Rīgas 95.vidusskola

Informācijas tehnoloģijas ieviešanu visās ārpusklases aktivitātes

Zinātniski pētnieciskais darbs informatikas sekcijā

Darba autors: Sergejs Bondarenko
Darba vadītāja: Lukaševiča Larisa

Rīga, 2020

Anotācija

Informācijas tehnoloģijas ieviešanu visās ārpusklases aktivitātes. Sergejs Bondarenko, darba vadītāja Larisa Lukaševiča, informatikas skolotāja.

Darbā analizēts – perspektīvas informācijas tehnoloģiju izmantošanu visās ārpusklases aktivitātes. Tehnoloģijas kas tiek izmantoti lai izveidot reālu maketu. Ieviešanas sarežģītība informācijas tehnoloģijas.

Darba mērķis - izpētīt perspektīvas informācijas tehnoloģiju ieviešanu visā ārpusklases aktivitātes.

Darba uzdevumi:
1. Создать сайт для мероприятия.
2.

Darba satura rādītājs

Ievads

В работе исследуются перспективы, методы и сложности внедрения информационных технологий во внеклассную деятельность школьного самоуправления Рижской 95 средней школы. На примере одного из мероприятий устраиваемых школьным самоуправлением будет показано каким образом возможно внедрить информационные технологии во внеурочную деятельность. Также в работе предоставлены технологии и описан процесс создания реального макета сайта используемого для организации и подготовки исследуемого мероприятия. Гипотеза выдвигаемая по началу исследования такова - возможности предостовляемые информационными технологиями оправдают сложность своего использования в сфере влеклассной деятельности.

Обзор литературы

В ходе исследовательской работы была использована техническая литература касаемая в основном создания веб-сайтов на платформе фреймворка Django, написанного на языке программирования Python версии 3.6.4. Также была взята информация из учебников и самоучителей по языку програмированию Python, касаемая тонкостей языка, к примеру таких как, генераторы, структуры данных, паралельные функции, конкурентность, многопоточность, также особенности синтаксиса языка об объекто-ориентированной его состовляющей. В то же время была использована информация и литература о шифровке данных, кодирования данных, геолокации, передачи данных через глобальную сеть Интернет, о протоколах передачи информации посредством протоколов HTTP&HTTPS, FTP, SSH и многих других протоколов. Не обошлось без научных статей об использовании распределенной системы управления версиями Git и платформы GitHub и в том числе о бронировании хостинга(pythonanywhere.com), размещении сайта на платформе хостинга, конфигурации сайта на платформе хостинга, покупки и регистрации доменного имени и настройки DNS в рамках самой платформы хостинга.

Методы

Для внедрения информационных технологий во внеклассную деятельность школьного самоуправления было выбрано мероприятие для 10-классников называемое “Посвящение 10-классников”. Суть мероприятия посвятить новоиспеченных старшеклассников в жизнь старшей школы. Организация и план мероприятия были следующими.

Во время урока два человека из самоуправления под видом военных заходили в класс к 10-класссам(всего два 10-ых класса), приказывали всем немедленно выйти из класса и проследовать за ними на стадион. На стадионе их ждала группа переодетых в военных людей. По прибытию всех 10-классников им объяснялось, что они попали в зону химической атаки и были заражены вирусом Z и что им необходимо найти антидот. Тут вступает в роль самая ключевая состовляющая всего мероприятия. За 2 недели до мероприятия началось разработка сайта фото-квеста. Его суть была следующей. Команда, которых всего было 6, должна искать места, коих всего было 6 для каждой команды, по фотографии данной на сайте и по прибытию на место им необходимо было сделать фотографию самого места и селфи на фоне этого места, потом отправить на сайт, а на самом сайте модераторы, в лице участников самоуправления, проверят валидность отправленных им фотографий. Если фотографии валидны, а то есть отправленные участниками фотографии в ответ соответствуют фотографии места необходимого найти участникам квеста, то модераторы отправляют игрокам часть ключа, необходимого им после прохождения первой части квеста и фотографию следующего места. После того как все участники найдут все 6 мест и получат все 6 частей ключа, которые зашифрованы шифром Цезаря, им необходимо будет расшифровать ключ, который на самом деле является адрессом места последнего испытания, прибыть по этому адрессу на место и выполнить задание которое даст им уже в живую находящийся там человек из самоуправления. После выполнения последнего испытания, их отправят обратно на стадион, где их также будут ждать участники самоуправления и дадут им антидот, коим условно являлся сироп от кашля.

Техническая сторона квеста. Сайт фото-квест. Сайт написан на языке Python с помощью фреймврока Django. Фреймворк Django придерживается архитектуры MVT – Model View Template. В качестве модели, в узком понимании это база данных, используется SQLite. Выбор был обусловлен тем, что она(база данных) не занимает много места на жестком диске, достаточно быстрая, компактная и легко интегрируемая в проект, ведь в Django уже предустановлен движок SQLite. Для создания User Interface(Дальше UI) был использован CSS фреймворк Bootstrap, Javascript(дальше JS) библиотека Jquery, popper.js и fontawesome для шрифтов и иконок. Компонент View или логика сайта, будет разобрана позже.

Алгоритм действий игрока по прибытию на сайт. Сначала игроку необходимо зарегистрироваться или войти под уже существующем логином и паролем. Далее он должен выбрать команду и ввести пароль. Название команды и пароль дают перед началом самого мероприятия на стадионе. После того как он зашел в команду начинается игра. Его задача искать места по фотографии и чем быстрее он их найдет тем быстрее его команда выиграет. После каждого успешно найденного места команда получает часть ключа зашифрованного шифром Цезаря. Собрав все части ключа и расшифровав его, он узнают место последнего испытания.

Логика сайта и обратная сторона действий игрока. По прибытию игрока на сайт в первый раз он регистрируется на нем заполняя поля имени, почты и паролей. После нажатия кнопки Зарегистрироваться POST запрос отправляется на сайт, где его обрабатывает своя View форма. View форма проверяет валидность полей, прежде экранизирую их во избежании дыр в безопасности сайта. Если значения полей валидны, то затем в форме(это может быть функцией или классом) проверяется существует ли такой пользователь в базе данных, если нет, то пользователь вноситься в таблицу Users, со стандартными разрешениями в подгруппу Users. Пароль по занесению в базу данных шифруется по алгоритму PBKDF2 с хэшем SHA256, поэтому никто не сможет узнать пароль человека, ни хакер, ни администратор сайта. После успешной регистрации игрок перенаправляется на страницу с выбором команды. Там он увидит не только список с названиями, но и количество игроков которые уже зашли в лобби, кнопку Присоединиться, по нажатию которой, с помощью Bootstrap modals, плавно появиться окно с предложением ввести пароль для того чтобы войти в команду. Индикатор с количеством игроков статичен, и обновляется только по случаю обновления страницы, в целях оптимизации и нецелесообразности динамического индикатора. Menubar на странице выбора команды также присутствуют. Игрок может посмотреть карту зон, часто задаваемые вопросы, собранные им ключи или просто выйти. После ввода пароля команды GET запрос отправляется на backend сайта посредством AJAX к соответствующей функции, которая обрабатывает GET запросы такого типа. Функция проверяет эквивалентны ли пароль отосланный игроком и пароль команды, указанный как поле объекта команды в таблице Teams базы данных. Если результатом проверки является True, то игрок перенаправляется функцией shortcut-ом redirect() на страницу самой игры. Если же пароль неверен, то есть False, то его перенаправит на ту же страницу выбора команды и появится сообщение об ошибке, отображаемой bootstrap’ом как компонент Alert с классом danger. После перенаправления игрока на страницу с самой игрой устанавливаются Many-to-One отношения от объекта Gamer к объекту User, в следствии этого если человек выйдет с сайта и зайдет на него снова под своим логином и паролем, то его сразу перенаправит на страницу игры, без надобности ввода пароля команды.

На странице с самим фото-квестом игрок увидет фотографию места, таймер показывающий сколько времени осталось до конца квеста, свой прогресс, название миссии и три кнопки – сделать фото места, сделать селфи и получить следующую миссию. Все фотографии хранятся на сервере вместе с исходным кодом логики сайта. Таймер написан на Javascript. При обновлении страницы сайт отправляет время сервера и время конца миссии пользователю, Javascript сравнивает и показывает текущее время, постоянно обновляя его в цикле. Время и дата в формате short date. Так как название миссии и прогресс команды статичные данные не требующие обновления, то эта часть страницы генерируется на сервере с помощью шаблонизатора Django. Все кнопки на странице сделаны на bootstrap'е. По нажатию на кнопки сделать фото или сделать селфи появляется предложение сделать фотографию или выбрать ее из галереи. Сделано это с помощью возможностей HTML5. Были попытки использовать JS библиотеки, но они проигрывали в кроссплатформености и казались не рентабельными, в силу того что большинство из них просто делали ненужную работу. После того как фотографии были выбраны они сразу кодировались по алгоритму base64, а кнопка меняла свое состояние отображая готовность фотографии к отправке и проверке модераторам. Если пользователь пытался получить следующую миссию не сделав одну из двух обязательных фотографий, то на странице появлялось сообщение об ошибке, которую клиент получал в виде JSON данных от сервера по AJAX запросу. В случае если все фотографии сделаны, то после нажатия на кнопку для получения следующей миссии отправлялся GET AJAX запрос на сервер. На сервере сначало проверяется отправлял ли кто-то из участников команды фотографии, если нет то весь body request декодировался и затем сервер получал JSON данные для удобства работы. Далее из этих данных извлекались массивы с закодированными фотографиями, фотографии декодировали из base64 и затем в виде одного объекта сохраняли в таблице базы данных с названием answerToCheck, доступ к которой имел только модератор. Для удобства проверки ответов игроков была разработа еще одна страница сайта, на которой можно было увидеть все ответы которые присылали игроки. На этой странице отображалось состояние команды, ее прогресс, правилен ли их ответ или нет и сам ответ в виде фотографий. Модератор сравнивает фотографии и решает правилен ли ответ. Если да, то в базе данных этот ответ отмечается как правильный, то есть True, и когда пользовательский цикл который проверяет состояние ответа увидет что ответ является True, то клиент сразу же получит следующую миссию вместе с частью зашифрованного ключа, который указывает в комментарии к ответу модератор. Зашифрованные части ключа отображаются если ответ был дан правильно в виду всплывающего окна, а потом его можно увидеть нажав в Menubar'е на надпись "Собранные ключи". Тогда клиента перенаправит на страницу, где из базы данных, сравнивая идентификаторы команд, выбирается ключи именно его команды.

Проблемы в ходе разработки сайта. Race condition. Это серьезная проблема проектирования возникает в случае когда несколько управляющих потоков управляют одним общим ресурсом. Проблема в сайте заключалась на стадии игры, когда несколько игроков одной команды могли одновременно нажать на кнопку "Получить следующую миссию", тогда возникала ситуация что в панели администратора появлялось два ответа одной и той же команды, и тогда из-за невнимательности модератора появлялась возможность "проскочить" этап, но тогда команда осталась бы без части ключа. Решением проблемы было блокирование всех игроков, когда хотя-бы один из них уже подал ответ. В таком случае у игроков не было возможности одновременно подать ответ, ведь запросы не обрабатываются паралельно, а последовательно, но очень быстро из-за чего возникает ощущение многопоточности. Not bi-directional communication. Проблема

создавала неудобства в плане коммуникации клиента и сервера. К примеру, вместо того чтобы делать интервальные зацикленные запросы к серверу узнавая правильный ли ответ, можно было бы создать соединение на WebSocket'ах, однако ни один хостинг для Python такого пока не поддерживал, а если и поддерживал, то создавал глубокую дырку в кармане, потому пропадает производительность сайта. Создавать интервальный цикл для каждого из игроков проблематично, особенно это проблема была бы критичной в случае большого траффика игроков. Впрочем, с наличием собственного сервера это проблема бы исчезла.

Результаты

Информационные технологии позволили задействовать меньше людей в работе и открыли больше возможностей для полета фантазии в организации мероприятия. На создания сайта было потрачено порядка 15 евро. Времени на создание сайта потребовалось 2 недели. Практика внедрения подобных технологий пока что была одноразовой. По результатам проведенного опроса 70% участников понравилось мероприятие и идея с сайтом.

Анализ

Создание и разработка собственного сайта в реальном времени задача достаточно трудная и наверняка мероприятие не оправдывает сложности и затраты при разработке сайта, потому что разработка требует человека разбирающегося в программировании и в тонкостях разработки интернет сайтов. Однако же, возможно создать платформу облегчающую разработку подобных сайтов и в будущем использовать ее для подобных мероприятий.

Хотя сложность создания оказалась не оправданной для исследуемого мероприятия, все же перспектив использования информационных технологий большое множество.

Во-первых, информационные технология позволяют использовать меньше людей, хотя с другой стороны нужен человек который сможет программно скоординировать все элементы компьютерной системы.

Во-вторых, есть возможность использования геолокации, медиа файлов, звонков и смс и других функций портативных гаджетов для взаимодействия с игроками. Реализовав такое без информационных технологий, потребовалось бы намного больше людей.

В-третьих, информационные технология можно использовать не только во время проведения мероприятий, но и для планировки и обсуждения мероприятия.
Наконец, можно сказать, что разработка собственных сайтов и приложений есть ресурсо-затратная операция, однако однажды создав платформу, облегчающую разработку подобных в данной научной работе сайтов, возможно использовать ее многоразово и без проблем интегрировать в различного рода мероприятия. Впрочем, гипотеза приведенная в данной научной работе не подтвердилась, только для того случая, если разработка сайта или приложения производиться с нуля.

Выводы

Все поставленные задачи были выполнены. Реальный проект сайта был создан и внедрен в реальное мероприятие. Сайт успешно справился со своей задачей. Цель научной работы была достигнута. Перспективы внедрения информационных технологий исследованы. Гипотеза про то, что перспективы использования информационных технологий оправдывает сложность своего создания подтвердилась. Было использовано меньше людей для проведения мероприятии и использовались почти все возможности портативных гаджетов.

Используемая литература

1. Django documentation for 3.0 version.
2. Д. М. Златопольский.Основы программирования на языке Python.
3. ru.wikipedia.org/wiki/Состояние_гонки
4. https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
5. Марк Лютз. Программирование на Python.
6. https://en.wikipedia.org/wiki/HTTPS
7. https://lv.wikipedia.org/wiki/FTP
8. https://en.wikipedia.org/wiki/Git
9. https://lv.wikipedia.org/wiki/Secure_Shell
10. https://lv.wikipedia.org/wiki/GitHub
11. https://en.wikipedia.org/wiki/Domain_Name_System
12. https://help.pythonanywhere.com/pages/
13. https://docs.python.org/3/tutorial/datastructures.html
14. https://en.wikipedia.org/wiki/Linux
15. gutl.jovenclub.cu/wp-content/uploads/2013/10/Linux.Shell_.Scripting.Cookbook.pdf
16. https://git-scm.com/docs/git-shell

Приложения

Исходный код сайта - https://github.com/leanwise/kvest/tree/master/kvest_app/kvest