УДК 004.4

РЕШЕНИЕ ПРОБЛЕМЫ ХРАНЕНИЯ СВЯЗЕЙ МЕЖДУ СЕРВИСАМИ

Шмелев П.Ю.

Пермский национальный исследовательский политехнический университет, г. Пермь, Россия

Научный руководитель: Прозорова Л.Ю. – канд. экон. наук, доцент кафедры ИТАС

Аннотация. Интерфейс программирования приложений *(API)* – это совокупность инструментов и функций в виде интерфейса для создания новых приложений, благодаря которому одна программа будет взаимодействовать с другой. В настоящее время API получил популярность и все чаще используется в приложениях для взаимодействия систем. Данные из одной системы могут передаваться в другую, видоизменяться и отправляться дальше, образуя тем самым зависимую «цепочку». В случае, если изменить поле в начале «цепочки», то дальнейшие узлы тоже придется менять. Возникает необходимость приложения для проектировщика, которое бы показывало последствия изменений состава передаваемых полей.

Ключевые слова: *API, APIManagement*, хранение связей, проектирование.

Введение. Современный рынок постоянно наполняется большим количеством *IT*-проектов. Для корректной работы внедряемых проектов необходимо интегрироваться друг с другом. С целью удобного взаимодействия проектов создан такой инструмент, как *API. API* – это веб-интерфейс для программного обеспечения. *API*, независимо от типа, прежде всего является интерфейсом: точкой, где две системы, субъекта, организации и т.д. встречаются и взаимодействуют [1].

Описание проблемы. С помощью *API* сервисы могут обмениваться данными независимо от того, какой язык программирования использует тот или иной сервис. При обмене данные могут состоять из нескольких полей. Кроме того, полученные данные из одного сервиса могут преобразовываться и передаваться в другой, образуя цепочку, в которой изменение первого элемента повлечет за собой изменение остальных. Опишем на простом примере. Допустим, имеется у нас начальная точка «Мобильное приложение». В нем пользователь заполняет свои персональные данные. Эти данные, включая идентификатор пользователя отправляются в сервис «Автовладельцы», где создается карточка на пользователя и присваивается Идентификатор автовладельца. «Автовладельцы» совершает запрос и передает данные, включая связку «Идентификатор автовладельца и идентификатор пользователя» в *WMS* (систему управления складом). Используя данное приложение, охранник на складе может идентифицировать данного человека и автовладельца, которого он представляет.

А теперь предположим, что компания решил изменить способ идентификации и поменять тип поля «Идентификатор пользователя» со строки на число. Порой данная цепочка может оказаться настолько длинной и не одиночной, что проектировщик может упустить звено, что приведет к критическим ошибкам (например, задержка сдачи проекта). Следующим шагом в развитии использования сервисов является прогнозирование изменений на этапе проектирования при замене того или иного поля. Это во многом ускорит проектирование схемы взаимодействия с сервисами и позволит уменьшить количество ошибок. Так, например, если исключить (или изменить свойства поля), то необходимо, чтобы система сообщала пользователю о нарушении цепочки передачи данных, как показано на рисунке 1 ниже.

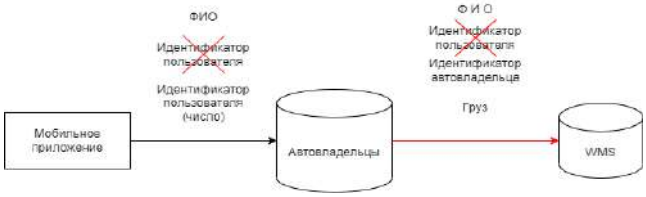

Рисунок 1 – Пример нарушения передачи данных по цепочке

Рассмотрение готовых решений. Практическая значимость заключается во внедрении системы для работы сотрудников компании ООО «Умная Логистика» при проектировании разрабатываемых систем, использующих интеграцию с другими сервисами. Существуют системы, предоставляющие функции *API Management*. *API Management* – это процесс создания и публикации программных интерфейсов веб-приложений *(API)*, обеспечения соблюдения их политик использования, контроля доступа, поддержки сообщества подписчиков, сбора и анализа статистики использования и отчетности о производительности. С другой стороны, *API Management* – это набор инструментов и сервисов, которые позволяют разработчикам и компаниям создавать, анализировать, применять и масштабировать интерфейсы *API* в безопасных средах. Среди достоинств данных систем можно отметить: аутентификация - система позволяет пройти аутентификацию, в том числе и через сторонние службы; управление трафиком – система регулирует входящий и исходящий трафик *API*; мониторинг *API*-система может помочь в мониторинге запросов/ответов клиента; преобразования – система позволяет преобразовать запросы/ответы *API*. К минусам можно отнести: увеличение *Latency* – шлюзу необходимо время для обработки запросов/ответов согласно настроенным политикам; Данные системы достаточно дорогостоящие и в комплекте поставки имеют множество других функций. Предпосылками для разработки новой системы хранения связей явились недостатки существующих систем, а именно: системы хранения встраиваются в проект, что ведет к дополнительным затратам и использованию лишних вычислительных мощностей; подключение избыточных функций, которые не используются, но их существование ведет к дополнительным затратам. [2].

Проектирование новой системы. Перечисленные выше факторы способствовали принятию решения о разработке новой системы. Принцип работы «как должно быть» данной системы описан ниже в нотации *IDEF0* на рисунке 2.

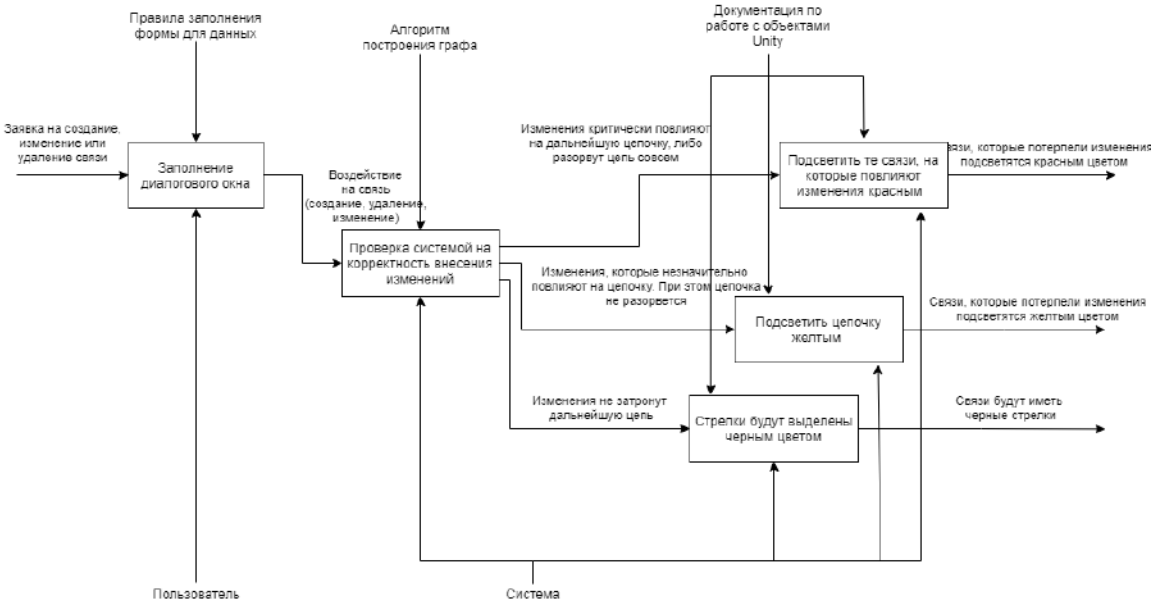

Рисунок 2 – Контекстная диаграмма работы системы для хранения связей *API* между сервисами

Опишем словестно данную диаграмму. В качестве входного параметра выступает заявка на создание, изменение или удаление связи. В первом процессе участвует Пользователь. Используя полученную заявку и правила заполнения формы для данных, он настраивает информацию в диалоговом окне свойства связи. Далее эту информацию принимает Система. Она, используя алгоритм проверки связей, проверяет цепочку связей на влияние внесенных изменений. В случае, если изменения критически повлияют на цепочку связей и дальнейшее использование будет невозможным, то эти связи, которые будут нарушены, выделятся красным цветом. В случае, если изменения не критически повлияют на цепочку, и данные будут передаваться без ошибок, то связи, которые потерпят изменения выделятся желтым

цветом. В случае, если изменения никак не повлияют на цепочку – связи будут иметь черный цвет. На выходе будем иметь графическое представление схемы связей *API* между сервисами. Перед началом разработки заказчиком были поставлены следующие требования: возможность работы нескольких сотрудников одновременно над одним проектом в облаке; интуитивно понятный интерфейс; наличие тонкого клиента, т.е. для работы с системой не устанавливается ПО; разработанная система не должна занимать много места. Так же был предоставлен прототип того, как должна выглядеть разрабатываемая система, которая представлена на рисунках 3 и 4 ниже.

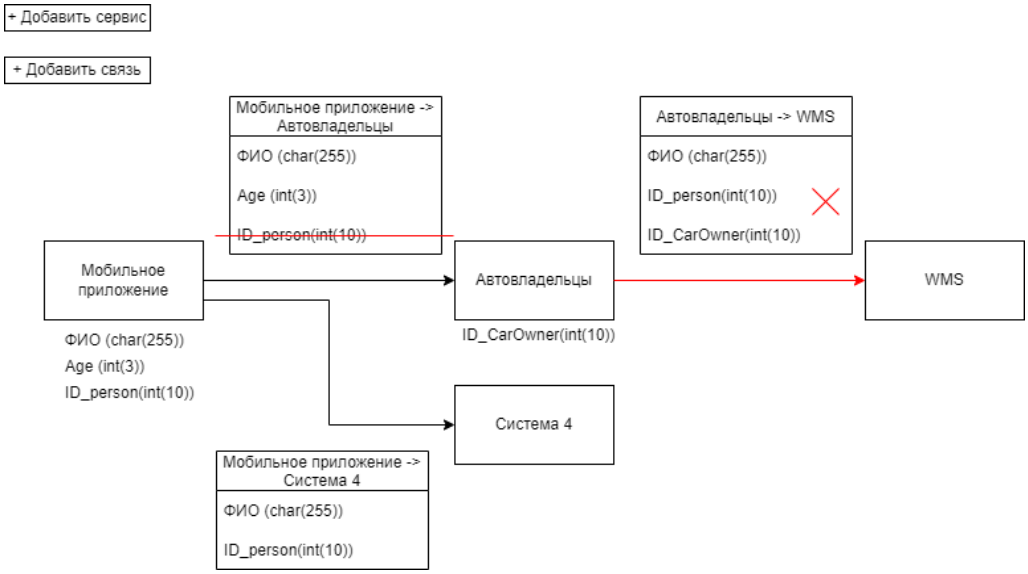

Рисунок 3 – Прототип интерфейса разрабатываемой системы

При нажатии на кнопку «Добавить связь» должна открываться форма заполнения данных о связи, которая изображена на рисунке 4. При нажатии на кнопку «Добавить сервис» будет открываться форма, где будет содержаться только одно поле для заполнения – Наименование сервиса.

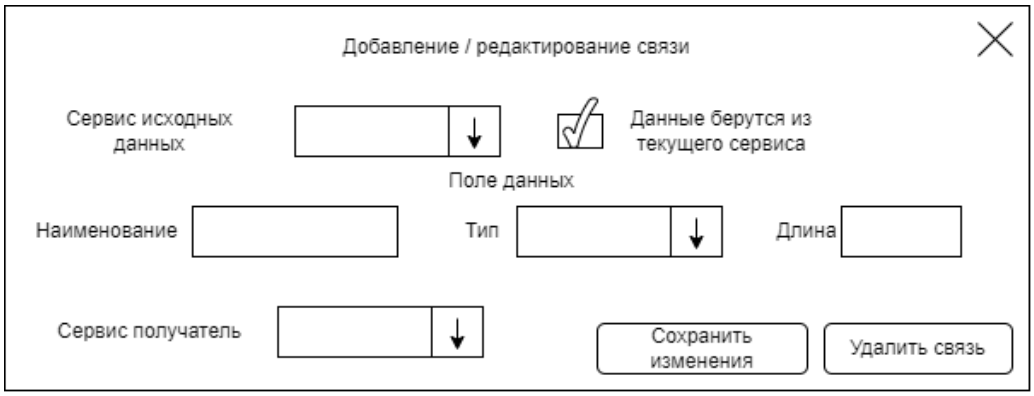

Рисунок 4 – Прототип интерфейса меню добавления / изменения связи

Исходя из перечисленных требований, для разработки была выбрана платформа *Unity* с использованием языка программирования *C#*. Выбранная платформа позволит реализовать такие возможности, как: скомпилированный проект можно разместить на серверах, куда пользователи смогут подключиться для работы без установки дополнительного ПО; готовое приложение на *Unity* будет легче, по сравнению с *Unreal Engine*.

Заключение. Проведенный анализ позволил сделать выводы о том, что эффективнее будет разработать собственную систему по хранению связей *API*, чем встраивать готовые решения.

Список литературы

1. *Арно Лоре. Проектирование веб-API // пер. с англ. Д. А. Беликова. М.: ДМК Пресс, 2020. 440 с.*
2. *Что такое системы API Management // Ресурс для публикации статей и работ IT-специалистов. [Электронный ресурс]. – 2021. – Режим доступа: https://habr.com/ru/company/X5Group/blog/543324/. – Дата доступа: 29.01.2022.*

UDC 004.4

THE MAIN ISSUES OF DEVELOPING STORAGE SYSTEMS FOR LINKS BETWEEN SERVICES

Shmelev P.Y.

Perm National Research Polytechnic University, Russia, Perm

Prozorova L.Y. – Candidate of Economic Sciences, Associate Professor of the Department of Information Technologies and Automated Systems (ITAS)

Annotation. The Application Programming Interface (API) is a set of tools and functions in the form of an interface for creating new applications, thanks to which one program will interact with another. Currently, the API has gained popularity and is increasingly used in applications for system interaction. Data from one system can be transferred to another, modified and sent further, thereby forming a dependent "chain". If you change the field at the beginning of the "chain", then further nodes will also have to be changed. There is a need for an application for the designer that would show the consequences of changes in the composition of the transmitted fields.

Keywords: API, API Management, link storage, design.