
- скорость доклада (расход времени/слов на каждый слайд/ среднее время/слова на каждый слайд)
- проверить укладываемость во временное ограничение
- измерить те же самые параметры в контексте ответов на вопросы
- сопоставить распознанные слова и текст речи или тезисы для выявления неосвященных тем*

Сценарий использования:

- пользователь открывает приложение
- пользователь загружает презентацию и указывает временное ограничение (количество минут на доклад)
- пользователь нажимает кнопку «Начать тренировку»
- на экране отображается презентация, обратный отсчет времени, номер слайда/общее количество слайдов, график (стрелочный индикатор) количества слов в минуту , кнопки переключения слайдов
- пользователь осуществляет доклад
- если темп речи превышает некоторый, заранее заданный порог, то график/индикатор

окрашивается красным
- по окончанию доклада пользователь нажимает кнопку «Доклад окончен»
- на экране отображается статистика выступления - общая и по отдельным слайдам

Ожидаемый результат: веб-приложение на языке Python/Turbogears + MongoDb.

4. Анализатор пулл-реквестов

Задача: доработка проекта для возможности реального использования. Текущие проблемы описаны в issues.

https://github.com/moevm/rePullet/issues

5. Информационная система кафедры: учёт студентов

Требуется расширить функционал компонентов информационной системы кафедры, которые обеспечивают:

- Управление студенческими данными
- Импорт/экспорт данных
- Рейтинги по предметам, курсовые, практики.
- Генерация отчетов.

Ожидаемый результат: веб-приложение на языке Python2.7/Django + MongoDb.

6. Информационная система кафедры: научная работа

Требуется расширить функционал компонентов информационной системы кафедры, которые обеспечивают возможность ввода/обработки/хранения/предоставления информации о:

- публикациях/научных трудах (группировка по годам/людям)
- участии (достижениях) в конференциях/конкурсах/НИОКР и т.д.
- численности аспирантов, докторантов, соискателей и стажеров
- Также приложение должно обеспечивать разные уровни доступа к информации для различных пользователей.

Ожидаемый результат: веб-приложение на языке Python2.7/Django + MongoDb.

7. Информационная система кафедры: индивидуальный план преподавателя

Требуется расширить функционал компонентов информационной системы кафедры, которые обеспечивают управление индивидуальными планами преподавателей:

- Планирование учебной нагрузки
- Переподготовка и повышение квалификации
- Участие в мероприятиях, конференциях,

- Руководство НИР
- Генерация отчетов
- ... и другие параметры индивидуального плана

Ожидаемый результат: веб-приложение на языке Python2.7/Django + MongoDb.

8. Автоматизация проверки задач для курса по программированию

Требуется разработать набор задач для автоматической проверки решений студентов для курса по программированию на платформе Stepik.

Данный курс рассчитан на студентов 1го года бакалавриата кафедры МОЭВМ, знающих основы Linux. Курс содержит практические задачи и лабораторные работы на языке Си.

Подробнее о типах задач в Stepik здесь

Ожидаемый результат: набор готовых проверяющих задач в курсе по программированию (Python2/Python3 + bash) + набор задач-тестов для каждой проверющей задачи в репозитории (язык C).

9. Генератор задач для онлайн-курса по GDB/Valgrind

Требуется автоматизировать проверку интерактивных задач на платформе Stepik для курса по GDB/Valgrind.

Цель: разработать интерактивные задачи на отладку и профилирование C программ по следующим темам:

1. сегфолтов
2. неправильных вызовов
3. ошибок преобразования типов
4. поиск ошибок параллельного исполнения
5. обратная инженерия
6. проблемы с производительностью

Подзадача: освоить инструменты обфускации / замусоривания исходного кода лишними инструкциями.

Задачи GDB:

- Сегфолты
  - Дан бинарный файл программы, собранный с отладочными символами. Программа падает при запуске с сегфолтом. Необходимо найти имя переменной и функции, где это происходит.
  - Тоже что и выше, только программа запущена в режиме «демона» (висит в памяти)(проверка на владение gdb attach).
- Исследование работоспособных программ:
  - Определение места в коде, где допущена ошибка.
    - Дан бинарный файл программы, собранный с отладочными символами. Исходный код выглядит достаточно непонятно (обфускатор). Программа

реализует (варианты):
- вычисление суммы чисел от 1 до N (либо какой-то похожий простой алгоритм) и делает это неправильно.
- запись данных в файл, но почему-то данные в файл попадают не все.
- сортировка строк, которая работает не правильно.

▪ Задача: с помощью gdb установить, где ошибка (номер строки/имя функции).

◦ Поиск значения в памяти:
▪ Дан бинарный файл программы, собранный с отладочными символами. Исходный код выглядит достаточно непонятно (обфускатор). Программа реализует итеративное вычисление какого-нибудь значения (строкового или числового):
- контрольная сумма
- шифрование
- приближенное значение мат. функции

▪ Задача: с помощью gdb установить, значение функции на итерации N.

Задачи valgrind:

- Утечка памяти. Дан бинарный файл программы, в котором допущена утечка памяти. Задача: определить переменную, через которую утекает память.
- Производительность. Дан бинарный файл программы, в котором очень много разных функций и их вызовов. Задача: определить самую часто вызываемую функцию.

Ожидаемый результат: набор автоматически проверяемых заданий, интегрированных в Stepik.

10. Генераторы задач для онлайн-курса по нереляционным БД

Цель: разработка системы автоматической проверки лабораторных работ для курса «Введение в нереляционные БД» на платформе Stepik.

Задачи:

1. Изучение операций в нереляционных БД;
2. Разработка скриптов автоматизации для генерации условий и проверки лабораторных;
3. Разработка эталонных и ошибочных решений;
4. Интеграция наработок в stepic.org;

Требование:

1. Python, Linux

Тематики:

1. MongoDB
    1. полнотекстовый поиск
    2. геозапросы
    3. GridFS
    4. MapReduce
2. ArangoDb
    1. Создание объектов БД всех видов

    2. Графовые операции
3. Cassandra
    1. установка и настройка
    2. создание БД
    3. запросы на чтение/запись
    4. BigTable специфика
4. OpenLink Virtuozo
    1. SPARQL запросы
    2. RDF
5. ExistDB
6. ...

Ожидаемый результат: набор автоматически проверяемых заданий, интегрированных в Stepik.

11. Автоматизация сборки мобильных приложений

Цель: разработать инструмент автоматизации сборки и тестирования мобильных приложений для набора репозиториев.

Задачи:

1. запуск сборки и тестирования в jenkins
2. генерация итоговых отчетов с результатами сборки и/или тестирования по набору репозиториев
3. создание apk и их публикация

Ожидаемый результат: инструмент, позволяющий автоматически оценивать качество N репозиториев с исходниками мобильных приложений путем их сборки/тестирования/установки; инструкции по развертыванию и настройке инструмента.

12. Автоматизация проверки задач для онлайн-курса "SSH-tricks"

Цель: разработка интерактивных задач для курса на платформе Stepik “SSH-tricks”.

Варианты задач:

- Интерактивные задачи (генератор среды + скрипты проверки):
    - выполняем команды на удаленном сервере
    - генерируем ключи
    - настраиваем авторизацию без пароля (authorized keys)
    - настраиваем авторизацию по ключу (-i)
    - учимся делать туннели
    - scp
    - sshfs

Каждая задача состоит минимум из двух скриптов - генератора среды и скрипта проверки.

Ожидаемый результат: набор автоматически проверяемых заданий, интегрированных в Stepik.