Летняя практика 2017

Geo2Tag

Цель: расширение возможностей использования для LBS-платформы Geo2Tag.

Задачи:

1. Завершение реализации REST интерфейса доступа к карте.
2. Автоматизация тестирования REST интерфейса доступа к карте.
3. Любые задачи из backlog.

Ожидаемый результат:

1. Набор функций и тестов для платформы Geo2Tag.

Навыки и знания:

1. python 2
2. html, css, js
3. MongoDB

Stepik

Цель: автоматизация проверки задач преподавателя практики по курсу «Программирование» на образовательной платформе Stepik.

Задачи:

Статистика

Консольное приложение на языке python, в котором на основе .csv файла с результатами прохождения модуля / курса студентов строится статистика.

Предоставление обратной связи студентам

Консольное приложение на языке python, в котором на основе решения студента, функции-генератора тестовых данных, функции решения и проверки получаем место ошибки в коде студента.

Получение информации о коде студента

Консольное приложение на языке python, в котором происходит проверка кода студента на выполнение некоторых правил задачи:

1. наличие определенных функций/методов, инструментов языка.
2. невмешательство в изначальный код, который был дан преподавателем.

Проверка на жульничество

Консольное приложение на языке python, в которое позволяет получить сводную таблицу о тех, кто жульничает.

Введение в ПИ

Цель: рефакторинг и завершение проектов студентов, которое участвовали в курсе «Введение в ПИ» осенью 16го года.

Список проектов

Автоматизация проверки лабораторных для курса "Введение в нереляционные БД"

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

Задачи:

1. Изучение простых операций в MongoDb.
2. Разработка сценариев автоматизации для проверки лабораторных, связанных с программированием PyMongo.
3. Разработка эталонных и ошибочных решений лабораторных работ.
4. Интеграция наработок в stepik.org.

Требования:

1. Python, Linux
2. MongoDB на самом базовом уровне

Результат: набор автоматически проверяемых заданий для студентов, изучающих работу в MongoDB через Python-интерфейсы.

- https://bitbucket.org/mark_zaslavskiy/nosql_inroduction/overview
- http://se.moevm.info/doku.php/staff:courses:no_sql_introduction

Сервис анализа пулл-реквестов Pullet

Цель: доработка и внедрение сервиса (https://github.com/moevm/rePullet) на кафедре.

Задачи:

1. Интеграция с веб-сервером Apache;
2. Повышение удобства использования веб-интерфейса.
3. Интеграция сервиса с Github API.
4. Автоматизация установки сервиса и настройки программной среды.

Требования:

1. Python, Flask, JS

Результат: веб-сервис, установленный на кафедральном сервере.

Веб-интерфейс для инструмента версионируемой загрузки курсов на Stepik

Цель: создание веб-интерфейса к инструменту командной строки https://github.com/OSLL/stepic_uploader , позволяющего автоматизировать создание уроков и курсов в рамках платформы Stepik. Реализуемые сценарии использования:

1. Создание урока-контрольной по существующему набору вопросов с выбором ответа.
2. Загрузка и выгрузка курса в машиночитаемом виде в систему контроля версий.

Задачи:

1. Создание пользовательского интерфейса.
2. Подключение авторизации с использованием данных Github и Stepik по протоколу OAuth2.
3. Автоматизация установки сервиса и настройки программной среды.

Требования:

1. Python, Flask, JS

Результат: веб-сервис загрузки и выгрузки для курсов на stepik.org .

Система проверки студенческих решений для онлайн-курсов "Основы программирования для Linux / Программирование в ядре Linux"

Цель: разработка системы, осуществляющей виртуализированную проверку студенческих решений.

Задачи:

1. Создание и отладка сценариев проверки отдельных заданий.
2. Маршрутизация HTTP-запросов к системе и горизонтальное масштабирование экземпляров системы.
3. Архитектурное разделение проверяющей системы и сценариев проверки отдельных заданий.

4. Разработка заданий для изучения инструментов отладки и профилирования (gdb, valgrind, callgrind).

Требования:

1. Основные технологии: Ruby, C.
2. Дополнительно: Vagrant, Docker, Libvirt, проргаммирование ядра Linux, Bash.

Результат: изменения, заинтегрированные в основную ветку репозитория проекта, и развернутые в курсах на Stepik.

Веб-сервис сбора и анализа статистики курса "Основы программирования в Linux"

Цель: доработка и реализация новых функций веб-сервиса сбора статистики, использующем данные журнала проверяющей системы курса «Основы программирования в Linux».

Задачи:

1. Изучение принципов статистического анализа с помощью Python и MongoDb.
2. Полнотекстовый поиск с помощью интерфейсов MongoDb.
3. Сбор и вычисление статистики курса (самые сложные задачи, скорость решения отдельных задач, наиболее частые ошибки).
4. Оперативная загрузка данных журнала проверяющей системы курса «Основы программирования в Linux».
5. Реализация графического представления статистических показателей.
6. Создание и выгрузка отчетов.

Требования:

1. Python, Django
2. JS библиотеки для построения графиков и диаграмм.

Результат: веб-сервис, который позволяет вести наблюдение за статистическими показателями прохождения курса и отслеживать появление определенных событий в журнале работы проверяющей системы.

Система автоматической проверки наиболее частых ошибок в формальных текстах

Цель: создать веб-сервис анализа формальных текстов (научные статьи, курсовые работы, пояснительные записки, отчеты) на соответствие критериям, определяемых пользователями сервиса. Критерии представляют собой типичные алгоритмически-верифицируемые ошибки, возникающие при подготовке документов. Примеры критериев:

1. Личные предложения и личные формы глаголов.
2. Отсутствие ссылок или неверные ссылки на элементы списка литературы, изображения, таблицы.

3. Повторы слов в пределах двух предложений.
4. Стоп-слова:
    1. жаргонизмы: скачать, пост, либа, тул;
    2. личные местоимения.

Задачи:

1. Разбор и извлечение текста из файлов формата doc(x), ppt(x), odt, pdf.
2. Авторизация пользователей с помощью протокола OAuth2.
3. Хранение пользовательских критериев в стандартизированном виде.
4. Асинхронная проверка выполнения больших наборов критериев.
5. Создание веб-интерфейса.

Требования:

1. Python, MongoDb
2. Представление о формате XML.

Результат:

1. Приложение командной строки для анализа документов на ошибки.
2. Веб-сервис, реализующий интерфейс пользователя к приложению, функции авторизации и хранения критериев.