Направления для НИР (магистры)

Веб-интерфейс генерации мобильных приложений outdoor-квестов

Цель: создание веб-приложения для генерации пересонализируемых мобильных приложений outdoor-квестов. Содержимое приложений генерировать не требуется (есть шаблон).

Задачи:

- Анализ аналогичных сервисов.
- Разработка веб-интерфейса, позволяющего пройти все этапы персонализации мобильного приложения, в том числе
  - настройку целевых платформ,
  - загрузку ключей для подписи apk (или их создание и хранение в сервисе)
  - сборку,
  - подключение пользовательских тем.
- Виртуализация и автоматизация сборки мобильных приложений.

Требования:

- Python + TurboGears.
- Базовые знания разработки для Android.

Результат: веб-сервис с помощью которого, человек, имеющий нулевой или близкий к тому опыт, сможет сгенерировать персонализированное приложение.

Инструмент разработки и модерирования outdoor-квестов

Цель: разработать веб-сервис создания сценариев для outdoor-квестов.

Задачи:

- Исследовать примерные аналоги - приложения для составления маршрутов на карте, визуализаторы gps-треков и т.д.
- Разработать веб-интерфейс, который позволит
  - осуществлять CRUD для квестов;
  - настраивать генератор квестов и создавать квесты с его помощью;
  - визуализировать квесты на карте, проводить грубый анализ (примерная длительность прохождения, протяженность, потенциальная сложность …);
  - предоставлять интерфейс модерировния квестов;
  - предоставлять интерфейс для ручного создания/редактирования квестов;
  - выбирать источники данных для подбора точек.
- Подключиться к интерфейсам генератора, передавать ему правила генерации и получать созданные квесты.

Требования:

- Python + TurboGears.
- Bootstrap

- Leaflet.js

Результат: веб-приложение, позволяющее не знакомому с технической частью пользователю быстро создавать свои квесты либо модерировать и править квесты, созданные генератором.

Темы ВКР бакалавров

Автоматическая система проверки задач для МООС "Мобильная разработка для Android на Kotlin"

Цель: создание набора автоматически проверяемых лабораторных по мобильной разработке на Kotlin, интегрированных в stepik.org.

Задачи:

- обзор онлайн-курсов для изучения Java+Android, Kotlin, Kotlin+Android и сравнение предлагаемых задач;
- изучение интерфейса командной строки для сборки, запуска и тестирования мобильных приложений в эмуляторе Android, автоматизация типичных сценариев использования;
- подготовка образа виртуальной машины для задач курса, содержащего необходимые инструменты и среду;
- разработка неинтерактивных задач:
  - описание условий задач,
  - создание эталонных решений (правильных и содержащих различные ошибки),
  - создание скриптов проверки задач (на Bash и Espresso)
- интеграция наработок на Stepik.org;
- исследование производительности полученного решения.

Требования:

- опыт Android разработки и интерес к данной предметной области;
- представление о работе виртуальных машин;
- начальные знания Bash.

Результат: набор автоматически провереемых задач с описаниями, подключенных к курсу на Stepik.org.

Система автоматической проверки наиболее частых формальных ошибок в научных статьях/отчетах

Цель: разработать настраиваемый статический анализатор для формальных текстов - научно-технических статей, отчетов и дипломов. Анализ происходит по набору наиболее частых ошибок, которые при этом являются машинно-проверяемыми, например:

1. личные предложения, формы глаголов и местоимения;
2. отсутствие ссылок или битые ссылки на элементы списка литературы/рисунки/таблицы;
3. отсутствие упоминания ключевых слов в тексте статьи;

4. повторы слов в пределах двух предложений;
5. “телеграфность” - повторение начальных слов абзацев (“Было принято решение”);
6. стоп-слова:
    1. жаргонизмы (использовать список): скачать, пост, либа, тул;
    2. там, тут, здесь.

Задачи:

1. изучение как ближних (например spellcheck), так и дальних аналогов (lint'ы и статические анализаторы для языков программирования);
2. формулировка сценариев использования и создание макета UI;
3. разработка модели данных для правил;
4. парсинг docx/ppt/pdf;
5. полнотекстовый поиск;
6. хранение и эффективная интерпретация правил;
7. создание веб-интерфейса для инструмента;
8. аннотирование проверяемого документа (например добавление комментариев прямо в doc(x));

Требования:

- начальный опыт написания формальных текстов и примерное понимание, почему нужно выполнять большую часть правил, описанных в цели работы;
- опыт обработки машино-читаемых документов (HTML, XML, JSON, CSV ….);
- начальные знания Python;
- начальные знания веб-технологий.

Результат: веб-сервис, который производит анализ структуры и содержимого загруженных документов.