Санкт Петербургский
Государственный Университет
Математико-механический факультет

Магазин приложений для роботов ТРИК

Курсовая работа студента 371 группы
Евтушенко Владислава Валерьевича

Научный руководитель: ст. преп. Брыксин Тимофей Александрович

Санкт-Петербург
2016

Оглавление

Введение

Сегодня Интернет предоставляет множество возможностей, начиная от отправки письма и заканчивая покупкой дома. Количество задач, которые можно решить с помощью Всемирной сети, постоянно растет. Также наблюдается тенденция переноса популярных приложений в веб. Самый яркий пример – это появление набора офисных приложений Google Docs[1] и вслед за ними Microsoft Office Online[2].

С появлением сети Интернет пришли и онлайн-магазины, в которых можно найти все, что угодно, в том числе и программное обеспечение. Интернет-магазины, в которых продают или предлагают бесплатно программы, называются магазинами приложений.

Магазин приложений – центральное приложение во многих платформах, которое позволяет пользователям устанавливать, обновлять и удалять приложения на своих устройствах. Обычно магазин приложений – это удобный каталог, где пользователь легко может выбрать нужное ему приложение из множества доступных. Такие магазины есть у Ubuntu[3], Windows[4] (начиная с версии 8.0), Google Android[5], Apple iOS[6]. Бывают также и сторонние для платформы магазины приложений, например, Steam[7] для Windows.

Роботов, созданных на основе контроллера ТРИК[8], можно рассматривать как самостоятельную платформу, вокруг которой сформировалось сообщество разработчиков и пользователей. Для платформы ТРИК существуют инструменты, например, TRIK Studio[9], которые позволяют создавать программы для робота, в том числе и с помощью графических схем. Созданные программы можно переиспользовать на многих роботах, а для этого их необходимо разместить централизованно в одном общедоступном ресурсе. Таким ресурсом станет магазин приложений для роботов.

[1] https://www.google.com/docs/about/
[2] https://www.office.com/
[3] https://apps.ubuntu.com/cat/
[4] https://www.microsoft.com/ru-ru/store/apps
[5] https://play.google.com/store?hl=en
[6] https://itunes.apple.com/en/genre/ios/id36?mt=8
[7] http://store.steampowered.com/
[8] http://www.trikset.com/
[9] http://blog.trikset.com/p/trik-studio.html

Цели и задачи

Цель курсовой работы заключается в создании магазина приложений для роботов на платформе ТРИК, который в дальнейшем также можно будет использоваться для конфигурирования периферийных устройств робота.

Для достижения цели были выделены следующие задачи:

- Изучить предметную область и существующие решения.
- Выбрать необходимые для решения задачи инструменты.
- Перенести функциональность существующего решения на платформу .NET[10].
- Дополнить существующее решение возможностью обновлять и удалять программы, установленные на роботе.
- Развернуть решение в облачном сервисе.
- Провести апробацию решения на нескольких роботах ТРИК.

[10] https://www.microsoft.com/net

Обзор существующих решений

Существующие магазины приложений

Магазины приложений можно разделить на две категории – встроенные и сторонние.

На некоторых платформах, встроенные и сторонние магазины приложений могут успешно сосуществовать, например, на мобильной операционной системе Google Android помимо стандартного Google Play можно использовать Amazon AppStore[11]. Но, в частности, на Apple iOS использование сторонних приложений и магазинов не приветствуется.

Обычно встроенные магазины популярнее сторонних. Например, на платформе Android, у Amazon Appstore в 2015 году было в 4 раза меньше приложений, чем у Google Play. Но есть и исключения, например, на платформе Windows в поисках игр пользователи предпочтут сторонний Steam встроенному магазину Windows Store.

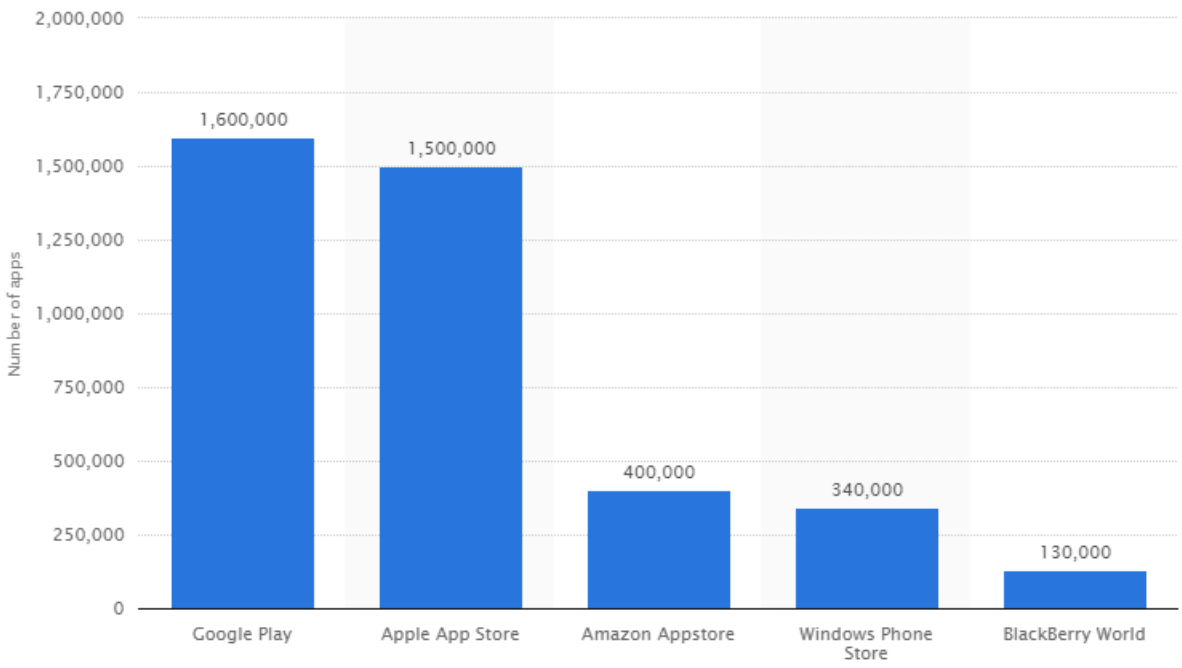

Рисунок 2. Количество приложений в магазинах приложений по данным на июль 2015 года[12].

Некоторые магазины приложений, являются частью более крупного магазина. Например, сервис Google Play – это не только магазин приложений, еще в нем можно

[11] http://www.amazon.com/gp/mas/get-appstore/android/ref=mas_rw_ldg
[12] http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

приобрести игры, фильмы, музыку и книги. В тоже время, Steam – специализируется на играх.

Большинство магазинов приложений требует от разработчиков плату за размещение в них приложений. Для Google Play 25$ одноразовая плата, в Apple Appstore и Amazon Appstore плата составляет 99$ ежегодно. В Steam просят единовременный взнос в 100 $, который будет перечислен на счет благотворительной организации Child's Play[13].

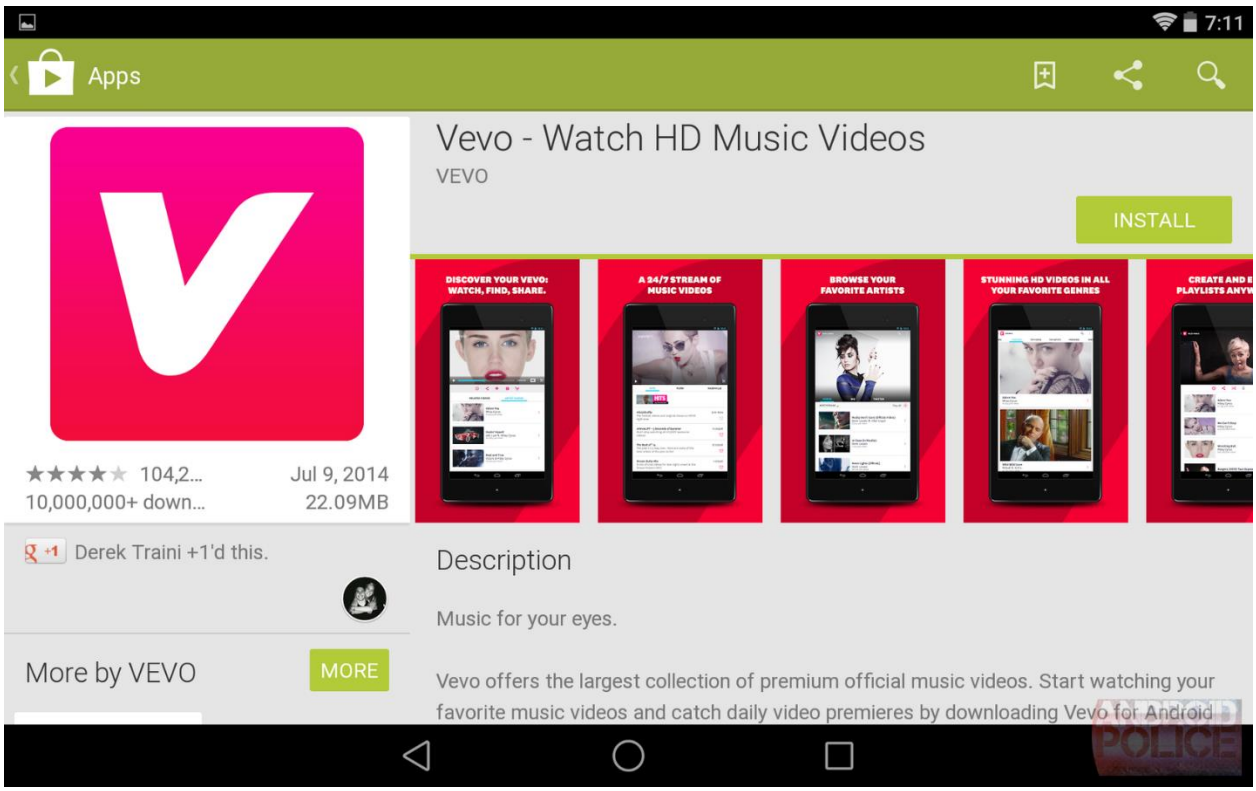

Рисунок 3. Интерфейс магазина приложений Google Play для платформы Android.

В результате анализа существующих магазинов приложений мы установили общие свойства и функции присущие большинству магазинов приложений:

- Наличие рейтингов и отзывов.
- Наличие технических требований к устройству пользователя.
- Наличие описания приложений, с возможностью прикреплять изображения и видео.
- Проверка приложений от разработчиков на соответствие правилам магазина приложений.

[13] http://childsplaycharity.org/

- Структурированность по категориям, удобный для пользователя поиск и рекомендации.
- Контроль версий приложений, уведомления пользователя о доступных обновлениях.

Архитектура проекта строилась с расчетом на поддержку перечисленных свойств и функций в будущем.

Пример пользовательского интерфейса магазина приложений изображен на рисунке 3.

Реализация Д.А. Агеева

В 2015 году в рамках бакалаврской выпускной квалификационной работы студента 444 группы Дениса Агеева был представлен проект – инфраструктура для конфигурирования и отправки программ на робота [1]. По ряду причин проект не был завершен, но потребность в нем сохранилась.

Проект позволял пользователю, работая с высокоуровневым веб-интерфейсом, устанавливать программы на робота и конфигурировать его периферийные устройства. Основные компоненты решения Д.А. Агеева показаны на рисунке 1.

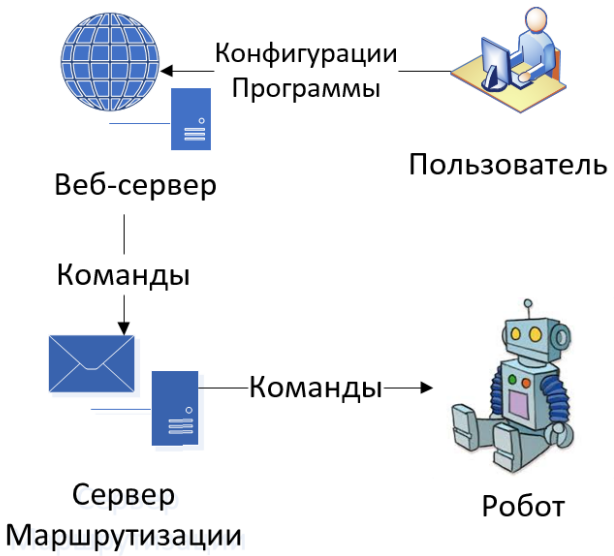

Рисунок 1. Архитектура решения Д.А. Агеева.

В данном решении остался ряд недостатков:

- Излишняя сложность архитектуры серверной части. В рамках решения была создана отдельная обособленная компонента – сервер маршрутизации, задача которого была лишь в посредничестве при общении между сервером и роботом.

- По сравнению с взаимодействием сервера и робота напрямую, общение с использованием сервера маршрутизации в 2 раза дороже, потому что сообщения проходят 2 сериализации и 2 десериализации данных.
- Клиентский код встроен в прошивку контроллера. Чтобы проверить работоспособность системы на роботах, нужно будет обновить прошивку на каждом из них. На этапе прототипа удобнее выделить клиентскую часть в стороннюю программу.

В связи с этим было принято решение разработать новую серверную и клиентскую части системы на основе полученного опыта.

Решение

Взаимодействие компонент системы

Разработанное решение состоит из трех основных частей: веб-сервера, веб-интерфейса пользователя и клиентской части робота. Взаимодействие этих компонент схематично показано на рисунке 4.

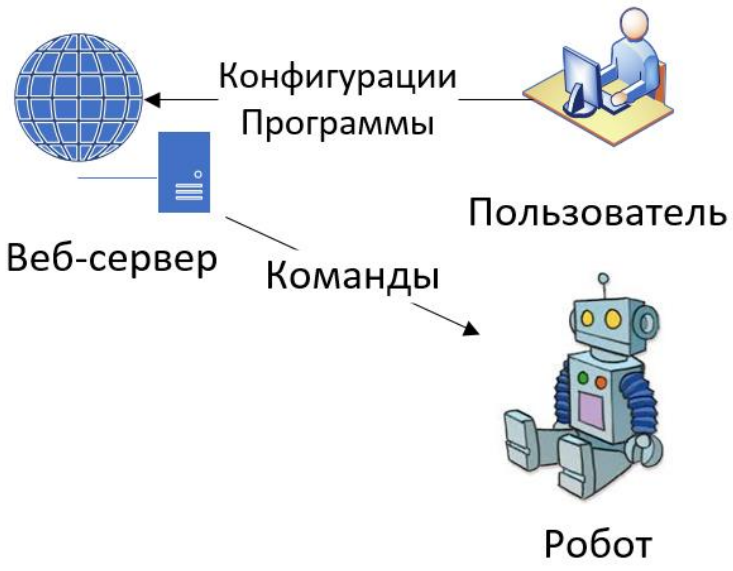

Рисунок 4. Взаимодействие основных компонент решения.

Для конечного пользователя разработан высокоуровневый интерфейс. С помощью этого интерфейса пользователь устанавливает, удаляет или обновляет программы робота.

При работе пользователя с интерфейсом, последний отправляет запросы на веб-сервер. Веб-сервер обрабатывает запросы и транслирует некоторые действия пользователя в соответствующие команды для робота.

Робот с некоторой периодичностью запрашивает команды у веб-сервера. Получив команду, робот осуществляет попытку исполнить ее. Далее робот отправляет веб-серверу отсчет о выполнении команды. Веб-сервер обрабатывает отсчет и отдает пользователю актуальную веб-страницу.

Веб-сервер

Веб-сервер состоит из трех проектов – библиотеки классов, в которой хранится вся бизнес-логика и сущности проекта, непосредственно веб-приложения (пользовательский веб-интерфейсы и интерфейс прикладного программирования для

взаимодействия с роботом) и отдельного проекта с модульными тестами [2]. Их взаимодействие показано на рисунке 5.

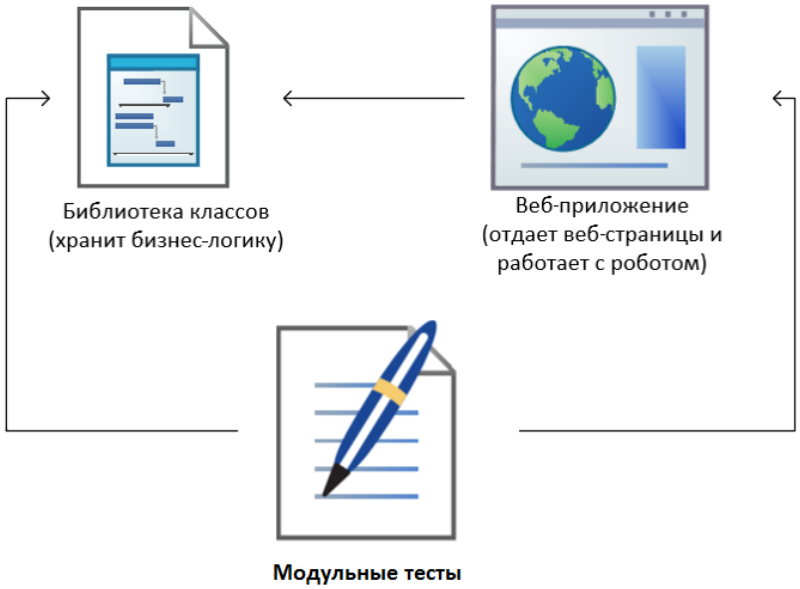

Рисунок 5. Основные компоненты веб-сервера.

Библиотека классов состоит из сущностных и управляющих классов. Пример сущностных классов – это робот, программа, пользователь. Управляющие классы манипулируют объектами сущностных классов, образуя действия бизнес-логики.

Веб-приложение состоит их двух модулей: интерфейс прикладного программирования для взаимодействия с роботом и функции, обеспечивающие работу пользовательского интерфейса. Взаимодействие как с роботом, так и с интерфейсом пользователя происходит по протоколу HTTP[14]. При проектировании веб-приложения применялся шаблон проектирования Model View Controller [3] для архитектуры в общем, для работы с данными использовались шаблоны Repository и Singleton [3].

Модульных тестов в проекте более тридцати, они написаны для каждой функции с целью большего контроля процесса разработки [4]. Тесты бывают двух видов: для бизнес-логики и для интерфейса прикладного программирования, используемого роботом. Тесты бизнес-логики – это простая проверка, вместо базы данных используются массивы с тестовыми данными, тогда как тесты для интерфейса прикладного программирования немного сложнее. Они представляют собой обращение к запущенному экземпляру приложения и проверяются на реальных данных.

Данные веб-сервер хранит в базе данных. При работе с ней используется подход «код вперед» [5], при котором таблицы базы данных создаются по сущностным классам

[14] https://tools.ietf.org/html/rfc2616

бизнес-логики. При чтении и записи данных в базу, данные представляются в виде объектов (Object-Relational Mapping) [5].

Для платформы .NET самая крупная и удобная библиотека для создания веб-приложений – это ASP.NET[15]. В качестве языка программирования под этот инструмент был выбран C#.

Веб-интерфейс

Одной из особенностей магазина приложений для роботов на платформе ТРИК является отсутствие возможности размещения удобного пользовательского интерфейса непосредственно на устройстве – контроллере. По этой причине интерфейс пользователя представлен в виде веб-приложения. Часть веб-интерфейса изображена на рисунке 6.

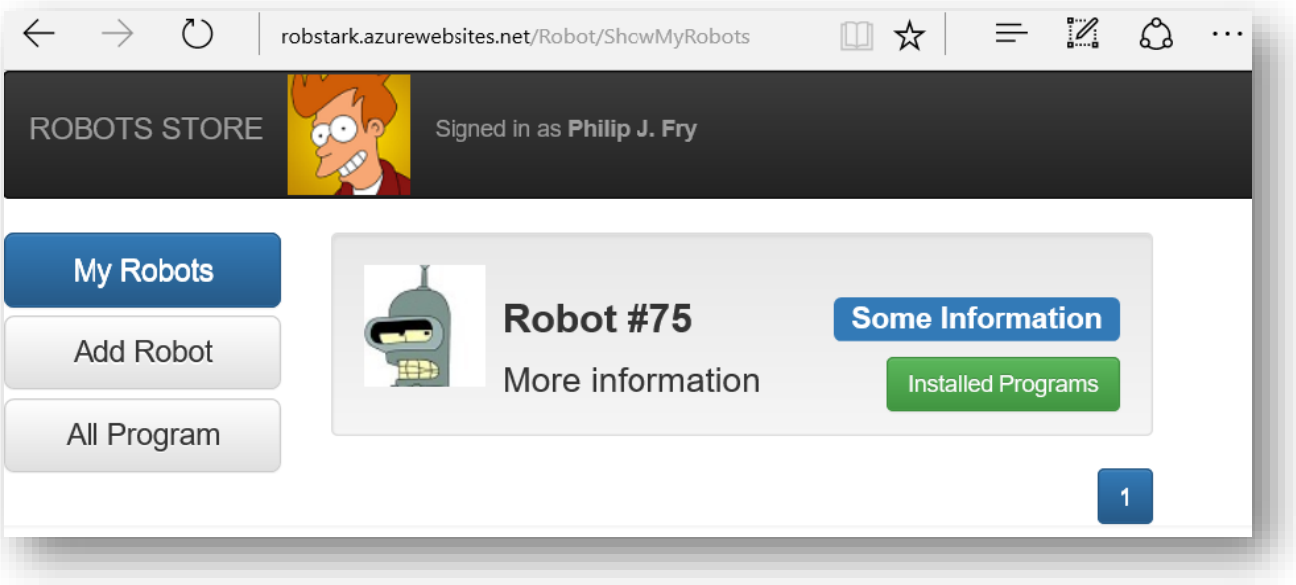

Рисунок 6. Часть веб-интерфейса пользователя.

Веб-интерфейс позволяет управлять установленным программным обеспечением на роботах пользователя, а также изменять параметры периферийных устройств контроллера. И еще можно посмотреть список всех доступных программ и отправить понравившуюся на робота.

Веб-интерфейс представляет собой несколько веб-страниц с двумя общими мастер-страницами. Первая для навигации, вторая – это статус-панель в верхней части страницы. Взаимодействие с веб-сервером происходит с помощью отправки асинхронных HTTP запросов (Ajax) на специально созданный HTTP интерфейс [2].

[15] http://www.asp.net/

Для обеспечения взаимодействия между элементами веб-страниц используется библиотека JQuery[16] [6], а для создания минимального дизайна используется библиотека Bootstrap[17].

Клиентская часть

Клиентская часть представляет собой Python скрипт, состоящий из трех компонент. Он работает в двух потоках: основной для интерфейса разработчика и фоновый для исполнения команд пользователя [7]. Основные элементы изображены на рисунке 7.

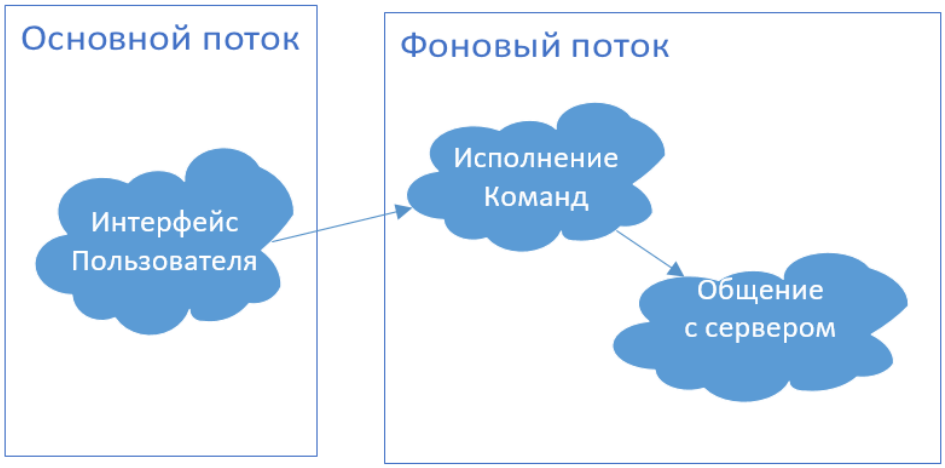

Рисунок 7. Архитектура клиентской части.

На стадии прототипа интерфейс клиентской части – это консольное приложение с набором команд для разработчика. Далее консольное приложение превратится в системный процесс (демона).

Для общения робота с сервером был создан отдельный модуль – веб-клиент. Для получения команд и программ робот отправляет на сервер HTTP Get запросы, для отправки отсчетов о выполнении команд используются HTTP Post запросы.

Получив команды, робот записывает их в файл или добавляет к уже существующим, если файл не пуст. Далее робот последовательно считывает команды из файла и выполняет их. Выполнение команд происходит в отдельном модуле, который работает с файловой системой контроллера, добавляя, обновляя и удаляя программы.

Также у робота имеется отдельный конфигурационный файл для работы с магазином приложений. В нем хранится уникальный ключ доступа к веб-серверу. При первом запуске клиентской части робот регистрируется и получает уникальный ключ от

[16] https://jquery.com/
[17] http://getbootstrap.com/

веб-сервера, ключ хранится в конфигурационном файле. При последующих обращениях к веб-серверу робот вкладывает свой ключ в заголовок HTTP запроса.

Развертка решения

Во время разработки веб-сервер, база данных и клиентская часть размещаются локально на одной машине. Клиентская часть – кроссплатформенный Python[18] скрипт, который одинаково работает как под операционной системой Windows, так и под Linux[19], на которой работает контроллер ТРИК. Веб-сервер работает под управлением локального экземпляра Microsoft Internet Information Services[20] (IIS), база данных работает под управлением локальной системы управления базами данных Microsoft SQL Server[21].

Далее работоспособные версии клиентской и серверных частей размещаются на роботе и на облачном сервисе Microsoft Azure[22] соответственно. В рамках программы Microsoft BizSpark[23] мы можем бесплатно[24] пользоваться услугами размещения веб-приложения и базы данных на сумму до 10 000 рублей в месяц. Этого более чем достаточно, чтобы обеспечить работу веб-сервера с несколькими роботами-клиентами.

[18] https://www.python.org/
[19] https://www.linux.com/
[20] http://www.iis.net/
[21] https://www.microsoft.com/en-us/server-cloud/products/sql-server/
[22] https://azure.microsoft.com/ru-ru/
[23] https://www.microsoft.com/bizspark/
[24] В рамках участия в программе Microsoft Bizspark в течение двух лет.

Апробация решения

Мы взяли робота на контроллере ТРИК, подключились к нему по протоколу SSH[25], скопировали на него скрипт клиентской части. При первом запуске скрипт регистрирует робота в системе, получает ключ для дальнейших запросов к серверу и код активации.

Робот отображает пользователю код активации. Далее пользователь регистрируется или входит в систему, переходит в раздел добавления роботов, вводит код активации и добавляет нового робота.

Затем пользователь работает со списком программ, выбрав нужную, он находит своего робота в списке и нажимает кнопку “отправить”. В этот момент веб-сервер обрабатывает запрос и создает команду установки программы для робота.

В любой момент робот может обратиться к веб-серверу для получения команд, по умолчанию это происходит раз минуту. Получив команду, робот распознает ее и понимает, что это – команда установки программы. Далее робот делает запрос на получение этой команды. После этого он устанавливает ее, то есть приводит к виду, пригодному для запуска. Завершив установку, робот отправляет отсчет об установке программы веб-серверу.

Сервер принимает запрос, обрабатывает его и изменяет веб-станицу, содержащую информацию о программах, установленных на роботе. Теперь у пользователя появляются возможности обновить или удалить недавно установленную программу с робота.

Подробно этот процесс показан на рисунке 8.

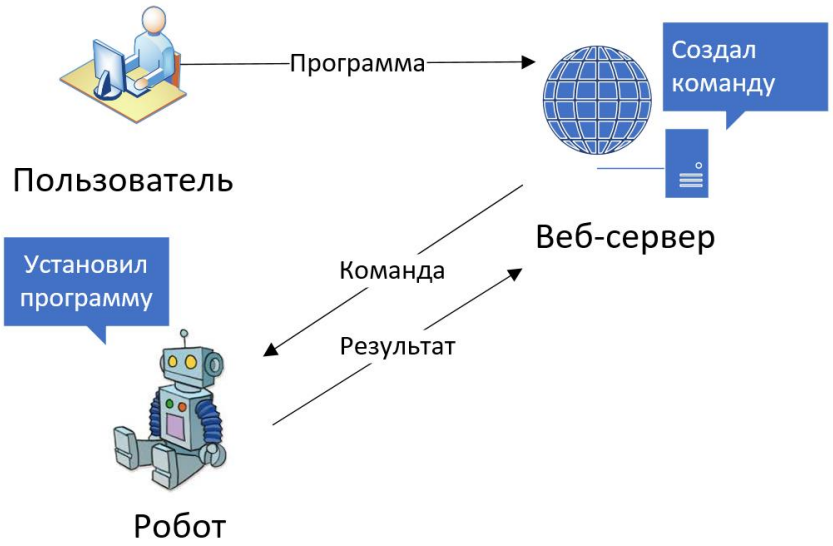

Рисунок 8. Процесс установки новой программы на робота.

[25] https://tools.ietf.org/html/rfc4253

Результаты

В результате курсовой было сделано:

- Изучены существующие магазины приложений и существующее решение Д.А. Агеева.
- Функциональность существующего решения, дополненная возможностями обновления и удаления установленных программ на роботе, перенесена на платформу .NET.
- Полученное решение развернуто в облаке Microsoft Azure.
- Функциональность существующего решения опробована на роботе ТРИК.

Направления дальнейшей деятельности

Интеграция со средой ТРИК

В рамках среды ТРИК существуют такие проекты как QReal: Web[26] и Trik Studio. Одна и первых следующих итераций будет интеграция с ними. Программы, созданные в этих инструментах, можно будет размещать в магазине приложений и, наоборот, программы из магазина приложений можно будет редактировать внутри этих инструментов.

Веб-интерфейс

Одна из современных тенденций в веб-разработке – это создание одностраничных веб-приложений. По сравнению с многостраничными их создание сложнее, но скорость работы выше за счет загрузки небольших элементов в рамках одной страницы. В следующих итерациях веб-интерфейс будет изменен с многостраничного на одностраничный, используя библиотеку Angular JS[27].

Для небольших интернет-магазинов принято использовать готовые движки (платформы) для сайтов, такие как WordPress[28]. А также готовые системы управления содержимым сайтов (Content Management System). Их использование облегчит разработку веб-интерфейса, а также позволит использовать более удачные решения в дизайне и пользовательском интерфейсе, прилагая при этом меньше усилий.

В рамках курсовой работы не был реализован ряд популярных функций и свойств магазинов приложений, а именно комментарии, рейтинги, отзывы, поиск и категории. Эти детали будут реализованы в следующих итерациях.

Также будет добавлена возможность конфигурирования периферийных устройств робота.

Веб-сервер

Существует вариант размещения бизнес-логики на уровне базы данных, далее будут исследованы преимущества этого варианта в рамках решения. Также бизнес-логику можно экспериментально перенести на другой язык программирования среды .NET, например, на F#[29] и сравнить скорость разработки и скорость работы системы с полученной реализацией на C#.

[26] https://github.com/qreal/qreal-web
[27] https://angularjs.org/
[28] https://wordpress.org/
[29] http://fsharp.org/

Список литературы

1. Агеев Д.А. Бакалаврская работа «Создание онлайн инфраструктуры для конфигурирования и управления роботами» // СПбГУ, Математико-механический факультет, кафедра Системного программирования. 2015. 23 p.
2. Adam Freeman. Pro ASP .NET MVC 5 // Apress. 2013. 832 p.
3. Erich Gamma, Richard Helm, Ralph Johnson, John Vissides. Design Patterns: Elements of Reusable Object-Oriented Software // Pearson Education. 1994. 383 p.
4. Matt Stephens, Doug Rosenberg. Design Driven Testing: Test Smarter, Not Harder // Apress. 2010. 368 p.
5. Sandeep Chandra, Damien Foggon. Beginning ASP.NET 4.5 Databases // Apress. 2013. 280 p.
6. Jack Franklin. Beginning JQuery // Apress. 2013. 204 p.
7. Gregory Walters. The Python Quick Syntax Reference // Apress. 2013. 152 p.