УДК 004.657

# РАЗРАБОТКА СИСТЕМЫ ИНТЕГРАЦИИ ДОКУМЕНТООБОРОТА ПРЕДПРИЯТИЯ С ОБЛАЧНЫМ СЕРВИСОМ «МОЙ СКЛАД»

Ерзунов Г. Е., студент гр. ИТб-181, IV курс
Рогов Д. Е., студент гр. ИТб-181, IV курс
Научный руководитель: Алексеева Г. А., старший преподаватель
Кузбасский государственный технический университет
имени Т. Ф. Горбачева
г. Кемерово

Ни одно современное предприятие не может обойтись без развитой IT-инфраструктуры, для обеспечения автоматизации различных процессов деятельности. Безопасная, гибкая и надежная ИТ-инфраструктура в значительной степени помогает предприятию добиться поставленных целей и получить конкурентное преимущество на рынке. В связи с этим необходимо рассматривать автоматизацию производственных процессов не только с точки зрения автоматизации отдельных функций, но и с позиций комплексной и интегрированной автоматизации производственных процессов, направленных на повышение конкурентоспособности промышленного предприятия.

Наиболее актуальна данная проблема для таких процессов как ведение бухгалтерского учета, контроль движения товаров и продукции и т. д. Существует несколько способов автоматизации этих процессов, например предприятие может приобрести несколько рабочих станций с установленными на них прикладными решениями для ведения бухгалтерского учета. Однако подобный подход требует постоянных затрат на поддержание IT инфраструктуры предприятия, что при большом обороте будет выливаться в огромные финансовые затраты. Поэтому многие предприятия постепенно переходят на использование облачных сервисов бухгалтерского учета, так как при таком подходе нет необходимости в содержании парка рабочих станций, а также других сопутствующих затрат.

Одним из таких прикладных решений, позволяющих автоматизировать документооборот предприятия, является облачный сервис «Мой склад». Основными возможностями данного решения являются:

1. Управление торговыми точками.
2. Формирование аналитических отчетов о продажах и выручке.
3. Управление закупками: формировать заказы поставщикам, проводить приемку товаров, делать возвраты поставщикам.
4. Заниматься продажами: формировать счета покупателей, возвраты, отгрузки.
5. Складские операции: поступления, учета, перемещения, списания, учета по серийным номерам.

6. Ведение производства: составление технологических карт, планирование технологических операций.

Но у сервиса «Мой склад» существуют и недостатки, например в нем отсутствует возможность создания несколько заказов поставщикам за один раз, что значительно увеличивает время необходимое для документооборота предприятия. По этой причине целесообразно разработать систему интеграции с данным облачным сервисом, в которой была бы решена выявленная проблема. В данной статье представлен вариант разработки системы интеграции с облачным сервисом «Мой склад», позволяющей в значительной мере сократить временные издержки на ведение документооборота предприятия.

На первом этапе данные о заявках покупателей из сервиса «Мой склад» извлекаются в локальную базу данных предприятия. Далее эти данные анализируются в разработанной системе интеграции, и на их основании формируются заказы поставщикам. В процессе обработки информации пользователь имеет возможность взаимодействовать с ней с помощью WEB-интерфейса. На следующем этапе данные о заказах поставщикам отправляются в локальную базу данных предприятия. Откуда попадают в облачный сервис «Мой склад».

Помимо этого, в разработанном приложении присутствует возможность проверки остатков необходимых материалов у поставщика на складе, а также возможность составления плана работ, что позволяет уменьшить временные издержки на организацию деятельности предприятия, за счет автоматизации менеджмента по управлению персоналом. Диаграмма, представляющая общий принцип работы приложения представлена на рисунке 1.

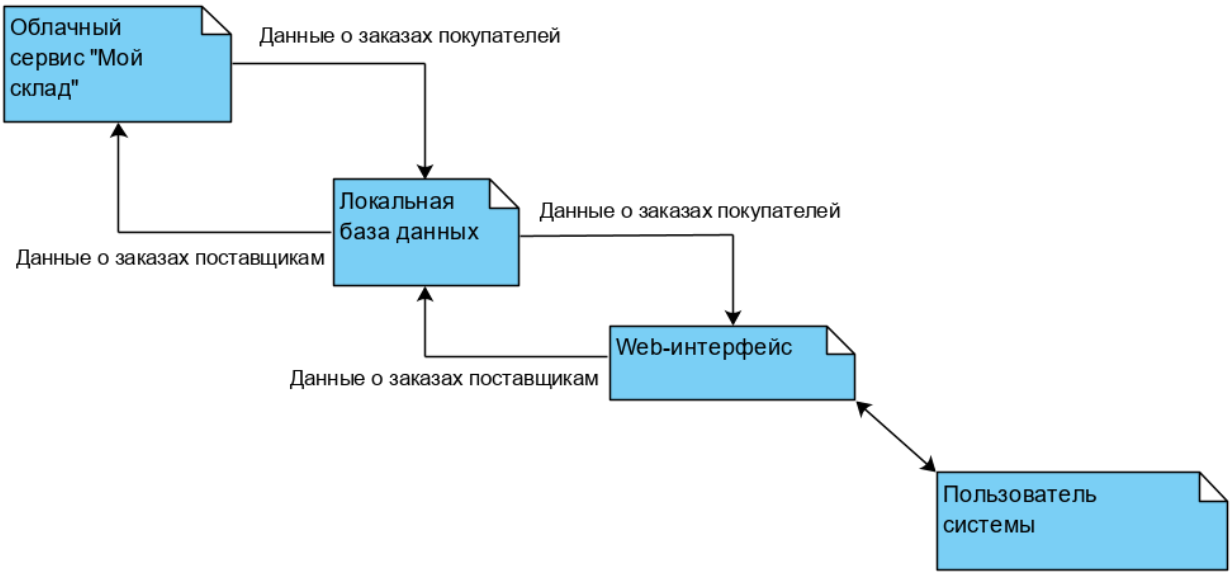


Рисунок 1 – Диаграмма, отображающая общий принцип работы системы

Диаграмма деятельности процесса формирования заказа поставщику представлена на рисунке 2.

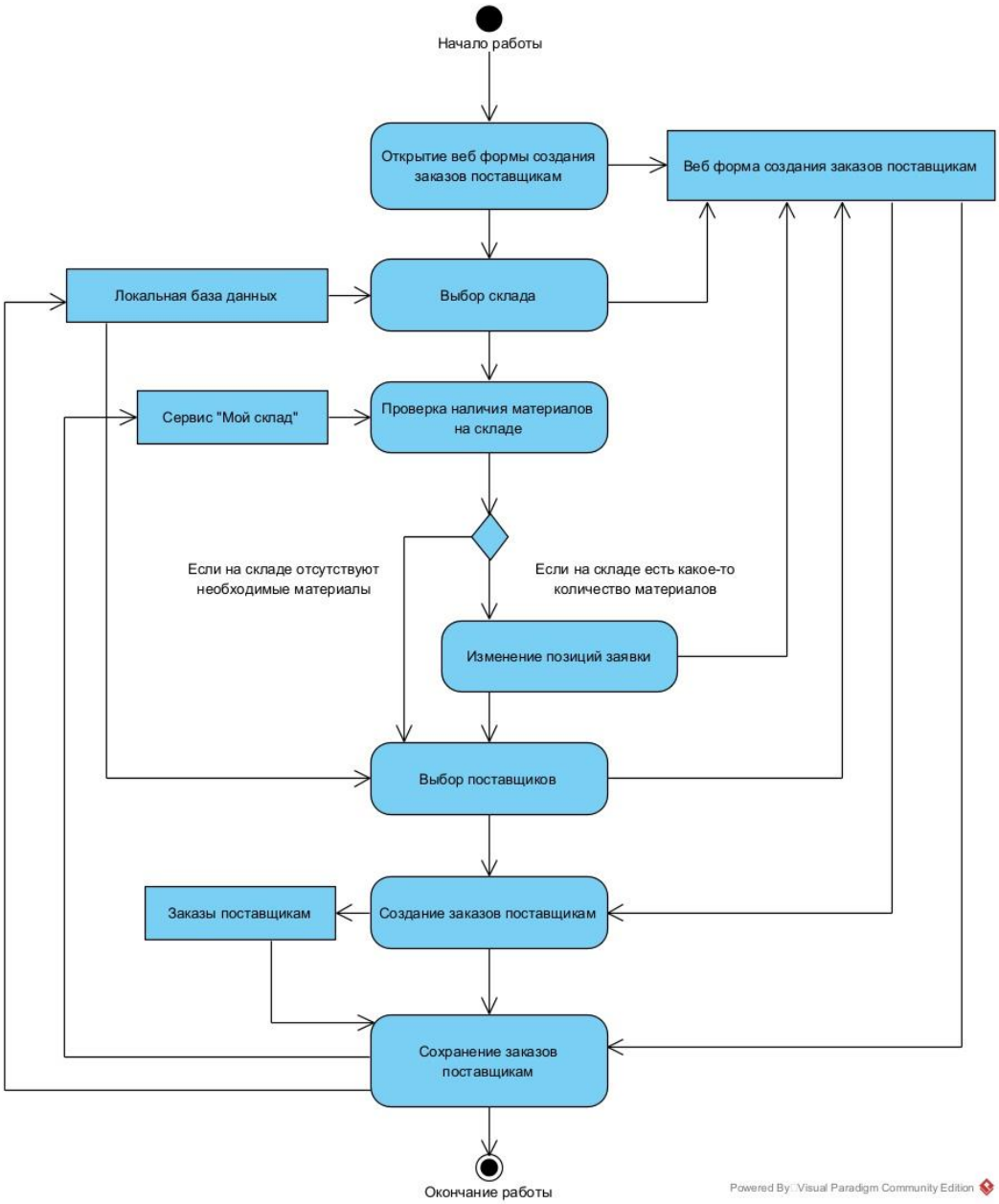


Рисунок 2 – Диаграмма деятельности для процесса создания заказов поставщикам

Диаграмма деятельности процесса составления плана работ представлена на рисунке 3.

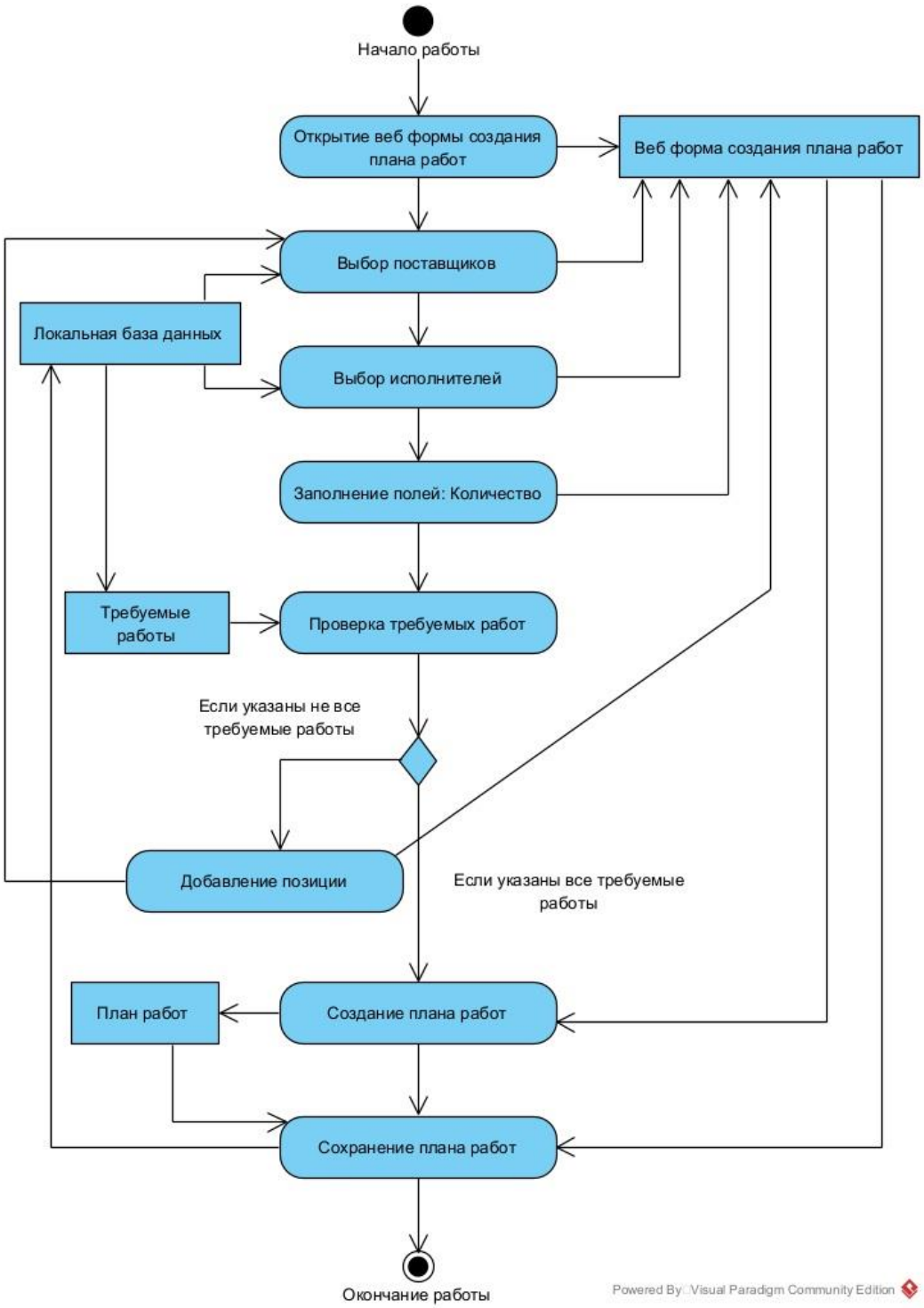


Рисунок 3 – Диаграмма деятельности для процесса составления плана работ

Данное прикладное решение строится на классической архитектуре MVC, которая позволяет условно разбить приложение на три отдельных компонента, а именно контроллер, интерпретирующий действия пользователя, оповещая модель о необходимости изменений, модель, которая непосредственно взаимодействует с данными и представление, отображающее пользователю результат работы модели. Необходимо отметить, что в модели содержаться классы и методы, обращающиеся напрямую к локальной базе

данных и облачному сервису «Мой склад», они могут как принимать какую-либо информацию, так и отправлять.

С программной точки зрения, созданное приложение строится на классической клиент-серверной архитектуре. Серверная часть системы интеграции написана на языке программирования C#, в интегрированной среде разработки Visual Studio. Эта часть служит для обмена информацией между сервисом «Мой склад» и информационной системой, а конкретно для выгрузки и загрузки данных о заказах. Также в систему входит база данных (БД), использующаяся для хранения информации, связанной с документооборотом предприятия, созданная в СУБД Microsoft SQL Server. Клиентская часть представлена в виде web-интерфейса, написанного с использованием html, css, JavaScript и позволяющего взаимодействовать с системой интеграции, а именно получать данные о заказах покупателей, анализировать их, и т.д.

Разработанное приложение осуществляет передачу данных между локальной базой данных и облачным сервисом «Мой склад», посредством REST JSON API, с использованием следующих библиотек:

1. Newtonsoft.Json используется для парсинга Json-объектов.

System.Data.SqlClient позволяет взаимодействовать с локальной базой данных предприятия.

2. System.Data предоставляет доступ к классам представляющим архитектуру ADO.NET, позволяет создавать компоненты, эффективно управляющие данными из нескольких источников данных.
3. System.Web.Configuration - Содержит классы, используемые для взаимодействия с ASP.NET.
4. System.ComponentModel.DataAnnotations Предоставляет классы атрибутов, используемые для определения метаданных для ASP.NET MVC и элементов управления данными ASP.NET.
5. System.Data.EntityFramework используется для взаимодействия с базой данных, а также представления таблиц в виде объектов.

Для того, чтобы приложение могло связаться с облачным сервисом, посредством REST API, необходимо создать HTTP-запросы, осуществляющие получение содержащихся в сервисе «Мой склад» данных о заказах, контрагентах и т.д., а также отправку и сохранение указанной информации в сервисе «Мой склад». Создание HTTP-запроса предполагает определение заголовков запроса и его тела. Заголовки содержат метаданные, необходимые для отправки и передачи HTTP-запроса. В то время как тело содержит непосредственно сам запрос, зачастую тело представлено в виде JSON файла. Листинг кода HTTP-запроса для сохранения данных в сервисе «Мой склад» представлен на рисунке 4.

```
var client2 = new RestClient("https://online.moysklad.ru/api/remap/1.2/entity/purchaseorder");
client.Timeout = -1;
//Отправка заказов поставщикам в МОЙ СКЛАД
var request2 = new RestRequest(Method.POST);
request2.AddHeader("Authorization", "Basic " + credentials2);
request2.AddHeader("Accept", "*/*");
request2.AddJsonBody(Json);
IRestResponse response2 = client2.Execute(request2);
string a2 = response2.Content;
```

Рисунок 4 – Листинг кода HTTP-запроса для сохранения данных в сервисе «Мой склад»

Информация, с помощью которой происходит загрузка\выгрузка данных из сервиса «Мой склад», имеет формат JSON. Из чего следует, что необходимо привести к этому же формату данные, которые приложение будет посылать сервису «Мой склад». Часть кода формирования JSON представлена на рисунке 5.

```
// Формирование JSON тела для HTTP запроса
SupplyOrderJSon.Root root1 = new SupplyOrderJSon.Root();
// Юр Лицо
root1.organization = new SupplyOrderJSon.Organization()
{
    meta = new SupplyOrderJSon.Meta()
    {
        href = orgId,
        type = "organization",
        mediaType = "application/json",
        metadataHref = "https://online.moysklad.ru/api/remap/1.2/entity/organization/metadata"
    }
};
// Контрагент
root1.agent = new SupplyOrderJSon.Agent()
{
    meta = new SupplyOrderJSon.Meta()
    {
        href = sendOrder.AgentHref,
        metadataHref = "https://online.moysklad.ru/api/remap/1.2/entity/counterparty/metadata",
        type = "counterparty",
        mediaType = "application/json",
    }
};
root1.name = "";
root1.code = "404";
root1.description = "Создан через страничку StorageTest.Somee.com";
root1.applicable = true;
root1.positions = new List<SupplyOrderJSon.Position>();
//Привязка к заказу покупателя
SupplyOrderJSon.CustomerOrder ord1 = new SupplyOrderJSon.CustomerOrder()
{
    meta = new SupplyOrderJSon.Meta()
    {
        href = "https://online.moysklad.ru/api/remap/1.2/entity/customerorder/"+ orderCode,
        metadataHref = "https://online.moysklad.ru/api/remap/1.2/entity/customerorder/metadata",
        type = "customerorder",
        mediaType = "application/json",
    },
};
root1.customerOrders = new List<SupplyOrderJSon.CustomerOrder>();
root1.customerOrders.Add(ord1);
var Positions = db.SupplyPositions.Where(p => p.SupplyIndex == sendOrder.SupplyId);
```

Рисунок 5 – Листинг кода формирования JSON

Отдельно необходимо отметить преимущества использования JSON для осуществления связи базы данных и облачного сервиса «Мой склад». К основным преимуществам относятся:

1. Простота в использовании — JSON API является стандартом высокого уровня, что весьма упрощает его использование.
2. Производительность — JSON формат занимает малое количество памяти, из-за чего уменьшается время отклика системы.
3. Бесплатность — библиотека JSON имеет открытый исходный код и бесплатна для использования.
4. Функционал — в отличие от XML, формат JSON способен работать с массивами данных.

Использование этого программного продукта поможет в большой степени увеличить производительность предприятия за счет сокращения временных издержек при ведении документации предприятия, сделав его гораздо более эффективным, что очень важно в реалиях современной рыночной экономики.

**Список литературы:**

1. Арно Лоре, Проектирование веб-API / Пер. с англ. Д. А. Беликова.– М.: ДМК Пресс, 2020.– 440 с. ISBN 978-5-97060-861-6

2. Мюллер, Джон Пол, Семпф, Билл, Сфер, Чак. М98 C# для чайников.: Пер. с англ. - СПб.: ООО "Диалектика", 2019. - 608 с. : ил. - Парал. тит. анг'.11. ISBN 978-5-907144-43-9

3. Троелсен, Эндрю, Джепикс, Филипп. Т70 Язык программирования C# 7 и платформы .NET и .NET Core, 8-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2018 — 1328 с. : ил. — Парал. тит. англ. ISBN 978-5-6040723-1-8.

4. Фримен, Адам. Ф88 ASP.NET Core MVC 2 с примерами на C# для профессионалов, 7-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2019. — 1008 с. : ил. — Парал. тит. англ. ISBN 978-5-6041394-3-1.