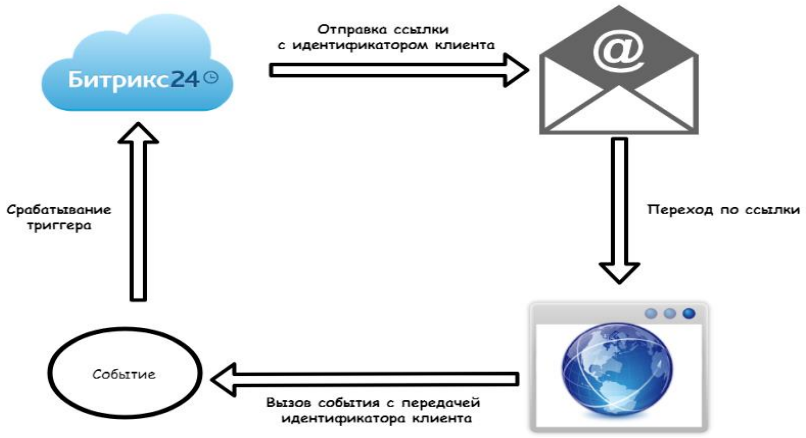

Рисунок 3 – Пример применения триггеров

Список используемых источников
1. CRM Битрикс24. [Электронный ресурс]. – Режим доступа: https://www.bitrix24.by/. – Дата доступа: 04.03.2018.
2. Создание приложений Битрикс24 [Электронный ресурс]. — Режим доступа: https://www.bitrix24.ru/apps/dev.php – Дата доступа: 04.03.2018.

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ОБУЧЕНИЯ ОСНОВАМ КРИПТОГРАФИИ

Институт информационных технологий БГУИР,
г. Минск, Республика Беларусь

Лупач В.Д.

Калитеня И.Л. – ассистент каф ИСиТ, м.т.н.

Сегодня едва ли удастся найти компанию, на компьютерах которой не обрабатывалась и не хранилась бы важная коммерческая информация. Полноценная система ее защиты имеет исключительное значение и без криптографии здесь никак не обойтись. Применительно к компаниям шифрование обычно используется в трех случаях, а именно в целях обеспечения безопасности хранения данных, защиты информации при передаче через открытые каналы связи и по локальной сети, наконец, для цифровых подписей. Все представленные задачи решаются разными средствами с применением различных криптографических технологий.

В современных корпоративных сетях вся важная информация сосредоточена на одном или нескольких серверах. Такой подход по сравнению с хранением данных на компьютерах конечных пользователей имеет множество преимуществ. Первостепенной задачей здесь является обеспечение надежного шифрования документов именно на сервере, но при этом система безопасности не должна серьезно осложнять работу сотрудников с информацией, которая необходима для исполнения служебных обязанностей.

Знание криптографических методов важно для развития и профессионального роста программистов в современном мире. Программисты с подобными знаниями смогут профессионально заниматься защитой корпоративной информации.

В дипломном проекте ставилась задача создать программное средство, которое могло бы обучать практическим основам криптографии и проверять знания учащихся. Ставилась задача создать программное средство, которое помогало бы в процессе обучения, а также в котором учащийся смог бы самостоятельно изучить пропущенные темы. Процесс обучения, как и процесс проверки знаний, проходит в несколько этапов. В режиме обучения, при неправильных действиях учащегося, подсказки в каждом этапе помогут найти и исправить свою ошибку. При проверке знаний, учащийся будет иметь возможность проверить свой ответ и продемонстрировать процесс решения.

Программное средство имеет следующие функции:
- авторизация учащегося;
- выбор режима администратора;
- выбор криптографического метода;
- реализация двух методов: обучения и проверки знаний;
- реализация возможности ввода любых текстов для зашифровки и любых возможных ключей;
- помощь и подсказки при обучении;
- проверка знаний по какому-либо криптографическому методу;
- сохранение оценки пользователя в базе данных;
- сохранение в базе данных времени, затраченного на выполнение уча-щимся задания;
- просмотр оценок учащихся администратором.

Список использованных источников:
1. Криптографические основы безопасности. [Электронный ресурс]. – Режим доступа: http://www.intuit.ru/department/security/networksec/ - Дата доступа: 27.12.2017

ОСОБЕННОСТИ КЭШИРОВАНИЯ КОМПОНЕНТОВ В UNITY

Институт информационных технологий БГУИР,
г. Минск, Республика Беларусь

Мультан Р.И. Мазур А.Д.

Бакунова О.М., ст. преподаватель каф ИСиТ, м.т.н.
Образцова О.Н., доцент каф. ИСиТ, к.т.н,, доцент

Огромная часть Unity-разработчиков понимают, что не стоит злоупотреблять дорогими для производительности вычислениями, к таким вычислениям относится и получение компонентов. Чтобы избежать дорогих вычислений необходимо использовать кэширование. В докладе рассмотрены различные способы кэширования, их реализация, особенности и производительность.

Данная статья будет рассматривать в основном внутреннее кэширование — это процесс получения различных компонентов на объекте. Объект в Unity, находящийся на сцене, представляет собой GameObject - т.е. вместилище для разнообразных компонентов. Компоненты бывают стандартными (Rigidbody, RectTransform, Animator), так бывают и написанными разработчиками(класс, который наследуется от MonoBehaviour). Для того, чтобы получить компонент на объекте можно использовать метод GetComponent(), но не стоит им сильно злоупотреблять. Если компонент используется в скрипте не одиножды, существует подход при котором возможно объявить его переменную и получить его компонент с объекта всего лишь один раз с помощью метода GetComponent() (рисунок 1)

.

```
public class CachingExample : MonoBehaviour
{
    private Rigidbody rigidbody;

    void Start()
    {
        rigidbody = GetComponent<Rigidbody>();
    }

    void Update()
    {
        rigidbody.AddForce(Vector3.up * Time.deltaTime);
    }
}
```

Рисунок 1 – Объявление компонента, как переменную.

Заметим, что прямое получение компонентов на объекте более производительно, чем остальные способы кэширования, в большинстве случаев. Другие вариации кэширования опираются на базовое и упрощают нам доступ к компонентам, лишая нас необходимости писать один и тот же код, когда нам необходим какой-либо компонент.

Для следующего метода кэширования используются свойства. Свойство - это разновидность члена класса, предоставляющий гибкий механизм для чтения, записи и вычисления значения частного поля. Явным преимуществом этого метода является то, что кэширования произойдет только тогда, когда мы первый раз обратимся к свойству. Явным недостатком является то, что нам приходится писать больше однообразного кода (рисунок 2).