жения. Кроме времени выполнения операции, можно оценить загрузку ЦП и количество используемой памяти при работе приложения.

3. Разработка нативного приложения, которое будет содержать в себе элементы гибридного приложения. При этом различные части приложения будут выполнять одинаковые функции, а приложение будет фиксировать время выполнения для каждого из вариантов. Как результат, можно вывести сводную сравнительную информацию о времени выполнения аналогичных операций.

В качестве вывода следует отметить, что при выборе способа разработки приложений необходимо четко понимать, какие цели оно будет преследовать и какие функции выполнять. Если это достаточно простое приложение, которое должно работать на различных платформах и не требующее доступа к низкоуровневым функциям устройства, следует выбирать одну из гибридных технологий. Если же это сложный продукт, использующий множество аппаратных ресурсов, то лучшим вариантом будет разработка нескольких вариантов нативных приложений.

Список использованных источников:

1. А.Хиллегасс, Программирование под iOS для професисоналов, 608 стр. (2013 г.);
2. Lee Barney, Developing Hybrid Applications for the iPhone, 183 стр. (2009 г.).

МУЛЬТИПЛАТФОРМЕННЫЙ СЕРВИС ЛОГИРОВАНИЯ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Шабанец Я. Р.

Волосевич А. А. – канд. физ.-мат. наук, доцент

Грамотно спроектированная архитектура современных программных продуктов включает в себя модули обработки ошибок и ведения журнала событий, возникающих в ходе работы приложения. Возникновение ошибок в программе во многих случаях обусловлено некорректными данными, переданными приложению, либо некорректной последовательностью действий пользователя при работе с программой. Для обнаружения и устранения проблем, обнаруженных в системе, разработчики используют записи из журналов ошибок и сообщений приложения для восстановления контекста выполнения, приведшему к возникновению проблемы.

Мультиплатформенный сервис логирования предназначен для ведения журнала сообщений приложения, их хранения и фильтрации, предоставления статистики в режиме реального времени. Сервис является готовым решением, которое можно применять в существующих или проектируемых системах без необходимости разработки собственного решения, уменьшая время, необходимое на разработку и затраты на создание программных продуктов.

Проект состоит из нескольких частей: сервис приема сообщений, сервис поставки сообщений, компонент для высокоэффективной работы с хранилищем сообщений, библиотеки классов для различных платформ, содержащие в себе модули для работы с сервисами и веб интерфейс сервиса.

Классы для работы с сервисом предоставляют разработчикам несколько перегруженных методов: Debug, Info, Warn, Error, Fatal для отправки сообщений на сервис и возможность подписывать собственные обработчики сообщений, получаемых сервисом из других уровней приложения.

Веб интерфейс сервиса предоставляет доступ к архиву сообщений приложения с возможностью экспорта, фильтрации и сортировки данных.

Сервис реализован с использованием следующих технологий платформы .NET: ASP.NET MVC4 для реализации веб интерфейса, ASP.NET Web API для реализации сервисов приема и поставки сообщений.

Для хранения данных, в целях повышения производительности, было выбрано документо-ориентированное сетевое хранилище данных типа «ключ-значение» с открытым исходным кодом Redis. Redis хранит базу данных в оперативной памяти, снабжена механизмами снимков и журналирования для обеспечения постоянного хранения.

Отличительной чертой реализации логирования является доступ и синхронизация сообщений всех уровней приложения, которые могут быть разделены физически и написаны с использованием различных технологий. Также ведение журнала событий потребляет ресурсы приложения, таких как дисковое пространство, процессорное время и оперативная память. Использование сервиса позволяет избежать подобных затрат на серверах разрабатываемой системы и улучшить ее нагрузочные характеристики.

Разработанное решение может применяться для ведения журнала сообщений приложений и может использоваться в компаниях, занимающихся разработкой программного обеспечения, для диагностики создаваемых или сопровождаемых программных продуктов.

Список использованных источников:

1. Tiago Macedo, Fred Oliveira Redis Cookbook / Tiago Macedo, Fred Oliveira – O'Reilly Media 2011. – 78 p.
2. Jamie Kurtz ASP.NET MVC 4 and the Web API / Jamie Kurtz – Apress, 2013. – 152 p.