SOI: 1.1/TAS DOI: 10.15863/TAS

International Scientific Journal

Theoretical & Applied Science

p-ISSN: 2308-4944 (print) **e-ISSN:** 2409-0085 (online)

Year: 2019 **Issue:** 01 **Volume:** 69

Published: 15.01.2019 **http://T-Science.org**

SECTION 4. Computer science, computer engineering and automation.
UDC 004.4

QR – Issue QR – Article

Vadim Andreevich Kozhevnikov
Senior Lecturer
Peter the Great St. Petersburg Polytechnic University
vadim.kozhevnikov@gmail.com

Oleg Yurievich Sabinin
Candidate of Engineering Sciences, Associate Professor
Peter the Great St. Petersburg Polytechnic University
olegsabinin@mail.ru

Alla Nurbievna Tyulparova
student
Peter the Great St. Petersburg Polytechnic University
gelny@yandex.ru

PREREQUISITES ANALYSIS AND DEVELOPMENT OF AN AUTOMATED COMPOSITE SYSTEM FOR COMMERCIAL ELECTRIC POWER METERING

Abstract: *The objects of the analysis are the automated composite system for commercial electric power metering using the open source program libraries and stack of modern technologies.*

The purpose of the work is the development concrete composite system for commercial electric power metering.

Key words: *replication strategy, power line communication protocol, distributed computer system.*

Language: *Russian*

Citation: Kozhevnikov, V. A., Sabinin, O. Y., & Tyulparova, A. N. (2019). Prerequisites analysis and development of an automated composite system for commercial electric power metering. *ISJ Theoretical & Applied Science, 01 (69)*, 56-62.

Soi: http://s-o-i.org/1.1/TAS-01-69-12 ***Doi***: https://dx.doi.org/10.15863/TAS.2019.01.69.12

АНАЛИЗ ПРЕДПОСЫЛОК И РАЗРАБОТКА КОМПЛЕКСНОЙ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ КОММЕРЧЕСКОГО УЧЕТА ЭЛЕКТРОЭНЕРГИИ

Аннотация: *Объектом исследования являются системы комплексного учета электроэнергии с использованием открытых программных продуктов и технологий*

Цель работы – разработка конкретной интегрированной комплексной системы автоматизированного учета электроэнергии.

Ключевые слова: *метод репликации, протокол передачи данных по силовым линиям, распределенная компьютерная система.*

Введение

В темпе современной жизни, где все больше внимания уделяется удобству пользователя, высокие технологии востребованы везде и всюду. В частности, в сфере жилищно-коммунального хозяйства (ЖКХ) с недавнего времени широко используются программно-технические комплексы (ПТК) по приему, учету и расчету платежей за услуги ЖКХ, чему способствует непрерывное развитие современной элементарной базы, предоставление практически всеми производителями приборов учёта интерфейсов для дистанционного считывания показаний. Несмотря на, казалось бы, подготовленную базу, ниша полнофункциональных систем, объединяющих в

себе возможности контроля и учёта потребления электроэнергии, формирования документов для оплаты потребления, а также учёта произведённых платежей, остаётся в значительной степени не заполненной.

Совершим экскурс в недалёкое прошлое: совсем недавно в каждом садовом домике был установлен допотопный электросчётчик, данные с которого при возможности считывал бухгалтер садоводства, а то и сам потребитель. Показания по участкам суммировались, но их сумма оказывалась далека от показаний общего счётчика садоводства, по которому расплачиваются с поставщиком энергии. Недоимки восполняли всем миром, за счёт назначения целевых взносов, членских платежей, повышенных тарифов. Но развитие технологий и удешевление элементарной базы, сделали возможной установку у каждого индивидуального потребителя (как правило, вне участка, на столбе ЛЭП) электронного двухтарифного счётчика с возможностью дистанционного считывания показаний [2].

Постановка задачи

Для передачи данных от счетчиков используется протокол PLC (power line communication) – силовая линия 0,4 кВ. Данные собираются специальным концентратором, часто установленным в трансформаторной подстанции, и, с использованием сетей Ethernet, GSM или оптоволокна, через Интернет или напрямую передаются на компьютер для обработки. Часто станцией обработки данных служит ПК или ноутбук правления, или председателя садоводства. Иногда данные о киловатт-часах потребления выдаются в сыром виде, и приходится использовать бумажные либо электронные таблицы для пересчёта в рубли. Поставщики оборудования, как правило, снабжают его минимальным набором драйверов и утилит, что, безусловно, решает проблему безучётного потребления, но несколько затрудняет жизнь правления СНТ и, кроме того, не позволяет самим потребителям наблюдать свои величины. Разрабатываемый программный комплекс призван устранить вышеозначенные недостатки.

Было принято решение о создании комплексной автоматизированной системы коммерческого учёта электроэнергии, способной взаимодействовать с оборудованием различных производителей, предоставлять потребителям необходимую информацию, используя веб-интерфейс, предоставлять широкие возможности для администрирования, получения статистической информации, совмещая это с возможностью регулярного резервного копирования, простотой обслуживания и быстротой разработки.

Сформулированы следующие задачи:

- Должен вестись централизованный автоматический сбор информации с приборов электроэнергоучета.
- Необходимо реализовать интеграцию с оборудованием различных производителей.
- Обновление информации должно проходить не реже раза в сутки.
- Доступ в систему должен быть возможен с любого компьютера, подключенного к сети Интернет, посредством единого веб сайта.
- Аутентификация и авторизация системы должны обеспечить максимальную безопасность личных данных пользователя.
- Система должна автоматически вести учёт вносимых платежей и расчёт сумм к оплате по каждому участку в соответствии с установленными тарифами.

Теоретические основы работы

Встал вопрос о выборе используемых технологий. Основными критериями при выборе были быстрота получения первого рабочего функционала (или проще говоря скорость разработки), актуальность выбранного стека технологий, что позволит избежать необходимости быстрого переезда на что-то новое, возможность и простота масштабирования по мере увеличения нагрузки. Всем выставленным требованиям в значительной степени удовлетворяет стек технологий от Microsoft, базирующийся на использовании MS SQL [8] в качестве сервера БД.

Это современное, высокопроизводительное хранилище данных, готовое к нагрузкам, значительно превосходящим ожидаемые, и готовое к масштабированию. Начиная с версии 2017, анонсирована возможность размещения MS SQL на платформе Linux, а также в Docker, что предоставляет более широкие возможности размещения нашего малобюджетного проекта высокой степени доступности. Как правило, Linux хостинг значительно дешевле аналогичного по мощности хостинга на базе Windows Server, поскольку не требует оплаты дорогостоящей лицензии. Размещение в Docker даёт широкие возможности для масштабирования, поскольку дополнительные экземпляры Docker могут быть активированы достаточно быстро по мере возрастания нагрузки и так же быстро выгружены в случае простоя, освобождая серверные ресурсы.

В качестве платформы для веб-сайта взяли ASP.NET MVC [5], в первую очередь из-за

удобства и прозрачности конструирования. Имея модель данных, первые базовые страницы можно получить на основе шаблонов буквально в несколько кликов. В то же время, возможности кастомизации практически безграничны.

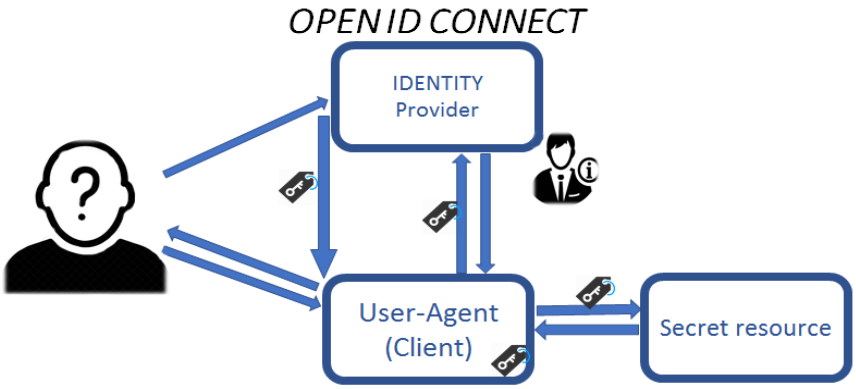

Рисунок 1 - Прохождение аутентификации и авторизации

Естественным связующим звеном ASP.NET MVC и MS SQL является Entity Framework в качестве ORM.Технология Code First [7] позволяет обновлять модель данных в коде и автоматически приводить БД в соответствие с этими изменениями при помощи механизма миграций.

Для передачи данных от локальных источников (в садоводствах) на сайт и в БД системы, была задействована технология WCF (Windows Communication Foundation) [10], представляющая собой простую в обращении, но сложную в устройстве реализацию концепции RPC (remote procedure call) или RMI (remote method invocation). Задача передачи данных суточного потребления электроэнергии от службы-репликатора в садоводстве на сервер системы, в соответствии с определённым

контрактом, была решена благодаря WCF. Суть технологии в том, что достаточно определить контракт и прописать в конфигурационных файлах информацию связывания (в простейшем случае – имя сервиса, протокол и контракт на сервере, а на клиенте – абсолютный URI сервиса (так называемый end-point), протокол и контракт, и всю остальную работу по организации прослушивания и обработки вызовов на серверной стороне, созданию прокси и осуществлению вызова метода на клиентской – возьмёт на себя инфраструктура.

Существуют два способа создания клиентских классов-обёрток (proxy). С помощью утилиты Svcutil.exe, получающей метаданные с сервера, либо с использованием разделяемой dll с контрактами и системной фабрики ServiceClient. Последний вариант является более аккуратным, вследствие отсутствия необходимости перегенерации proxy при обновлении контракта – достаточно обновить библиотеку. Им и воспользовались.

Обязательным атрибутом системы, хранящей и дающей доступ к разнообразной конфиденциальной информации, является надежная аутентификация и жестко структурированная авторизация - то есть, разграничение полномочий. Для этих целей будем использовать, пожалуй, самую современную спецификацию в данной сфере: OPEN ID Connect [9]. В простейшем случае одного MVC-приложения, выделенный контроллер в нём же будет выступать в роли Identity provider, но при желании, мы сможем легко переключиться на внешнего Identity provider, обеспечив единый вход через аккаунт Google, VK и т.п., а также задействовать двухфакторную аутентификацию. Рассмотрим работу протокола по простейшей схеме implicit flow. В рассматриваемом случае все ресурсы находятся в пределах одного сайта, поэтому identity provider и secret resource (защищенный ресурс) находятся в пределах одного приложения (рис.1).

- Не аутентифицированный пользователь открывает в браузере приложение и просит открыть личный кабинет.
- Приложение в браузере перенаправляет пользователя на страницу входа (Authorization endpoint согласно терминам протокола).
- После ввода логина-пароля пользователь (уже аутентифицированный) вновь перенаправляется в защищенную область.
- Серверный ответ содержит заказанные токены (только Identiy или дополнительно Access).

- Теперь пользователю доступен защищенный ресурс.

Для подключения OIDC провайдера, нам потребуется использование ещё одной технологии, OWIN [11], о которой расскажем отдельно.

Рассмотрим работу конвейера обработки HTTP-запроса в ASP.NET приложении под IIS. Прохождение запроса формирует определённое количество событий, каждое из которых может быть обработано набором предварительно сконфигурированных HTTPModule-ей. На заключительной стадии запрос обрабатывается соответствующим HTTPHandler-ом (в нашем случае MVC Handler) и клиенту возвращается HTTP Response. Из всей цепочки событий, нас особенно интересует AuthenticateRequest, где в контексте запроса появляется информация, идентифицирующая пользователя, отправившего запрос, как результат работы, например, FormsAuthenticationModule.

В случае с OWIN'ом, конфигурация HTTPModule-ей через секцию system.webServer конфигурационного файла приложения не используется. Вместо этого приложение конфигурируется прямо в коде метода Configure(IApplicationBuilder) класса Startup, путём регистрации в приложении цепочки последовательно вызываемых «звеньев» (Middleware см. рис. 2).

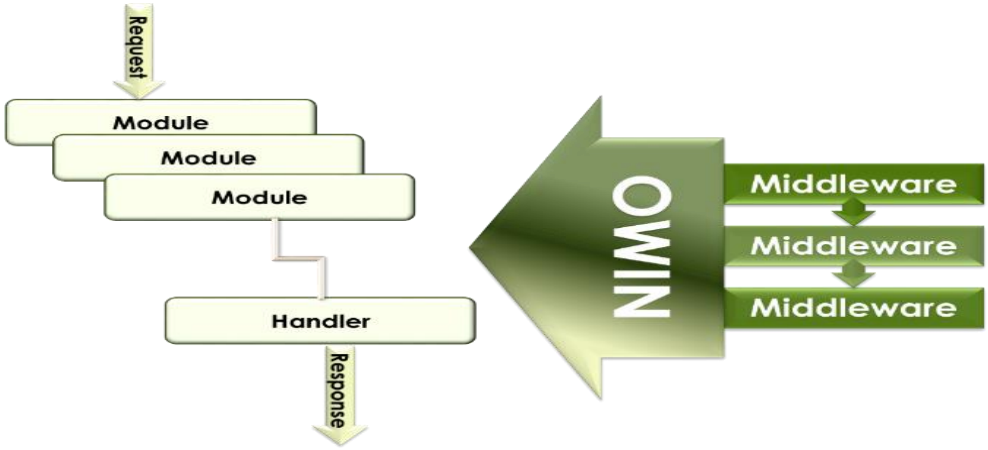

Рисунок 2 - Отработка запроса путем регистрации в звеньях Middleware.

Современные реализации протоколов аутентификации (но не только) выполнены именно в виде таких Middleware.

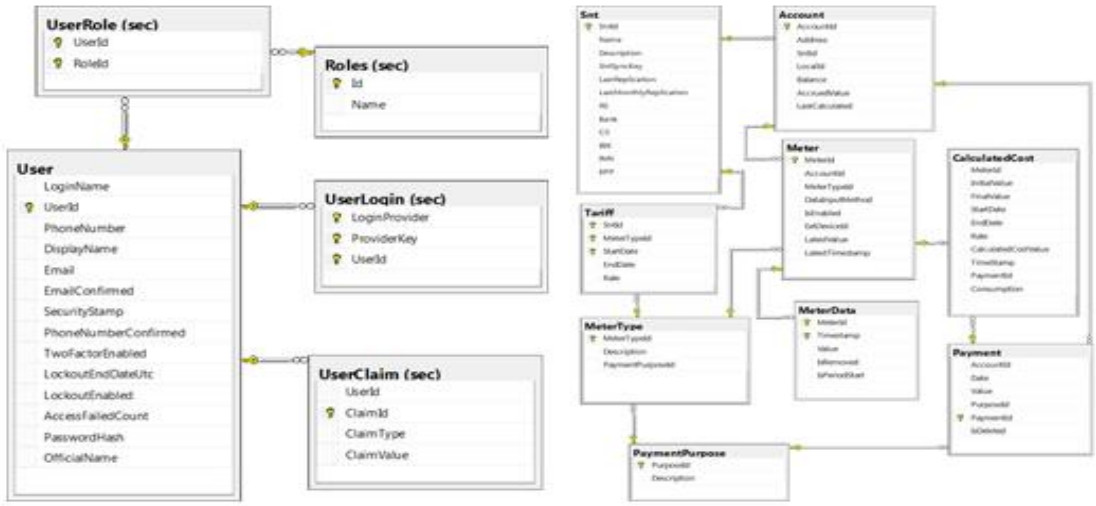

Рисунок 3 - Схема базы данных.

Реализация спецификации OWIN для .NET и IIS (проект Katana). Встраивает вызов цепочки Middleware по событию PreHandlerExecute стандартного цикла обработки, поэтому, к моменту вызова MVC Handler'a, наш реквест уже будет содержать в контексте идентификационные данные пользователя.

Если же необходимо вызвать тот или иной Middleware на более ранней стадии в pipeline, для этого достаточно проставить в коде Middleware'a специальный маркер.

Такое двуединство подходов возможно только при использовании библиотек OWIN в версиях .Net framework 5.1 [6] и старше. При переходе же к .NET CORE модель событий, модулей и хендлеров окончательно уступает место конфигурируемой в коде цепочке Middleware.

Практическая часть работы и оптимизация системы

База Данных логически разделена на две схемы (рис.3).

Это схема sec - security и схема dbo для хранения основной части данных, содержимое схемы sec генерируется автоматически с помощью технологии Code First [7] при использовании шаблона MVC. Тип ключа по умолчанию GUID был заменён на int, сценарии использования системы не предполагают регистрацию более 2 млрд. пользователей, а работа с целочисленным ключом эффективнее чем с ключом типа GUID.

Рассмотрим схему подробнее:

- Таблица Snt содержит данные о садоводствах, информацию для идентификации сервиса, передающего показания из технологической подсистемы: «Меркурий», «Матрица»-«ADDAX» [4] и др., время последней репликации данных садоводства.
- Таблица Account содержит информацию о садовых участках, такую как принадлежность к определенному СНТ, адрес, а также состояние индивидуального абонентского счёта и время последней операции пересчёта показаний приборов учёта.
- Таблица Meter хранит информацию о привязанные к определённому садовому участку приборах учёта: номер прибора, последние переданные показания и время их получения, тип прибора учёта и способ получения показаний (допустимые значения: автоматический, ручной со снятием уполномоченным лицом, ручной со снятием абонентом).

В рамках тестирования системы было произведено наполнение БД сгенерированными данными, призванное сымитировать работу системы спустя продолжительное время сбора и обработки показаний.

Произведено исследование обработки запросов средствами SQL Profiler.

Важно своевременно фильтровать данные. Как следствие, при наличии в запросе операторов DISTINCT и JOIN [12], желательно применить первый на как можно более ранней стадии. Отсутствие необходимых индексов мешает встроенному оптимизатору делать свою работу хорошо, и для того, чтобы не забыть построить нужные индексы поможет наблюдение за специальными системными представлениями:

sys.dm_db_missing_index_group_stats
sys.dm_db_missing_index_details
sys.dm_db_missing_index_groups

Хочется отметить следующие результаты: после подобной модификации кода в исследовании время выполнения запроса сократилось в пятьдесят (!) раз на семи миллионах строк тестовых данных.

Для передачи данных от распределенных технологических подсистем садоводства в центральное хранилище данных принята модель односторонней репликации предварительно агрегированных данных.

Алгоритм заключатся в следующем: сервис репликации при каждой активации осуществляет выборку последних данных каждого прибора учета, а кроме того, независимо, с определенной периодичностью осуществляется выборка определенных граничных показаний последних на каждый месячный интервал с отсечением последних двенадцати месяцев. Подобный подход позволяет с гарантированно высокой практической вероятностью переносить в основное хранилище данных информацию обо всех величинах, необходимых для использования при осуществлении платежей.

А теперь кратко рассмотрим веб-интерфейс, то есть само MVC-приложение.

В табличной форме представлены основные показатели каждого участка, такие как дата получения показаний, дата последнего внесённого платежа, сумма задолженности или переплаты. Для удобства пользователя, платежи, внесённые не более месяца назад от текущей даты, раскрашены зелёным цветом, а более двух месяцев – красным. Таким образом, в сочетании с возможностью сортировки по столбцу «Баланс», администратор может с лёгкостью выявить «злостных неплательщиков».

Приложение не использует клиентских javascript-фреймворков, таких как Angular или, являясь классическим MVC, однако (как, например, форма ввод платежа на форме списка

участков) выполнены в виде асинхронных всплывающих модальных окон, с использованием простейшей библиотеки JQuery. Этого достаточно, чтобы избежать построения страницы при многократном выполнении рутинных операций.

Для возможности осуществления платежей непосредственно на счет садоводств реализована функция печати банковских поручений с QR – кодом по ГОСТ Р 56042-2014 «Стандарты финансовых операций. Двумерные символы штрихового кода для осуществления платежей физических лиц» [1].

Для чего была использована библиотека ZXing.Net [13], представляющая видоизмененный и усовершенствованный перевод с языка Java на .Net.

Для удобства привилегированных пользователей был создан раздел статистика. Здесь можно визуализировать, например, данные по садоводству за год посмотреть размер задолженности. Для создания наглядных диаграмм – были выбрана библиотека APEX CHARTS [3]. Это современная JavaScript библиотека с открытым исходным кодом для создания красивых диаграмм (рисунок 4), переведенная с языка Java на .Net.

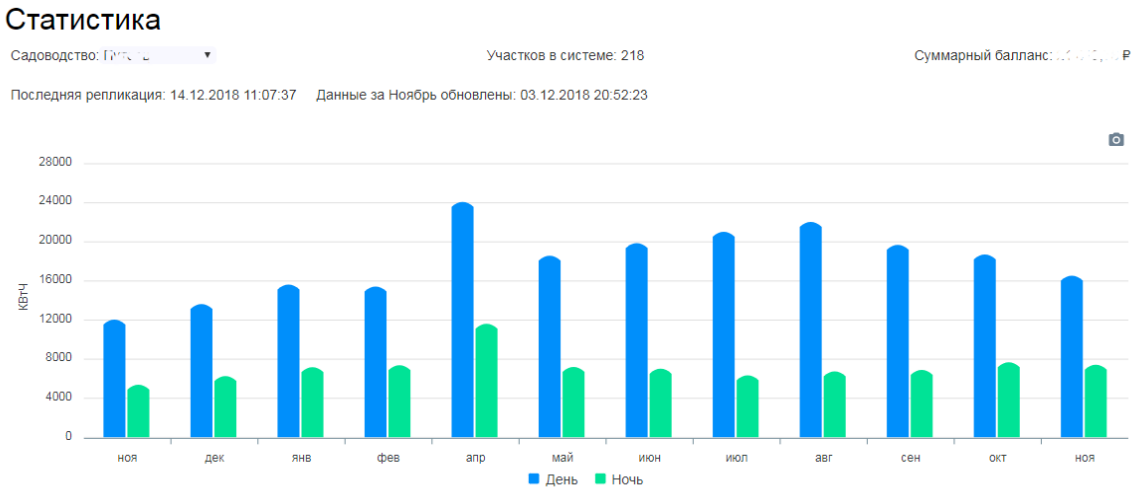

Рисунок 4 - Диаграмма "Статистика".

Необходимо упомянуть, что формат SVG имеет все преимущества XML: он хорошо сжимается, а так как XML - текстовый документ, имеющий регулярную структуру, то SVG легко взаимодействует с HTML и XHTML документами. SVG это открытый стандарт, прекрасно работающий во всех современных браузерах.

Заключение

В статье проанализировано взаимодействие служб выбранного стека программ в сочетании с использованным оборудованием, которое показало превосходный результат эффективной работы автоматизированной комплексной системы. В планах развития были рассмотрены новейшие технологии.

В дальнейшем разработанный продукт планируется дополнить возможностью сбора данных с «умных» счетчиков воды, газа и тому подобного оборудования. Система находится в состоянии непрерывного развития и эксплуатируется тысячами пользователей.

References:

1. (2014). GOST R 56042-2014 *Standarty finansovih operazyi. Dvumernie symvoly shtrihovogo coda dlya osuschestvleniya platezhey fizicheskyh liz. Standards of financial transactions. Two-dimensional barcode symbols for payments by individuals* [Text]. – Vved. 2014-09-01. M.: Standartinform.
2. (2013). *Schetchiky elektricheskoi energii odnofaznye seryi NP5 /Tehnicheskoe opisanie I rukovodstvo po expluatazii*. ADDM.410061.101, Moskva: OOO "Matriza", versya documenta 5.4. Retrieved January 14, 2019, from http://matrica.ru/images/manuals/TD%20AIIS%20KUE.pdf
3. (2019). *APEXCHARTS*. Retrieved January 14, 2019, from https://apexcharts.com
4. (2019). *ADD GRUP*. Retrieved January 14, 2019, from https://addgrup.com
5. (2019). *ASP.NET MVC 5*. Retrieved January 14, 2019, from https://metanit.com/sharp/mvc5
6. (2019). *Entity Framework Tutorial*. Retrieved January 14, 2019, from http://www.entityframeworktutorial.net/what-is-entityframework.aspx
7. (2019). *Entity Framework Tutorial Code First*. Retrieved January 14, 2019, from http://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx
8. (2017). *Microsoft SQL Server*. Retrieved January 14, 2019, from https://docs.microsoft.com/ru-ru/sql/sql-server/sql-server-technical-documentation?view=sql-server-2017
9. (2019). *OpenID Foundation*. Retrieved January 14, 2019, from https://openid.net
10. (2010). *WCF Overview*. Retrieved January 14, 2019, from https://relentlessdevelopment.wordpress.com/2010/03/30/wcf-overview
11. (2014). *What's this OWIN stuff about? Anders Abel*. Retrieved January 14, 2019, from https://coding.abel.nu/2014/05/whats-this-owin-stuff-about
12. (2019). *What You Need to Know about Adaptive Joins over Rowstore. Ben-Gan I.* Retrieved January 14, 2019, from http://www.itprotoday.com/microsoft-sql-server/what-you-need-know-about-adaptive-joins-over-rowstore
13. (2019). ZXing.Net. Retrieved January 14, 2019, from https://github.com/micjahn/ZXing.Net