МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Бизнес-информатика
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка единой API-платформы для автоматизации бизнес-процессов компании (на примере ООО «IT Consult»)

Студент А.Н. Белов _____
(И.О. Фамилия) (личная подпись)

Руководитель канд. пед. наук, доцент, Т.А. Агошкова
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

С. 66, рис.37, табл.5, лит. 27 источников, 1 приложение.

API-ПЛАТФОРМА, ПРЕДМЕТНАЯ ОБЛАСТЬ, ИНФОРМАЦИОННАЯ СИСТЕМА, МОДЕЛИРОВАНИЕ, ПРОЕКТИРОВАНИЕ, ТЕХНОЛОГИИ, БИЗНЕС-ПРОЦЕСС.

Разработана единая API-платформа для автоматизации бизнес-процессов компании (на примере ООО «IT Consult»).

Дано описание ООО «IT Consult». Проведено функциональное моделирование предметной области, которое включало: технико-экономическую характеристику ООО «IT Consult»; концептуальное моделирование предметной области; выбор технологии моделирования бизнес-процессов ООО «IT Consult»; моделирование бизнес-процессов ООО «IT Consult» для постановки задачи автоматизированного варианта решения; разработку и анализ модели бизнес-процесса «Как есть»; обоснование необходимости разработки API-платформы; анализ существующих API-платформ для автоматизации бизнес-процессов. После этого была поставлена задача на разработку проекта API-платформы и разработана модель бизнес-процесса «Как должно быть». Осуществлено логическое проектирование API-платформы для ООО «IT Consult». При этом был создан проект структуры и функционала API-платформы для ООО «IT Consult», определено информационной обеспечение и создана концептуальная и логическая модель базы данных для API-платформы для ООО «IT Consult», определены требования к аппаратно-программному обеспечению проектируемой системы.

Проведено физическое проектирование API-платформы, разработана физическая модель базы данных, выбрана архитектура платформы, выбраны технологии реализации и СУБД системы, разработана схема взаимодействия модулей, описаны возможности системы, проведено тестирование программного проекта.

Оглавление

Введение

Актуальность исследования. Новые технологии вносят существенный вклад в развитие современного бизнеса, при этом, можно сказать, что в некоторых видах бизнеса использование новейших технологий является основополагающим фактором успеха деятельности компании. Информатизация предприятий повышает их конкурентоспособность, повышает эффективность бизнес-процессов, автоматизирует множество операций, которые осуществляют сотрудники предприятий.

На данном этапе развития информационных технологий практически не осталось компаний, которые не используют технологии хранения, передачи, или обработки данных. Это имеет связь с широким распространением компьютерной техники, а также средств передачи данных (мобильная и кабельная связь). Возникающие потоки информации на основе введения информационно-коммуникационных технологий — это объективное отражение реальных процессов в организациях.

Новое значение приобретают возможности большей прозрачности и свободного обмена информацией, которые обеспечиваются за счет API, которые рассматриваются «не только как технологический инструмент, но и как возможность создания цифровых платформ бизнеса — более гибких и адаптивных к происходящим изменениям, позволяющим быстро и эффективно масштабировать любой бизнес» [9].

Формирование цифровой инфраструктуры компании необходимо для развития и масштабирования бизнеса, в частности обеспечения роста конкуренции и повышения доступности, качества и ассортимента товаров и услуг. Открытые API становятся одним из связующих элементов цифровой инфраструктуры бизнеса. Они обеспечат передачу данных между информационными системами различных структур и подструктур бизнеса, используя стандартизированные подходы и технологические решения. Кроме того, использование открытых API позволит существенно снизить сроки и

издержки на запуск новых услуг, а упростит их последующее развитие и масштабирование, повысит конкурентоспособность предприятия [1]. Наличие API позволит интегрировать информационную систему ООО «IT Consult» с другими программными продуктами, а также разрабатывать клиентские приложения для масштабирования бизнеса.

В связи с этим, целью работы является разработка единой API-платформы для автоматизации бизнес-процессов компании ООО «IT Consult». Для достижения цели нужно решить следующие **задачи:**

1. провести функциональное моделирование предметной области;
2. провести логическое моделирование предметной области;
3. осуществить физическую реализацию единой API-платформы для автоматизации бизнес-процессов компании ООО «IT Consult».

Объект работы: процессы автоматизации деятельности ООО «IT Consult». Предмет работы: единая API-платформа для автоматизации бизнес-процессов компании ООО «IT Consult».

Методы исследования. В процессе исследования использовались методы контент-анализа, синтеза, сравнения и обобщения, системного подхода, описательный метод, методы разработки и проектирования информационных систем, методы проектирования и реализации API, методы проектирования баз данных.

Структура работы: работа состоит из введения, трех глав, заключения, списка использованных источников, приложений. Текст работы изложен на 66 страницах, содержит 37 изображений, 7 таблиц, 27 источников литературы.

Глава 1 Функциональное моделирование предметной области

1.1 Технико-экономическая характеристика ООО «IT Consult»

ООО «IT Consult» является коммерческой организацией, созданной в организационно-правовой форме общества с ограниченной ответственностью, в соответствие с действующим законодательством Российской Федерации. Компания является юридическим лицом и строит свою деятельность на основании действующего Устава и действующего законодательства Российской Федерации: Конституции Российской Федерации, Гражданского Кодекса, Трудового Кодекса и Налогового Кодекса. Основной целью организации является получение прибыли в интересах акционеров общества. Устав является учредительным документом общества с ограниченной ответственностью.

Общество с ограниченной ответственностью «IT Consult» - является агентством по оказанию комплексных услуг, по продвижению бизнеса, а в частности: разработка и поддержка сайтов, реклама в интернете, внедрение Битрикс24, обучение в сферах бизнеса, предпринимательства, а также по разработке сайтов.

Общество вправе в установленном порядке открывать банковские счета на территории Российской Федерации и за ее пределами. Общество имеет круглую печать, содержащую его полное фирменное наименование на русском языке, а также указание на его место нахождения. Общество вправе иметь штампы и бланки со своим наименованием, собственную эмблему и другие средства визуальной идентификации.

ООО «IT Consult» отвечает по своим обязательствам всем принадлежащим ему на правах собственности имуществом. Участник имеет предусмотренные законом и учредительными документами Общества обязательственные права по отношению к Обществу».

Органами управления ООО «IT Consult» являются:

- общее собрание акционеров;
- совет директоров;
- генеральный директор (единоличный исполнительный орган).

Контроль за финансово-хозяйственной деятельностью общества осуществляет ревизионная комиссия.

Схема организационной структуры предприятия ООО «IT Consult» представлена на рисунке 3.

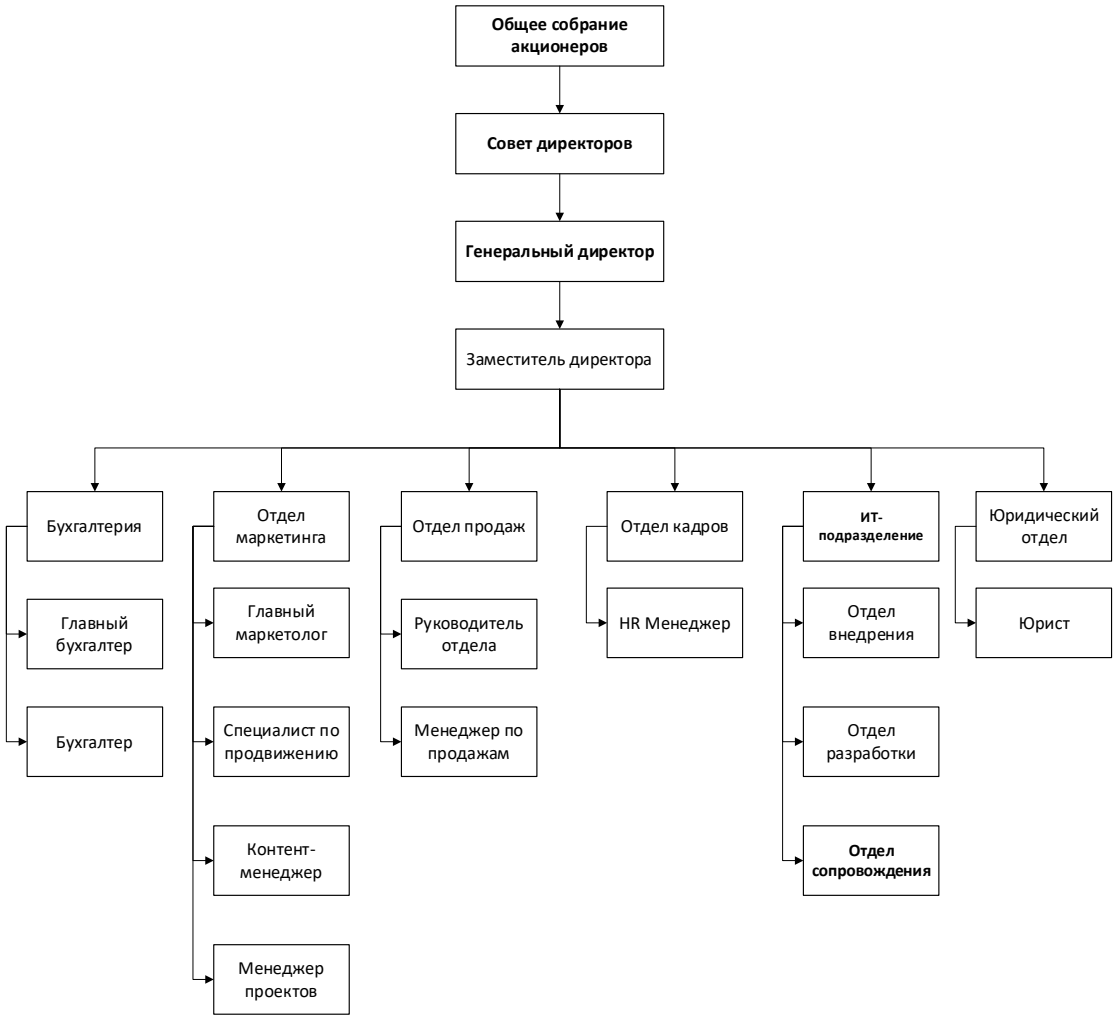

Рисунок 1 – Организационная структура предприятия ООО «IT Consult»

Из рисунка видно, что органами управления общества является: общее собрание, совет директоров и генеральный директор.

Генеральный директор без доверенности действует от имени Общества. Генеральный директор определяет позицию Общества (представителей Общества) по вопросам повестки дня общего собрания акционеров (участников) и заседания совета директоров дочерних и зависимых обществ за исключением случаев, когда в соответствии с уставом Общества такие полномочия отнесены к компетенции Совета директоров Общества.

Централизирующим звеном в последовательности продаж компании является её менеджер, обычно это стрессоустойчивый человек с высшим образованием, который легко находит общий язык с каждым клиентом и обладающий большим творческим потенциалом, его непосредственным начальником является коммерческий директор.

Ответственным за финансовую деятельность организации выступает главный бухгалтер, его главная задача – соблюдение норм законов в рамках бухгалтерской деятельности. На должность главного бухгалтера может претендовать человек с аналитическим складом ума и высшим образованием в сфере экономики, его работа состоит в ведении бухучета имущества, обязательств и хозяйственных операций, разработки и реализации процесса соблюдения рационального использования ресурсов и финансовой дисциплины.

За все вопросы закупки, маркетинга и продаж, а также формирования их планов, несет ответственность коммерческий директор.

Ответственным за руководства отделом продаж является начальник отдела, обязанности которого сводятся к контролю и координированию продаж в фирме, а также за разработку должностных инструкций для всех работников своего отдела, разработку мероприятий, направленных на расширение рынка сбыта и повышению продаж. Помимо всего этого, он занимается формированием планов и проведением обучения сотрудников отдела.

Главным для всех сотрудников профессиональным качеством является компетентность и вежливость, которыми должны обладать все сотрудники

компании, потому как работа сотрудника требует хорошего понимания потребностей клиента, а также их качественное и оперативное обслуживание.

Именно такой подход приводит к повышению уровню товаров и привлечению новых клиентов, для дальнейшего сотрудничества.

При возникновении жалоб покупателей, которые связаны с неудовлетворительным качеством товара, условиями транспортировки, отгрузки и т. д., устраняются в минимальные сроки.

Системный администратор отдела технического обеспечения решает текущие и перспективные задачи организации, связанные с ИТ. Осуществляет оперативную поддержку пользователей. Обеспечивает работоспособность или быструю замену техники. Разрабатывает предложения по формированию планов перспективного развития ИТ в организации.

Служба по внедрению выполняет функции по первичной настройке нового программного обеспечения (ПО), первичному обучению пользователей работе с новым ПО, проведению анализа данных, созданию файлов для загрузки данных в новое программное обеспечение.

Служба разработки выполняет задачи по созданию нового программного обеспечения на каждой стадии внедрения, доработки системы в условиях требований пользователей на стадии сопровождения системы, поддержке системы в рабочем состоянии, оптимизации процессов функционирования системы.

Служба сопровождения выполняет поддержку пользователей, консультирование по функциональным возможностям программы, написание технических заданий и выполнение критических операций по работе в системе.

1.2 Концептуальное моделирование предметной области

Для моделирования бизнес-процессов выбрана методология Idef0 [13]. Основным компонентом диаграммы Idef0 является блок. Блоки диаграммы

отображают работы, процессы, функции и задачи, выполняемые в определенное время.

Модели в нотации Idef0 необходимы с целью высокоуровневого описания бизнеса фирмы в функциональном плане. «Нотация DFD дает возможность отражения последовательности работ, которые выполняются по ходу процесса, и потоков сведений, циркулирующих между данными работами» [14].

Методология Idef0 использует метод функционального моделирования сложных систем, являющийся частью методологии структурного анализа и проектирования SADT (Structured Analysis and Design Technique). Idef0 позволяет «строить функциональные модели, которые описывают бизнес-процессы в виде иерархической системы взаимосвязанных функций» [13].

Нотация Idef0 используется для создания верхнего уровня модели бизнес-процессов. Подготовленную модель необходимо согласовывать с архитекторами и программистами, для подтверждения того, что структура бизнес-процессов наглядна, понятна и доступна.

Модели должны быть согласованы с ответственными специалистами организации, обладающими полной информацией по конкретному бизнес-процессу.

Бизнес-моделирование представляет собой «процесс разработки и внедрения различных бизнес-моделей организации (стратегия, бизнес-процессы, организационная структура, качество и др.) с целью формализации и оптимизации её деятельности» [24].

Моделирование бизнес-процессов — это «эффективное средство поиска пути оптимизации системы, средство прогнозирования и минимизации рисков» [23].

Моделирование бизнес-процессов позволит определить особенности существующего бизнес-процесса и на основании этих данных сформировать новые требования и разработать новую бизнес-модель «Как будет».

На рисунках 2–3 представлены функциональные модели деятельности ООО «IT Consult».

Рисунок 2 - Контекстная диаграмма деятельности (A-0)

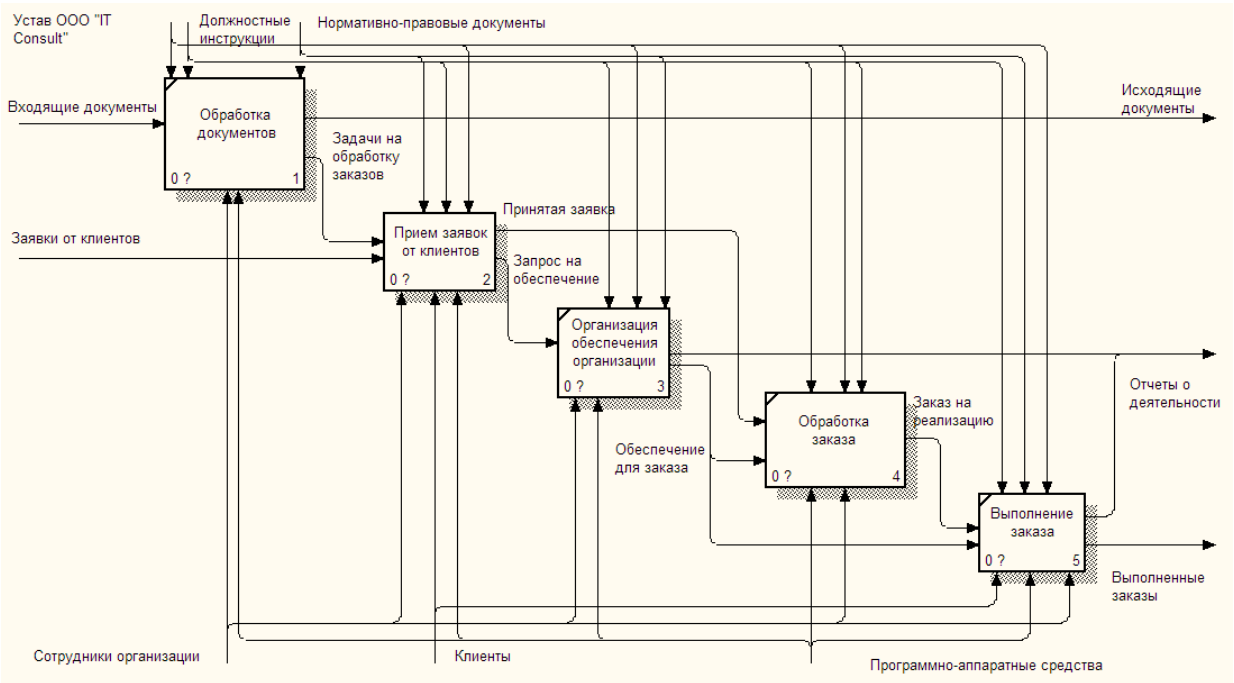

Рисунок 3 - Декомпозиция контекстной диаграммы (A0)

Как видно из рис. 2 и 3, входящими потоками являются входящие документы, заявки от клиентов, материально-товарные ценности. Управляющие потоки представлены статутом организации, должностными инструкциями и нормативно-правовыми документами. Механизмы представлены клиентами, сотрудниками организации и программно-аппаратными средствами. Выходные потоки «формируются с исходящих документов, отчетов о деятельности и выполненных заказов» [10].

Обработка заказа начинается с выбора клиентом актуальной для него услуги (товара), обрабатывает заказ менеджер по продажам. После «обработки заказа информация о нем записывается в базу данных и в процессе выполнения заказа, информация о нем берется из базы данных. После выполнения заказа создается счет представленных услуг (товаров)» [16].

Таким образом, можно сказать, что объектом, который требует автоматизации в организации ООО «IT Consult», является отдел продаж. Автоматизация отдела поможет повысить «эффективность работы менеджеров с клиентами, позволит поднять уровень продаж, упростить контроль и анализ деятельности предприятия» [9].

При анализе существующих процессов в отделе продаж можно выделить следующие недостатки:

- отсутствие автоматизации процесса учета заявок клиентов;
- отсутствие базы данных клиентов и заказов;
- отсутствие механизмов контроля деятельности отдела продаж и удобной статистики продаж;
- отсутствие систематизации в базе данных заказов;
- отсутствие возможностей масштабирования процессов компании.

Представленные недостатки снижают продуктивность работы отдела продаж и эффективность деятельности предприятия. Увеличивается количество времени, которое менеджер тратит на обработку заказов, снижается количество обработанных заказов в день.

Для решения представленных выше недостатков наиболее подходящим вариантом является создание приложения для обработки заказов, которое позволит обеспечить:

- «автоматизацию процесса учета заявок клиентами;
- механизмы контроля деятельности отдела продаж;
- систематизацию базы данных заказов;
- наличие базы данных клиентов» [16].

Более подробно опишем особенности менеджера по продажам. Менеджер по продажам должен осуществлять связь между покупателями, и производящими и торговыми организациями.

Основные задачи менеджера по продажам:

- Выполнение плана продаж.
- Управление продажами в своем сегменте.
- Управление взаимоотношениями с клиентами.
- Контроль дебиторской задолженности.

Функции менеджера по продажам:

- продажа товаров и услуг;
- ведение переговоров, выезд на встречи;
- поиск и привлечение новых клиентов;
- документооборот;
- выполнение поставленного плана;
- ведение базы клиентов;
- знание особенностей продаваемого товара;
- анализ объемов и итоговая отчетность по выполненным продажам.

На основании представленных выше данных раздела принято решение о необходимости разработки API-платформы, которая позволит повысить эффективность деятельности отдела работы с клиентами компании за счет совершенствования возможностей масштабирования бизнеса путем добавления специализированных веб-служб и API-интерфейсов к ним. API-

интерфейс позволит эффективнее масштабировать возможности интеграции с информационной системой «IT Consulting».

Цель проекта состоит в повышении эффективности деятельности компании в направлении разработки веб-служб и API для масштабирования процессов взаимодействия с клиентами.

Задачи проекта автоматизации бизнес-процессов включают:

- повышение уровня интеграции со сторонними программными продуктами;
- разработка и внедрение наборов API для управления взаимодействием с клиентами;
- снижение количества времени, которое необходимо для обработки заказа;
- повышение уровня качества обслуживания клиентов;
- снижение трудоемкости процессов обработки заказов;
- повышение надежность сохранения данных клиентов и заказов;
- совершенствование учета статистики деятельности отдела продаж;
- обеспечение взаимодействия клиентов с компанией посредством клиентского приложения (веб-, или мобильного).

В данном пункте выбрана технология моделирования бизнес-процессов ООО «IT Consult», смоделированы бизнес-процессы ООО «IT Consult» для постановки задачи автоматизированного варианта решения, разработаны и проанализированы модели бизнес-процесса «Как есть» и обоснована необходимость разработки API-платформы.

1.3 Анализ существующих API-платформ для автоматизации бизнес-процессов

При анализе существующих API-платформ для «IT Consult», которые обеспечивают автоматизацию процессов взаимодействия с клиентами следует учитывать следующие критерии:

– возможности интеграции посредством API;
– наличие удобной документации по API;
– наличие функционала взаимодействия с клиентами;
– наличие функционала учета заказов;
– возможность масштабирования;
– возможность управления контактами.

Для сравнительной характеристики API-платформ для бизнеса выбрано направление работы с клиентами, поскольку в этом направлении осуществляется прием и обработка заявок клиентов.

«Системы, которые обеспечивают такую автоматизацию, называются CRM-системы» [1]. «Это системы управления взаимоотношениями с клиентами (CRM система), представляющее собой прикладное программное обеспечение для организаций, предназначенное для автоматизации стратегий взаимодействия с заказчиками (клиентами), в частности для повышения уровня продаж, оптимизации маркетинга и улучшения обслуживания клиентов путём сохранения информации о клиентах и истории взаимоотношений с ними, установления и улучшения бизнес-процессов и последующего анализа результатов» [14]

Рассмотрим наиболее распространенные CRM-системы. В контексте темы интересуют системы, которые поставляются набором API, обеспечивающих поддержку системы взаимодействия с клиентами.

Интерфейс прикладного программирования (API) — это «программный интерфейс, который предназначен для использования другим программным обеспечением для интеграции с этим приложением» [25]. В то время как обычная программа используется пользователем компьютера (человеком), API — это программа, используемая другой программой.

JSON: API — это «спецификация того, как клиент должен запрашивать эти ресурсы для извлечения или изменения и как сервер должен на них реагировать» [19]. Он разработан, чтобы минимизировать как количество запросов, так и объем данных, передаваемых между клиентами и серверами.

Эта эффективность достигается без ущерба для удобочитаемости, гибкости или обнаруживаемости.

Следовательно, термин API будет относиться к Rest API, который соответствует спецификации JSON: API, которая предоставляет программный доступ для чтения и записи данных. Тело запроса и ответа должно использовать формат JSON.

«Одной из наиболее популярных систем взаимодействия с клиентами является AmoCRM. Это система для учёта клиентов и сделок, которая работает в режиме online. Система не требует установок и настроек – достаточно пройти регистрацию на сайте приложения и можно сразу приступать к работе» [24]. Для удобства интеграции система поставляется набором API, которые предназначены для предоставления сервисов клиентам.

Еще одной подобной системой является 1С: CRM, которая «состоит из двух связанных систем: из Управления торговлей и из CRM-системы. Большим плюсом системы является то, что разработчики сумели совместить управление торговлей и работу с клиентами» [24].

vTiger CRM – «система с открытым кодом (OpenSource), написанная на PHP. В отличие от классических CRM, позволяет автоматизировать не только продажи и поддержку, но и смежные аспекты работы — оплаты, ведение проектов, сервисные контракты, склад (ограниченно)» [27]. vTiger CRM работает на связке технологий PHP+MySQL.

SuiteCRM «построен на движке Community Edition от SugarCRM, но это не просто клон. Он добавляет ряд полезных и мощных функций в систему. Генерация счета-фактуры, и ведение лидов и контактов, и формирование отчетов и прочих документов. В системе можно предоставить клиентам возможность создавать и отслеживать свои собственные запросы» [24].

Zurmo – «настраиваемая система управления отношениями с клиентами с открытым исходным кодом (CRM) для малого бизнеса. Она предлагает приложения для управления контактами, автоматизации продаж и автоматизации маркетинга» [8].

Представленные программные средства предоставляют широкий функционал, часть которого не актуального в пределах поставленной цели, в связи с этим принято решение к самостоятельной разработке веб-приложения для обеспечения автоматизации обработки заявок клиентов ООО «IT Consult».

1.4 Постановка задачи на разработку проекта API-платформы

API-платформа для ООО «IT Consult» будет разработана собственными силами. Преимуществами этого подходя приобретения приложения состоит в полном контроле процессов проектирования и разработки, свободе выбора программного обеспечения для разработки, выбор более актуальных технологий для разработки информационной системы, понимание процессов разработки и функционирования системы, обеспечение возможности последующего обновления информационной системы за счет разработки и подключения дополнительных функциональных модулей.

Основной целью разрабатываемой системы является предоставление API для работы с заявками и заказами клиентов. В связи с этим, функционал системы определяет:

- регистрация в системе;
- добавление заявок;
- формирование заказов;
- изменение статуса заявок;
- учет клиентов;
- учет заказов.

Нефункциональные требования:

- система должна быть надежной, защищенной, производительной, сопровождаемой;
- программа оптимизирована и предъявляет не высокие требования к аппаратному обеспечению;

– приложение должно представлять собой веб-сервис для работы с заявками, заказами и клиентами;
– реляционная СУБД;
– API-интерфейс.

Таким образом была поставлена задача на разработку API-платформы которая включает определения особенностей функционала, архитектуры программного средства API-платформы для автоматизации деятельности предприятия.

1.5 Разработка модели бизнес-процесса «Как должно быть»

Целью создания системы является увеличение производительности труда и возможностей масштабирования в процессах взаимодействия с клиентами. Критериями эффективности данной системы являются:

– скорость и удобство обработки новых заявок;
– скорость поиска заявок, а также количество критериев для поиска;
– гибкость настройки системы под существующие бизнес-процессы в организации;
– надежность хранения информации в системе;
– простота внедрения системы и затраты на обучение сотрудников;
– простота эксплуатации;
– возможности интеграций посредством API;
– наличие широкого перечня методов API, которые реализуют все возможности взаимодействия с клиентами.

На основании представленных выше данных принято решение о необходимости разработки API-платформы с набором методов API, которые «позволит автоматизировать деятельность отдела продаж компании (форма-заявка для клиента)» [9].

Задачи проекта автоматизации бизнес-процессов включают [9], [10]:

- повышение уровня интеграционных возможностей со сторонними сервисами;
- разработка набора собственных методов API для обеспечения интеграций с другими сервисами;
- снижение количества времени, которое необходимо для обработки заказа;
- повышение надежность сохранения данных клиентов и заказов;
- совершенствование учета статистики деятельности отдела продаж;
- масштабирование (добавление новых клиентов) деятельности бизнеса.

На рис. 4–5 представлены диаграммы бизнес-процессов «To-Be» организации ООО «IT Consult».

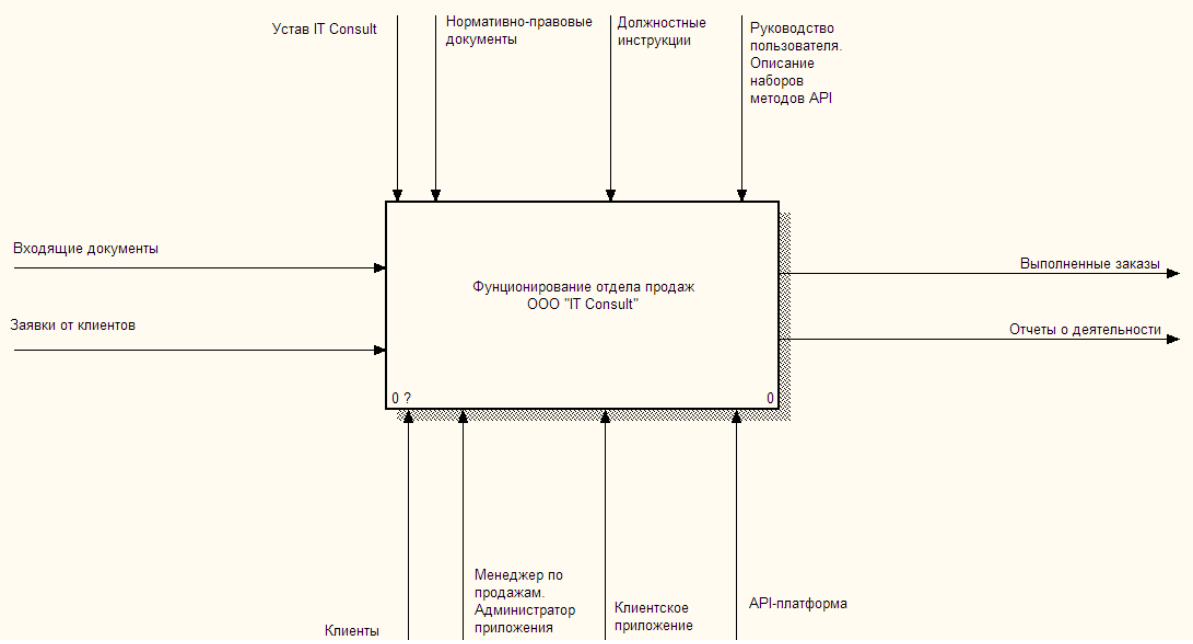

Рисунок 4 - Контекстная диаграмма деятельности (A-0) (Как должно быть)

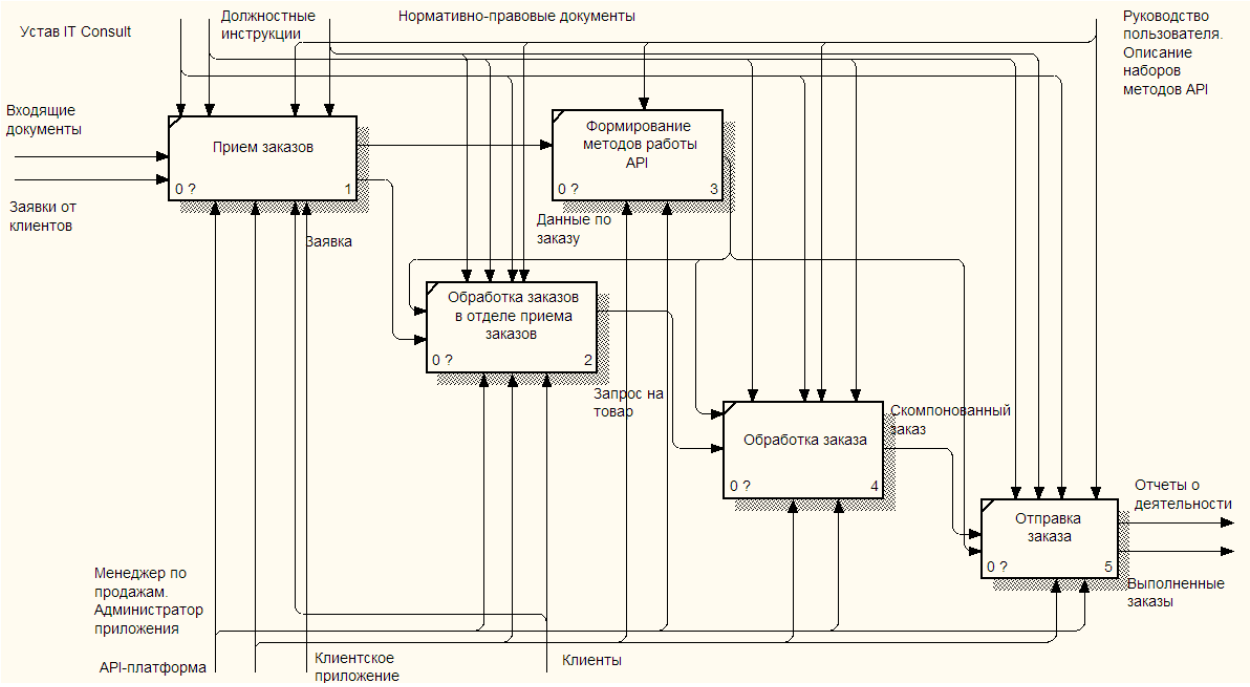

Рисунок 5 - Декомпозиция контекстной диаграммы (A0) (To-Be)

Выводы по главе

В результате выполнения 1 главы было проведено функциональное моделирование предметной области, которое включало: технико-экономическую характеристику ООО «IT Consult»; концептуальное моделирование предметной области; выбор технологии моделирования бизнес-процессов ООО «IT Consult»; моделирование бизнес-процессов ООО «IT Consult» для постановки задачи автоматизированного варианта решения; разработку и анализ модели бизнес-процесса «Как есть»; обоснование необходимости разработки API-платформы; анализ существующих API-платформ для автоматизации бизнес-процессов. После этого была поставлена задача на разработку проекта API-платформы и разработана модель бизнес-процесса «Как должно быть».

Глава 2 Логическое проектирование API-платформы

2.1 Проектирование структуры и функционала API-платформы

Для отображения функционала проектируемого приложения использованы «возможности нотации UML и диаграммы UseCase» [2]. С помощью диаграммы вариантов использования «можно визуально отобразить функции проектируемой платформы. Диаграмма описывает пользователей (участников) системы и функции, которые выполняет каждый субъект (пользователь)» [2].

«Вариант использования является первым концептуальным представлением при проектировании и разработке системы. Эта диаграмма состоит из действующих лиц, примеров использования и отношений между ними. При создании схемы также могут использоваться общие элементы обозначений: примечания и механизмы расширения.

Модель вариантов использования не отражает, как эта группа действий будет реализована и показывает только актеров с различными вариантами взаимодействия» [9].

Основными элементами диаграммы являются актер и сценарий использования (опция). Участник – это логически связанный набор ролей, выполняемых при взаимодействии со сценариями использования или активами (системой, подсистемой или классом). Участник может быть человеком, представляющим что-то, кроме сущности, или другую систему, подсистему или класс. Графически участник изображался как «человечек».

«Понятие прецедент (вариант использования) представляет собой определение ряда последовательных событий (включая опции), которые реализует система, приводящая к результату, наблюдаемому участником. Пример использования представляет поведение объекта, который описывает действия между участниками и системой. Прецедент не отражает «каким образом» прийти к результату, а показывает именно то «что» сделано.

Прецеденты изображаются в форме эллипса, который имеет внутри себя свое собственное название» [18]. Диаграмма показана на рисунке 6.

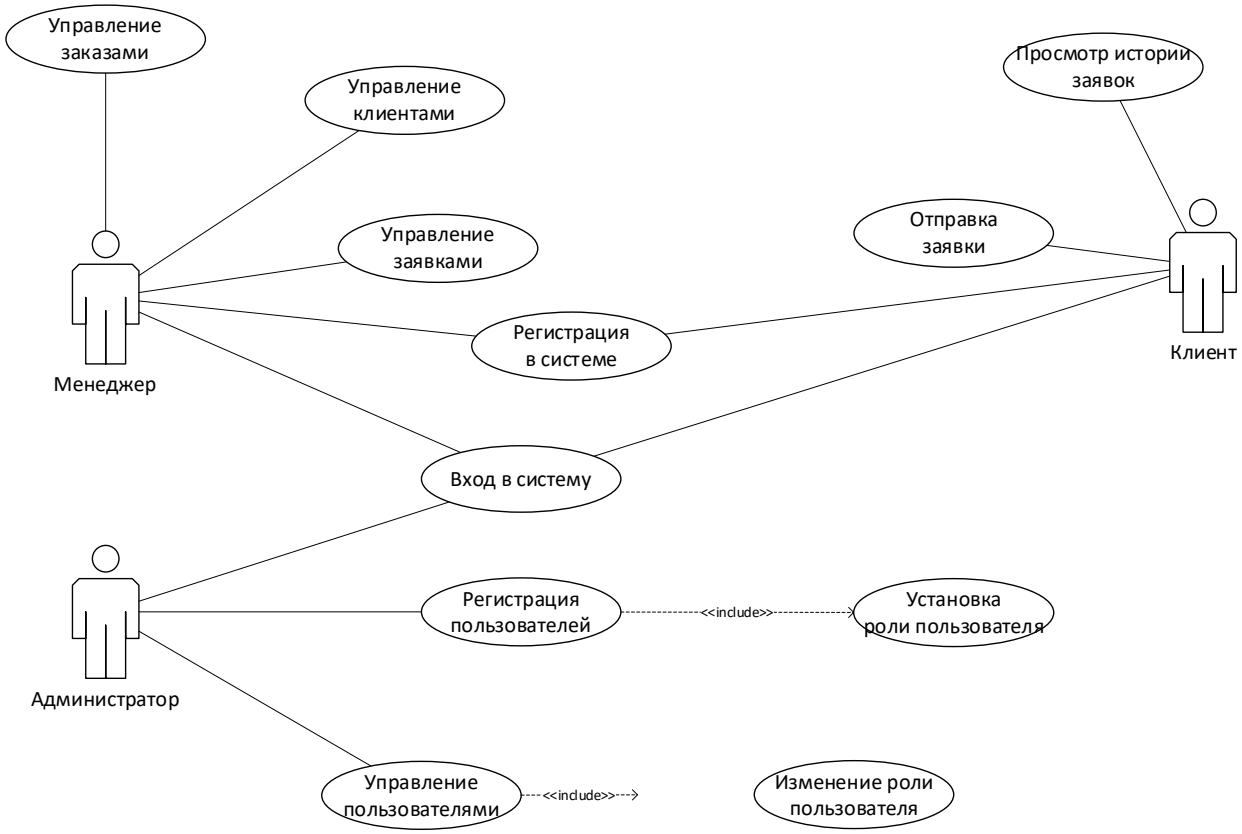

Рисунок 6 – Диаграмма вариантов использования

В данной диаграмме выделены три пользователя: менеджер, администратор системы и клиенты. Менеджеру свойственны следующие варианты использования: управление заказами, управление клиентами, управление заявками, регистрация в системе, вход в систему. Клиента: вход в систему, регистрация в системе, отправка заявки, просмотр истории заявок. Администратор системы: вход в систему, добавление пользователей, удаление пользователей, управление доступом пользователей к системе (установка роли пользователю).

Для отображения структуры приложения – API-платформы использованы возможности диаграммы компонентов. Диаграмма компонентов предназначена для распределения классов и объектов по

компонентам при физическом проектировании системы. Часто данный тип диаграмм называют диаграммами модулей.

«Диаграммы компонентов – это один из двух видов диаграмм, применяемых при моделировании физических аспектов объектно-ориентированной системы. Они показывают организацию наборов компонентов и зависимости между ними. Это статическая структурная диаграмма, показывающая разбиение системы на отдельные структурные компоненты и связи между этими компонентами. В качестве компонентов обычно выступают файлы, модули, библиотеки и другие программные компоненты» [5].

Архитектура программного средства состоит из таких компонентов:

– сервер веб-приложения;

– сервер БД;

– веб-клиент (веб-браузер);

– набор API для взаимодействия сервера и клиента.

Визуально архитектура приложения показана с помощью диаграммы компонентов, которая изображена на рисунке 7.

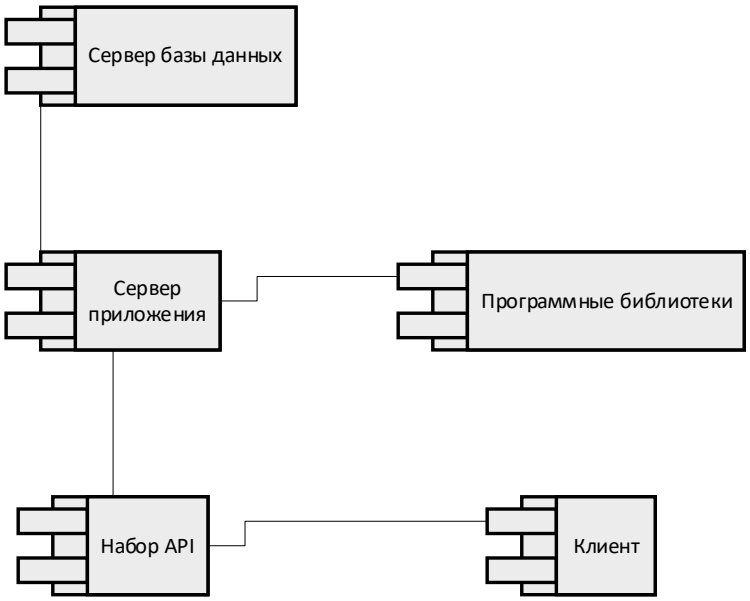

Рисунок 7 – Диаграмма компонентов

«Сервер веб-приложения отвечает за обработку запросов клиента. Веб-сервер отвечает за функционал приложения, который включает авторизацию, регистрацию и другие функции приложения, которые определяют обработку запросов клиента. Взаимодействие веб-сервера с другими компонентами происходит по протоколу HTTP посредством API» [12].

«Сервер БД осуществляет управление БД, отвечает за целостность и сохранность данных, которые попадают в базу данных с веб-приложения. Веб-сервер формирует запросы к БД, после чего запрос отправляется серверу БД, где обрабатывается, осуществляются необходимые операции и возвращается результат» [11].

«Веб-клиент (веб-браузер) отправляет и получает данные с сервера путем использования набора API» [17].

«Диаграмма развертывания – это диаграмма, целью которой является представление узлов выполнения компонентов программы реального времени и процессов с объектами» [21]. Диаграмма развертывания приведена на рисунке 8.

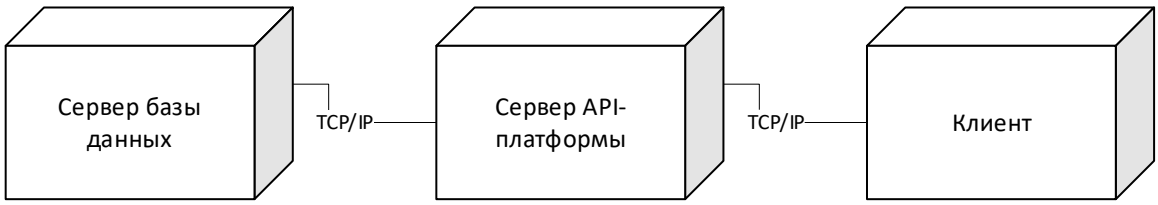

Рисунок 8 – Диаграмма развертывания

«Через интерфейс БД происходит общение с сервером БД, посредством программных библиотек осуществляется обработка данных, которые получены с целевых сайтов с использованием прокси-сервера, которые после сохраняются в БД» [11].

Система состоит из 3 узлов: веб-сервер, сервер БД, клиент. Клиент обращается к веб-серверу с помощью наборов API, сервер обрабатывает запросы клиента и возвращает ответ. В свою очередь веб-сервер обращается к

серверу БД. Взаимодействие веб-сервера с сервером БД обеспечивает сохранение и обработку данных, которые извлекает парсер.

2.2 Информационное обеспечение API-платформы

Эффективному поиску, обработке и передаче информации в информационной системе предшествует процесс ее адаптирования, для чего ее нужно классифицировать, закодировать, что позволит удобно представить информацию в цифровом виде. Как правило, данные действия проводятся со справочными реквизитами, которые являются признаками объектов и субъектов деятельности. Кодирование относится к признакам, по которым в дальнейшем будет реализована группировка данных. Для рассматриваемой API-платформы предварительно был произведен классификатор локального характера (классификация иерархическим методом). Классификаторы и системы кодирования показаны в таблице 1.

Таблица 1 – Классификаторы и системы кодирования

Наименование кодируемого множества объектов	Значность кода	Система кодирования	Вид классификатора
Код пользователя	XXXXX	порядковая	локальный
Код заказа	XXXXX	порядковая	локальный
Код заявки	XXXXX	порядковая	локальный
Код роли пользователя	XXXXX	порядковая	локальный
Код разрешения на доступ к материалам (страницам)	XXXXX	порядковая	локальный
Код клиента	XXXXX	порядковая	локальный
Код услуги	XXXXX	порядковая	локальный
Код товара	XXXXX	порядковая	локальный

Применение одинакового кода относится ко всем кодам. Их длина формируется исходя из максимально возможного количества объектов, которые относятся к одному типу. Представленная нормативно-справочная информация описывает кодификацию данных при разработке API-платформы

для компании ООО «IT Consult». Описание нормативно-справочной информации представлено в таблице 2.

Таблица 2 – Перечень нормативно-справочных документов

Название документа	**Содержание**	**Периодичность**
Список клиентов	Клиенты	Перед внедрением
Список заявок	Заявки клиентов	Перед внедрением
Список услуг	Услуги	Перед внедрением
Список товаров	Товары	Перед внедрением
Список ролей пользователей	Роли пользователей	Перед внедрением
Список разрешений для пользователя	Разрешения для пользователя	Перед внедрением

Входной информацией являются данные о заявках, заказах, клиентах, услугах, товарах, ролях пользователей. Передаются указанные данные на сервер посредством API-методов в формате JSON.

При описании выходной результатной информации отражается, какая информация и в каком виде должна быть получена в результате функционирования API-платформы. Выходной информацией являются данные в формате JSON которые отправляются сервером клиенту.

Выходной документ и пути его дальнейшего использования является результатом работы сервера API-платформы. Меняться будет только содержательная часть, поскольку состояние базы данной, в которой выбирается информация, постоянно актуализируется.

Выходной информацией являются следующие наборы данных:

- клиенты компании;
- заявки клиентов;
- заказы;
- перечень услуг ООО «IT Consult»;
- перечень товаров ООО «IT Consult»;
- данные пользователей;

– профиль пользователя;
– данные клиентов;
– данные заказов;
– роли пользователей;
– разрешения пользователя.

Таблица 3 - Характеристика выходной информации

Наименование	Назначение реквизита	Тип реквизита
Пользователи		
Номер пользователя	Номер пользователя	Числовой
Логин	Логин пользователя	Текстовый
Пароль	Пароль пользователя	Текстовый
Роль	Роль пользователя	Числовой, наличие связи (внешний ключ)
Клиенты		
Номер клиента	Номер клиента	Числовой
ФИО	Фамилия, Имя, Отчество	Текстовый
Номер телефона	Номер телефона клиента	Текстовый
Email	Электронный адрес	Текстовый
Дополнительно	Дополнительная информация	Текстовый
Заявки		
Номер	Номер заявки	Числовой
Дата	Дата создания заявки	Дата
Пользователь	Номер пользователя	Числовой
Статус	Статус заявки	Числовой
Клиент	Номер клиента	Числовой
Дата	Дата создания заявки	Дата/время
Заказы		
Номер	Номер заказа	Числовой
Заявка	Номер заявки	Числовой
Статус оплаты	Статус оплаты	Числовой
Сумма	Сумма заказа	Числовой
Оформил	Оформил менеджер	Числовой
Дата	Дата создания заказа	Дата/время
Услуги		
Номер	Номер услуги	Числовой
Краткое описание	Краткое описание услуги	Текстовый
Наименование	Наименование услуги	Текстовый
Стоимость	Стоимость услуги	Числовой

Продолжение таблицы 3

Товары		
Номер	Номер услуги	Числовой
Краткое описание	Краткое описание товара	Текстовый
Наименование	Наименование товара	Текстовый
Стоимость	Стоимость товара	Числовой
Роли		
Номер	Номер роли	Числовой
Наименование	Наименование роли	Текстовый
Разрешения		
Номер	Номер разрешения	Числовой
Наименование	Наименование разрешения	Тестовый
Разрешения роли		
Номер роли	Номер роли	Числовой
Номер разрешения	Номер разрешения	Числовой
Номер	Номер разрешения роли	Числовой
Товары (услуги) в заявке		
Клиент	Номер клиента	Числовой
Товар	Номер товара	Числовой
Услуга	Номер услуги	Числовой
Заявка	Номер заявки	Числовой
Номер	Номер записи о товаре (услуге) в заявке	Числовой

В результате рассмотрения информационного обеспечения программного средства были определены используемые классификаторы и системы кодирования, дана характеристика нормативно-справочной и входной оперативной информации, определен состав выходной информации. Разработанной информационное обеспечение послужит основой для дальнейшего проектирования базы данных программного средства.

2.3 Проектирование базы данных API-платформ

В настоящее время существует несколько приемов выделения сущностей и связей – нотации Чена, IDEF1X и другие. Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности.

«Атрибут (Attribute) – любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Наименование атрибута должно быть выражено существительным в единственном числе» [14].

«Связь (Relationship) – поименованная ассоциация между сущностями, значимая для рассматриваемой предметной области» [14].

В нотации Чена различают з»ависимые и независимые сущности. Возле каждой сущности на линии, соединяющей ее со связью, цифрами указывается класс принадлежности. Сущности и связи могут иметь атрибуты. Для каждой сущности находится атрибут (или набор атрибутов), значение которого однозначно определяет экземпляр сущности. Этот атрибут является ключом сущности. Связь также может иметь ключевой атрибут. В ряде случаев для удобства организации связей в состав атрибутов сущности вводится искусственный ключ (обычно число). Ключевой атрибут (набор атрибутов) на диаграмме отмечается двумя линиями снизу, внешние ключи отмечаются одной линией» [11].

В нотации IDEF1X «все сущности делятся на зависимые и независимые от идентификаторов аналогично нотации Чена. Независимая сущность изображается в виде обычного прямоугольника, зависимая – в виде прямоугольника с закругленными углами» [5].

Концептуальный уровень создаваемой БД является обобщающим представлением данных. «Концептуальная модель предметной области описывает логическое представление данных (рис. 9). Она является представлением требований к организации данных со стороны пользователей системы» [20].

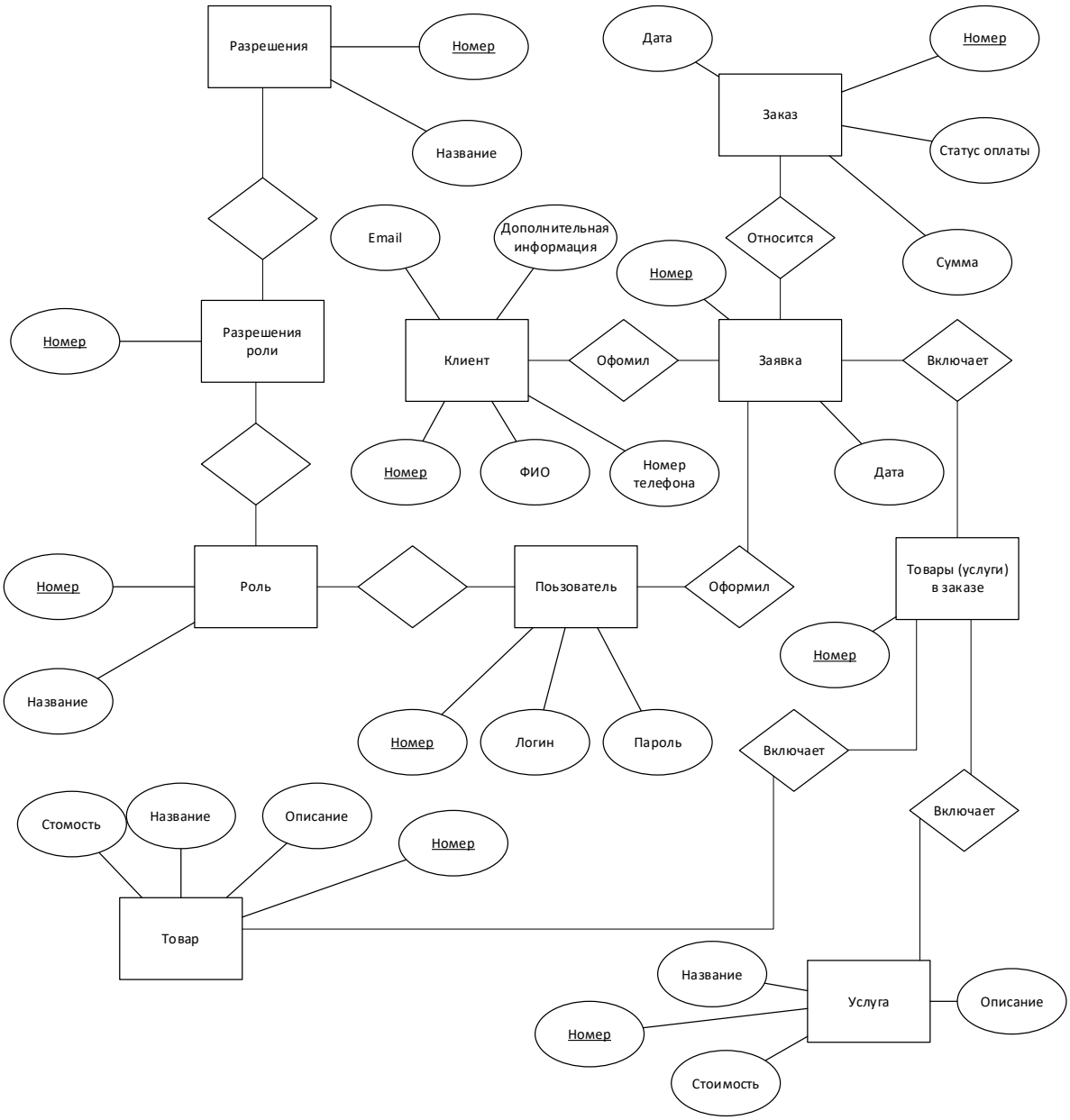

Рисунок 9 – Концептуальная модель данных API-платформы

«Концептуальная модель – это модель, которая отображает знания в предметной области об ее объектах и их взаимосвязях, процессах и результатах деятельности. Концептуальная модель может быть определена как модель, которая состоит из концепций и их отношений. Концептуальная модель — это первый шаг моделирования системы. Это помогает понять сущности в реальном мире и то, как они взаимодействуют друг с другом.

Концептуальная модель может быть определена как модель, которая состоит из концепций и их отношений» [20].

Таким образом была разработана концептуальная модель базы данных, которая послужит основой для дальнейшего логического проектирования, результатом которого будет разработанная структура базы данных API-платформы.

2.4 Разработка логической модели данных API-платформы

Для создания логической структуры база данных использованы возможности приложения CA ERwin® Data Modeler. ERwin Data Modeler – «CASE-средство для проектирования и документирования баз данных, которое позволяет создавать, документировать и сопровождать базы данных, хранилища и витрины данных» [16].

Созданная логическая модель базы данных API-платформы для компании «IT Consult» показана на рис. 10.

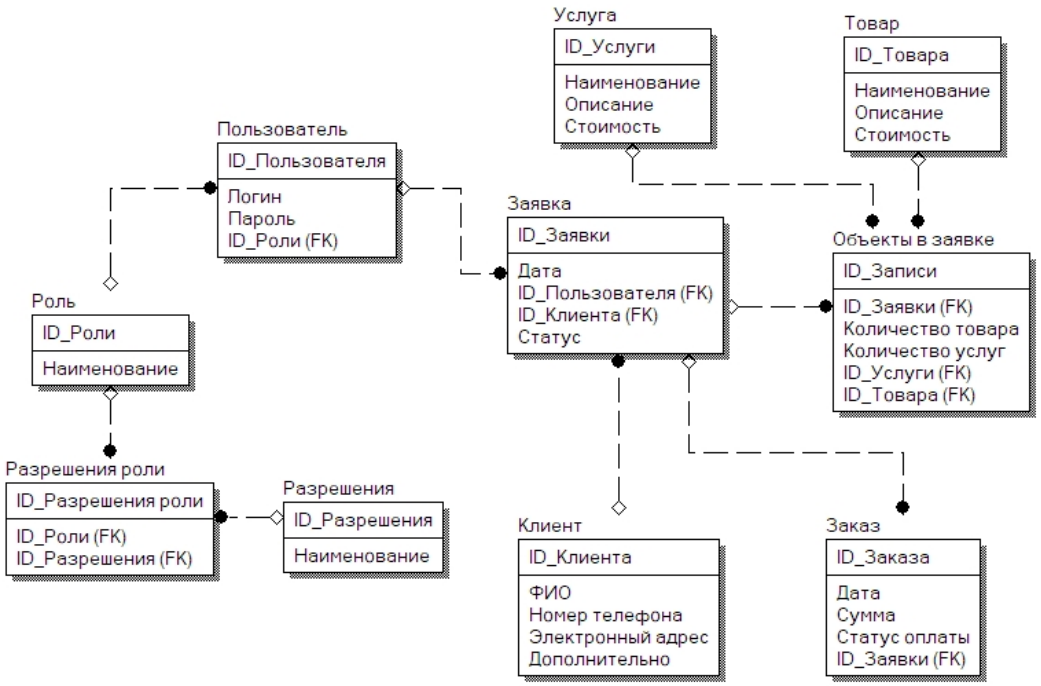

Рисунок 10 – Логическая модель базы данных

ERwin позволяет создавать логическую модель базы данных. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД [20, 25].

Таким образом была разработана логическая модель базы данных. Созданная логическая модель включает ряд сущностей, которые отображают особенности предметной области, а также включают дополнительные сущности, которые обеспечивают вспомогательные возможности по управлению программным средством.

2.5 Требования к аппаратно-программному обеспечению API-платформы

Аппаратно-техническое обеспечение по разработке API-платформы для ООО «IT Consult» включает перечень аппаратных и программных требований, которые представлены в таблице 4.

Таблица 4 - Требования к аппаратно-техническому обеспечению

Параметр/Характеристика	**Значение**
Процессор	2,2 GHz Intel Core i7
Количество ядер	**8**
Тактовая частота	2.8 Ghz
Оперативная память	16 Gb
Свободное дисковое пространство	SSD 256 Gb
Видеосистема и монитор	Разрешающая способность 1920 x 1080
Операционная система	Winsows 10 Pro 1809

В процессе работы над API-платформой необходим веб-сервер, который бы «обрабатывал запросы и отправлял в браузер данные из базы данных» [21]. В качестве сервера была использована программная среда OpenServer. Программный комплекс имеет тщательно подобранный набор серверного программного обеспечения, а также удобный и продуманный

многофункциональный интерфейс, который обладает мощными возможностями по администрированию и настройке компонентов» [22].

Внешний вид рабочих окон Open Server Panel показаны на рис. 11–14.

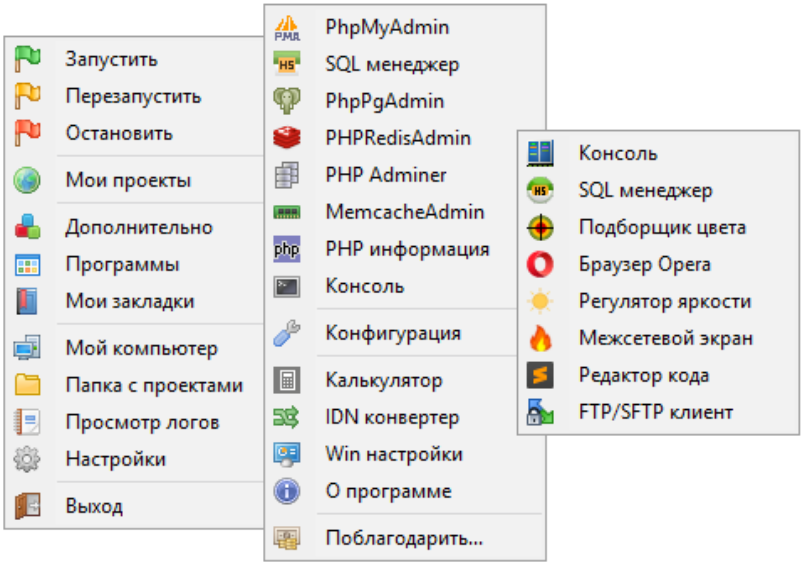

Рисунок 11 – Всплывающие окна управления OSPanel

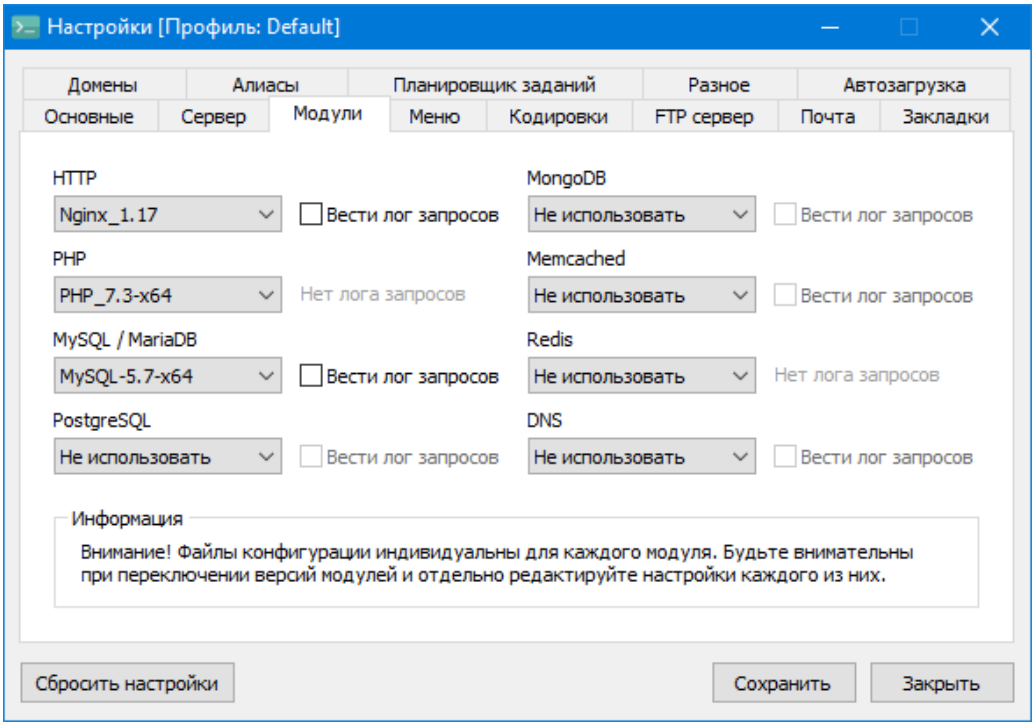

Рисунок 12 – Панель управления. Модули системы

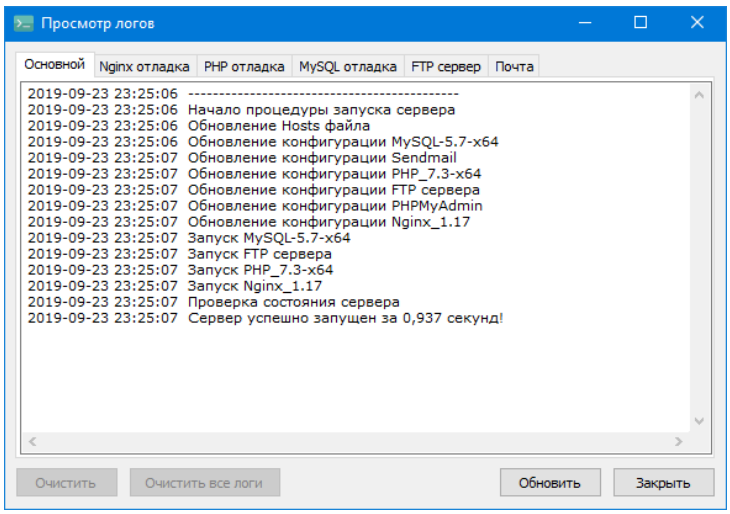

Рисунок 13 – Лог работы программы

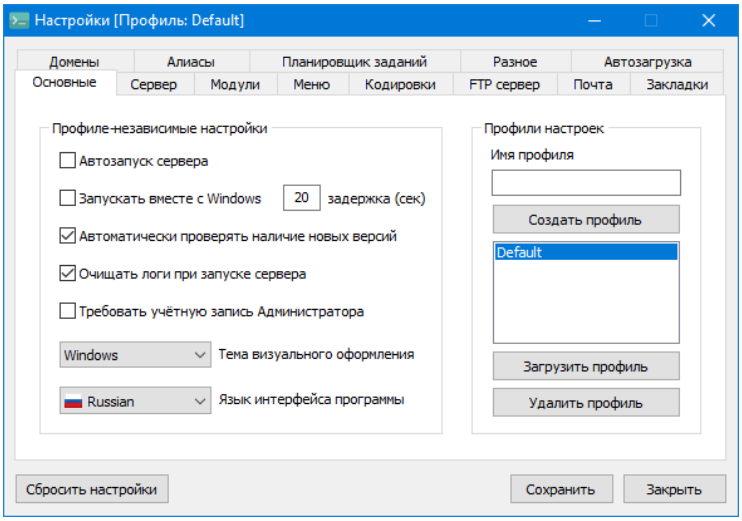

Рисунок 14 – Настройка профиля пользователя

Выводы по главе

При выполнении второй главы работы было осуществлено логическое проектирование API-платформы для ООО «IT Consult». При этом был создан проект структуры и функционала API-платформы для ООО «IT Consult», определено информационной обеспечение и создана концептуальная и логическая модель базы данных для API-платформы для ООО «IT Consult», определены требования к аппаратно-программному обеспечению проектируемой системы.

Глава 3 Физическое проектирование API-платформ

3.1 Выбор архитектуры API-платформы

Архитектура платформы основана на модели клиент-сервер, которая включает три компонента – сервер базы данных, сервер приложения, клиентский компьютер. На рисунке 15 показана трехзвенная архитектура приложения с базами данных.

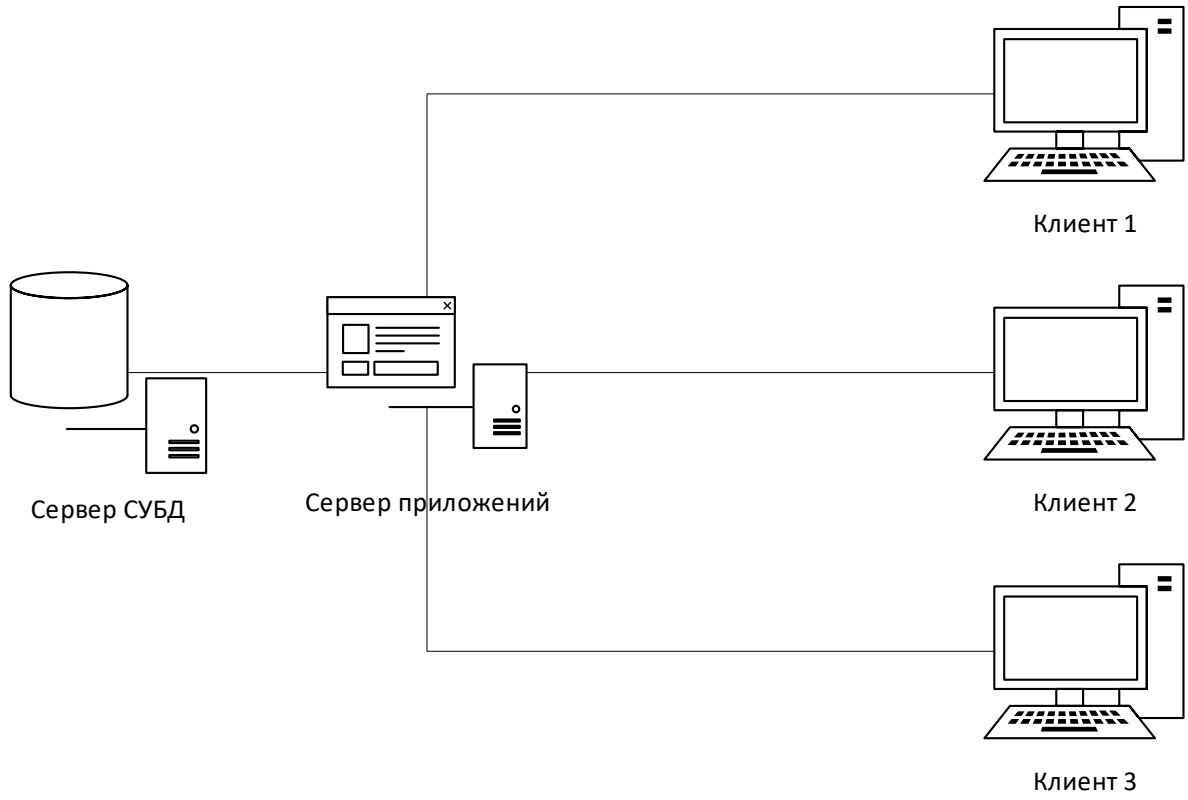

Рисунок 15 - Трехзвенная архитектура (сервер СУБД – сервер приложения – клиент)

Взаимодействие компонентов архитектуры приложения происходит благодаря использованию протокола HTTP (Hyper Text Transfer Protocol). Оптимальным решением для организации передачи данных между разными службами (модулями системы) является технология Rest API [3].

API или программный интерфейс приложения представляет собой набор правил, определяющих способ взаимодействия между приложениями или устройствами. Rest API — это API, соответствующий принципам архитектурного стиля Rest (от англ. Representational State Transfer —«передача состояния представления») [3]. По этой причине Rest API иногда называют Restful API.

Rest (сам термин был введен Роем Филдингом в его докторской диссертации в 2000 году) обеспечивает относительно высокий уровень гибкости и свободы для разработчиков. Но гибкость — лишь одна из причин, объясняющих популярность Rest API, как способа взаимодействия между компонентами и приложениями в микро-сервисной архитектуре.

Принципы проектирования Rest [4]. На самом базовом уровне API можно представить как механизм, с помощью которого одно приложение или сервис может получить доступ к ресурсу другого приложения или сервиса. Приложение или сервис, запрашивающий доступ, называется клиентом, а приложение или сервис, обладающий необходимым ресурсом, называется сервером.

Некоторые API, например, SOAP или XML-RPC, устанавливают строгие ограничения для разработчиков. Однако для разработки Rest API можно использовать практически любой язык программирования и разнообразные форматы данных. Единственное требование заключается в соблюдении шести принципов проектирования Rest, известных также как архитектурные ограничения:

Единый интерфейс. Все запросы API к одному и тому же ресурсу должны выглядеть одинаково, независимо от отправителя. В Rest API один и тот же элемент данных, например, имя или адрес электронной почты пользователя, должен принадлежать одному универсальному коду ресурса (URI). Размер ресурсов не должен быть слишком большим, однако ресурсы должны содержать всю информацию, которая может потребоваться клиенту [12].

Разделение клиента и сервера. В соответствии с требованиями Rest API клиент и сервер должны работать полностью независимо друг от друга. Клиентскому приложению достаточно знать URI запрашиваемого ресурса; иначе взаимодействие с серверным приложением будет невозможно. Аналогичным образом, серверное приложение не должно вносить никаких изменений в клиентское приложение, а только передавать запрошенные данные через HTTP.

Отсутствие сохранения состояния. Rest API функционируют без сохранения состояния, т. е. каждый запрос должен содержать всю информацию, необходимую для его обработки. Другими словами, для» Rest API не требуется установление постоянных сеансов с сервером. Серверным приложениям запрещено хранить какие-либо данные, связанные с запросом клиента» [4].

Поддержка кэширования. По возможности ресурсы должны поддерживать кэширование на стороне клиента или сервера. Ответы сервера также должны хранить информацию о том, разрешено ли кэширование для предоставленного ресурса. Это позволяет увеличить производительность на стороне клиента, одновременно повысив масштабируемость на стороне сервера.

Многоуровневая архитектура системы. В «Rest API вызовы и ответы передаются через разные уровни. Цепочка передачи данных может включать несколько промежуточных узлов. Rest API должны проектироваться таким образом, чтобы ни клиент, ни сервер не могли знать, взаимодействуют они с конечным приложением или промежуточным узлом» [4].

Код по требованию (необязательное требование). Rest API обычно отправляют статические ресурсы, однако в некоторых случаях ответ может содержать исполняемый код (например, Java-апплеты). В таких случаях код должен выполняться только по требованию.

Принцип работы Rest API [12]. Rest API используют запросы HTTP для выполнения стандартных функций базы данных, таких как создание, чтение,

обновление и удаление записей (так называемые функции CRUD). Так, Rest API может использовать запрос GET для получения записи, запрос POST для создания записи, запрос PUT для обновления записи и запрос DELETE для удаления записи. В вызовах API поддерживаются все методы HTTP. Хорошо продуманный Rest API можно сравнить с веб-сайтом, который работает в веб-браузере со встроенной поддержкой HTTP.

Состояние ресурса в любой момент времени («отметка времени») называется представлением ресурса. Эта информация может быть предоставлена клиенту практически в любом формате, включая JavaScript Object Notation (JSON), HTML, XLT, Python, PHP или текстовом формате. Популярность JSON обусловлена тем, что данный формат не зависит от языка программирования и понятен как человеку, так и компьютеру.

Важную роль в вызовах Rest API также играют «заголовки и параметры запросов, поскольку они содержат такую важную информацию, как метаданные, разрешения, универсальные коды ресурсов (URI), сведения о кэшировании, файлы cookie и многое другое. Заголовки запросов и ответов применяются в хорошо продуманных Rest API вместе с обычными кодами состояния HTTP» [12].

Рекомендации по использованию Rest API. С одной стороны, гибкость является огромным преимуществом при проектировании Rest API; с другой стороны, гибкость может стать причиной дефективного или неэффективного API. По этой причине профессиональные разработчики делятся передовыми практиками в спецификациях Rest API [5].

«Спецификация OpenAPI (OAS) определяет интерфейс для описания API, чтобы обеспечить разработчикам и приложениям полное понимание всех параметров и возможностей API, включая доступные конечные точки, разрешенные операции для каждой конечной точки, параметры операций, методы аутентификации и прочую информацию. Последняя версия, OAS3 (внешняя ссылка), содержит полезные инструменты, например OpenAPI

Generator, для создания клиентов API и серверных заглушек на разных языках программирования.

Для обеспечения безопасности Rest API также следует опираться на передовой отраслевой опыт, в частности использование алгоритмов хэширования для защиты паролей и HTTPS для безопасной передачи данных» [5]. Для ограничения прав доступа сторонних приложений можно использовать инфраструктуру авторизации, например OAuth 2.0 (внешняя ссылка). Кроме того, API может отклонять любые запросы, поступающие по истечении определенного периода времени, используя отметку времени в заголовке HTTP. Существуют и другие способы контроля доступа к API: проверка параметров и веб-маркеры JSON [14; 15].

3.2 Выбор технологии разработки программного обеспечения API-платформы

Для разработки платформы выбран язык программирования PHP и фреймворк – Symfony [6]. PHP - серверный язык создания приложений, ориентированный на веб-разработку. Код PHP интерпретируется веб-сервером и генерирует HTML-код или другой вывод, который отсылается браузеру пользователя [17, 18].

Symfony представляет собой один из самых популярных фреймворков для веб-разработки [6]. Неполный список самых известных проектов, использующих Symfony [6]:

- «Фреймворк Sylius eCommerce;
- Сервис социальных закладок Delicious;
- API Platform — специализированный фреймворк для дизайна Restful API;
- Laravel — доступный, RAD-ориентированный фреймворк;
- Платформа Oro и связанная с ней экосистема: Oro CRM, Akeneo, OroCommerce и Marello».

Symfony — это библиотека готовых решений, которые позволяют на языке PHP удобно и быстро создавать веб-скрипты, веб-приложения [7].

Модульность – одна из наиболее важных особенностей работы с Symfony. Symfony – это набор повторно используемых и автономных компонентов – бандлов [7].

Проанализировав вышеописанную информацию и опираясь на внешние источники, можно определить ряд причин, для использования в проекте PHP-фреймворка Symfony [6]. «Symfony обеспечивает всем, что нужно веб-разработчику: скорость работы, гибкость, многоповторяемость компонентов и т.д. Так же стоит отменить стремление компании SensioLabs постоянно развивать и улучшать Symfony. Symfony не самая быстрая технология, но его архитектура более прозрачна, чем у других фреймворков, что очень важно при создании больших и надёжных приложений» [6].

С технической точки зрения, можно выделить такие свойства Symfony, как: быстродействие и низкая ресурсопотребляемость; неограниченная гибкость в использовании Symfony, за счёт его прозрачной архитектуры; расширяемость; стабильность и устойчивость; простота в использовании.

3.3 Выбор СУБД API-платформы

Для проекта наиболее выбрана СУБД MySQL. СУБД MySQL является одной из самых популярных СУБД на сегодняшний день. СУБД MySQL рационально подходит к решению сложных задач. С помощью СУБД MySQL можно реализовать высоко функциональные системы, в том числе и систему [19].

К преимуществам СУБД MySQL относится [19]:

- упрощение ввода записей;
- упрощение поиска записей;
- гибкость поиска записей;
- гибкость вывода записей;

– доступ к записям многопользовательский;
– возможность передачи записей в электронном виде.

MySQL осуществляет поддержку «транзакций, гарантирующих согласованность данных и сокращающих риск их потери, а также репликации и кластеризации, являющихся методиками, значительно уменьшающими время простоя при сбое сервера» [27].

MySQL, это полноценная многопользовательская система. MySQL включает также «гибкую и мощную систему привилегий, дающую администраторам возможность защиты доступа к критическим данным, применяя комбинацию из схем по проверке подлинности хостов и пользователей» [19, 27].

3.4 Разработка физической модели данных API-платформы

Для проектирования базы данных справочной системы была использована среда MySQLWorkbench и CASE средство ERWin. MySQLWorkbench предназначена для визуального проектирования баз данных и управления сервером MySQL. Для построения моделей предназначена секция DataModeling. CASE средство ERWin использовано для логического моделирования базы данных.

Функционал программы MySQLWorkbench:

– «программа позволяет наглядно представить модель базы данных в графическом виде;
– наглядный, функциональный механизм установки связей между таблицами, в том числе «многие ко многим» с созданием таблицы связей;
– восстановление структуры таблиц из существующей на сервере БД (связи восстанавливаются в InnoDB, при использовании MyISAM – связи необходимо устанавливать вручную);

– удобный редактор SQL запросов, позволяющий сразу же отправлять их серверу и получать ответ в виде таблицы;

– возможность редактирования данных в таблице в визуальном режиме» [19].

В результате анализа предметной области были определены таблицы, которые нужны для реализации функционала системы:

– таблица «Клиенты» содержат информацию о клиентах;

– таблицы «Заказы» содержит информацию по заказам;

– таблица «Услуги» предназначена для хранения информации об услуге;

– таблица «Заявки» предназначена для хранения заявок клиентов.

– таблица «Услуги» предназначена для хранения данных об услугах.

Реализация физической модели база данных осуществлялась с помощью программы Mysql Workbencch (рис. 16).

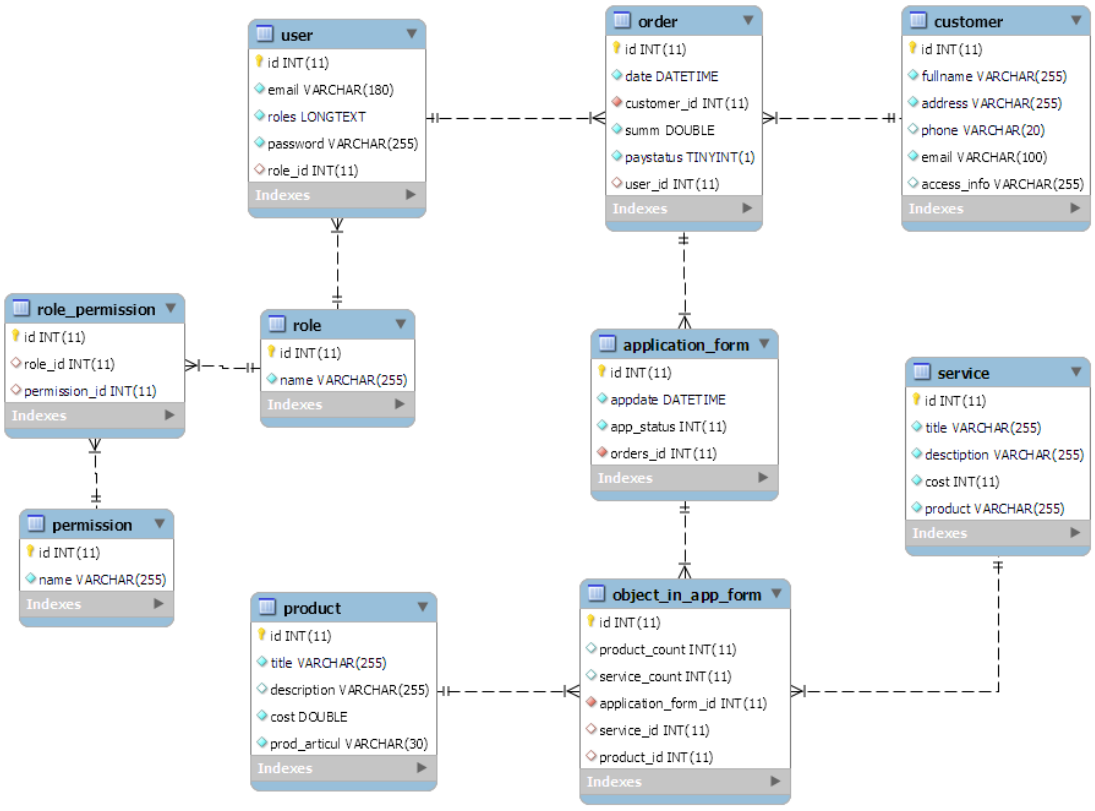

Рисунок 16 - Физическая модель базы данных

Структура таблиц в базе данных представлена на рис. 17 – 26.

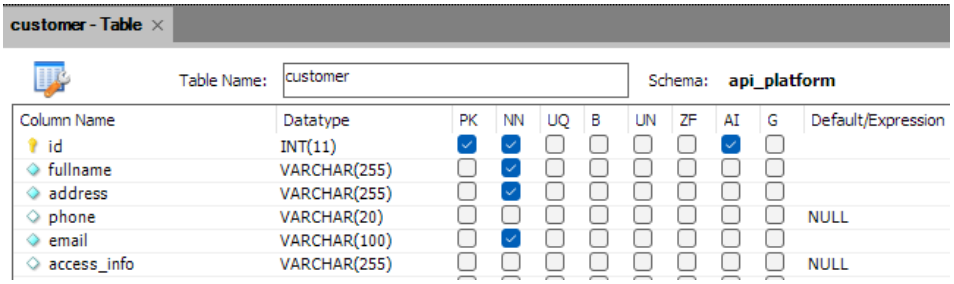
customer - Table

Table Name: customer Schema: api_platform

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT(11)	✓	✓					✓		
fullname	VARCHAR(255)		✓							
address	VARCHAR(255)		✓							
phone	VARCHAR(20)									NULL
email	VARCHAR(100)		✓							
access_info	VARCHAR(255)									NULL

Рисунок 17 - Таблица «customer» (клиенты)

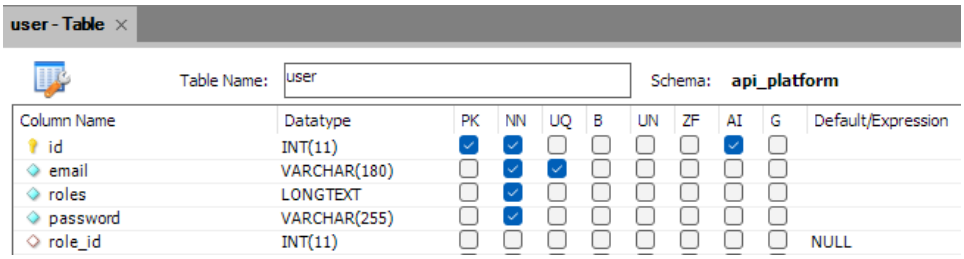
user - Table

Table Name: user Schema: api_platform

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT(11)	✓	✓					✓		
email	VARCHAR(180)		✓	✓						
roles	LONGTEXT		✓							
password	VARCHAR(255)		✓							
role_id	INT(11)									NULL

Рисунок 18 - Таблица «user» (пользователи)

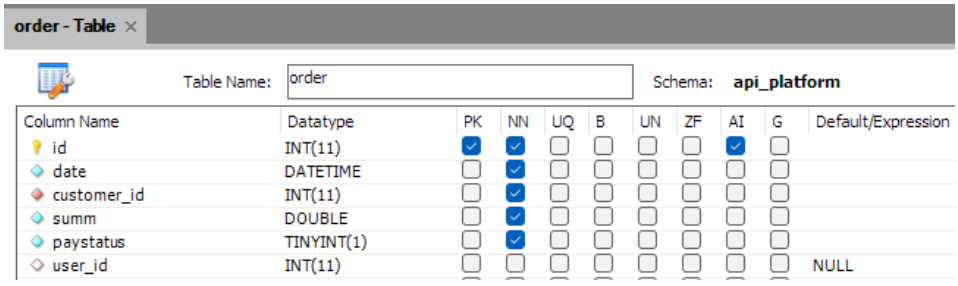
order - Table

Table Name: order Schema: api_platform

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT(11)	✓	✓					✓		
date	DATETIME		✓							
customer_id	INT(11)		✓							
summ	DOUBLE		✓							
paystatus	TINYINT(1)		✓							
user_id	INT(11)									NULL

Рисунок 19- Таблица «orders» (заявки)

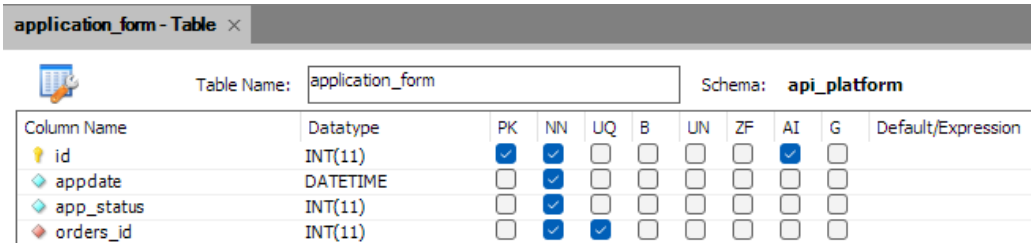
application_form - Table

Table Name: application_form Schema: api_platform

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT(11)	✓	✓					✓		
appdate	DATETIME		✓							
app_status	INT(11)		✓							
orders_id	INT(11)		✓	✓						

Рисунок 20 - Таблица «application_form» (заказы)

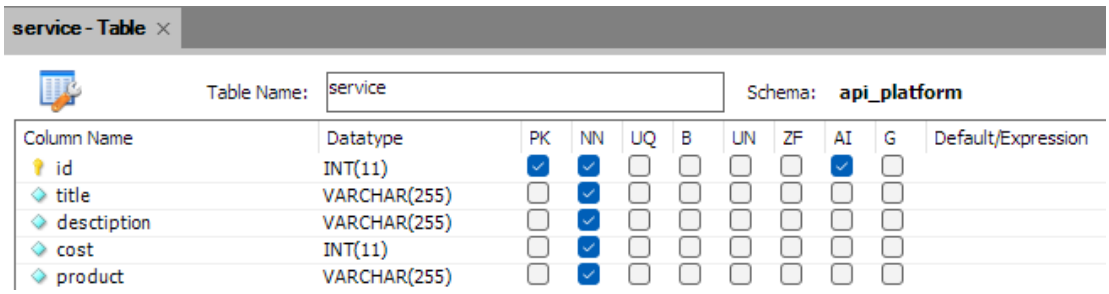

Рисунок 21 - Таблица «service» (услуги)

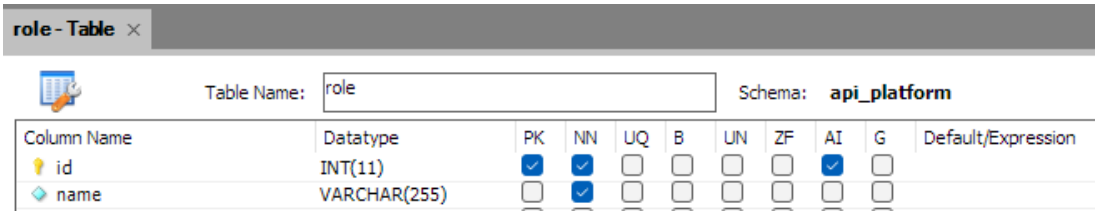

Рисунок 22 - Таблица «roles» (роли)

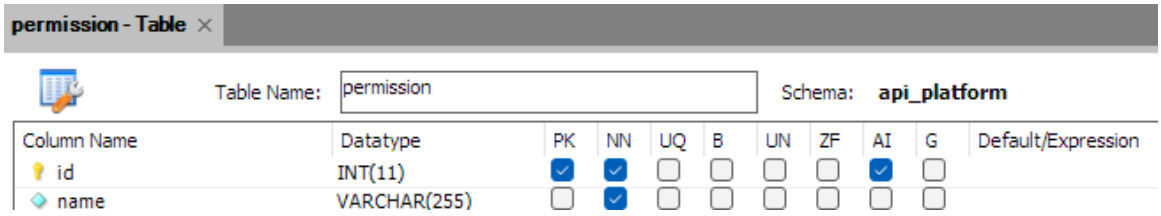

Рисунок 23 - Таблица «» (permission)

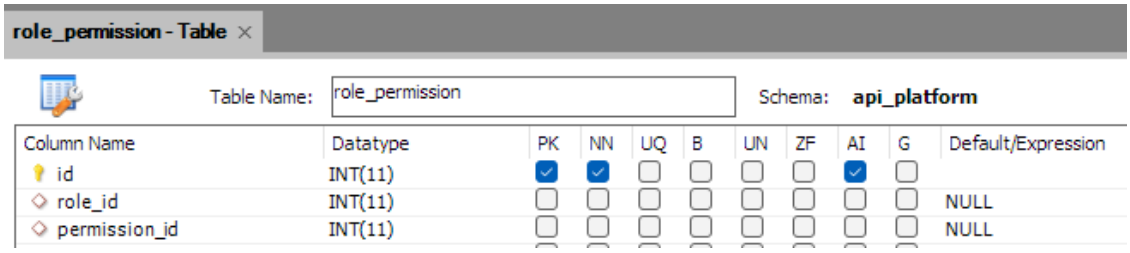

Рисунок 24 - Таблица «role_permission»

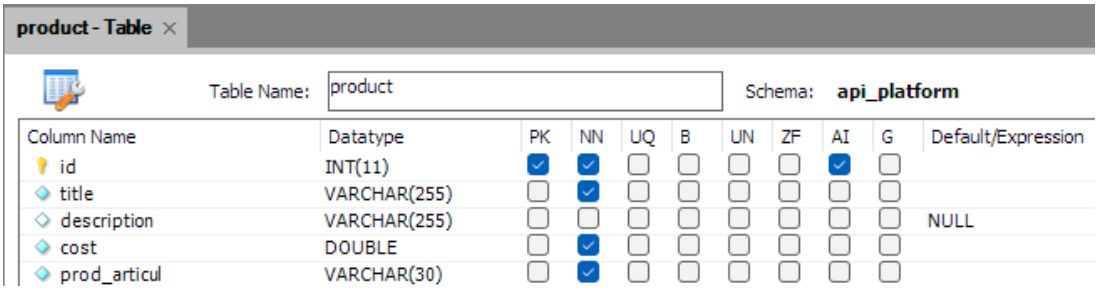

Рисунок 25 – Таблица «product»

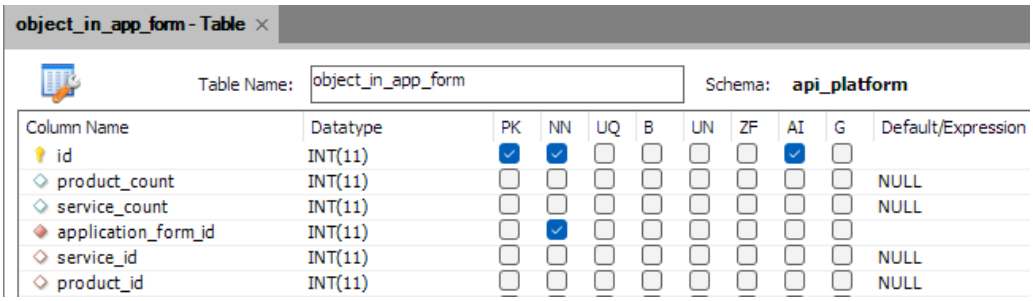

Рисунок 26 - Таблица «object_in_app_form»

После выполнения запроса на сервере MySQL была создана база данных, содержимое и структуру которой можно посмотреть с помощью веб-приложения phpMyAdmin (рис. 27).

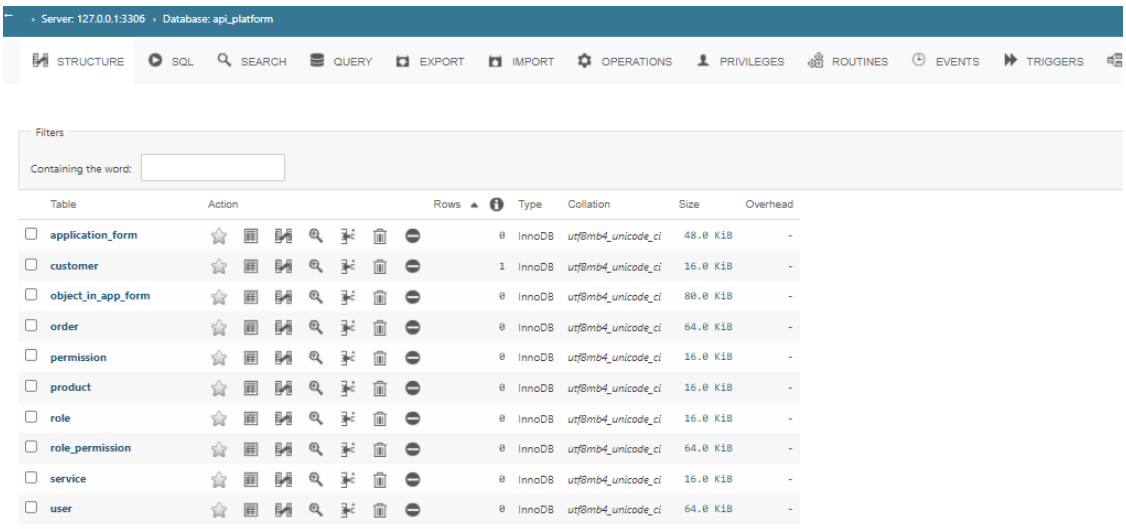

Рисунок 27 - База данных информационной системы

Таким образом была разработана модель базы данных и создана ее физическая реализация на сервере MySQL.

3.5 Разработка программного обеспечения API-платформы

Структура приложения представлена в виде иерархии модулей, когда есть главный модуль и подчиненные ему модули. Такая структура является оптимальной и позволяет реализовать удобное управление компонентами платформы. Схема модулей API-платформы представлена на рис. 28.

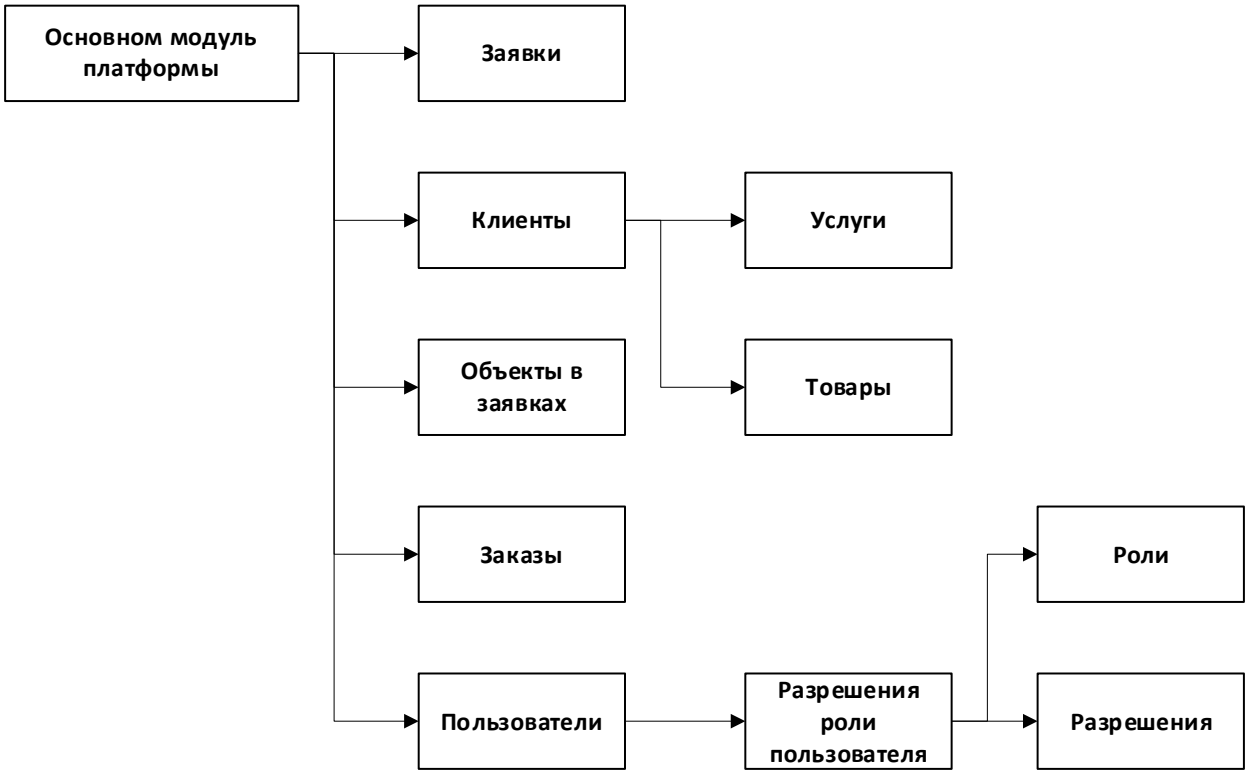

Рисунок 28 – Модули API-платформы

Программная реализация модулей API-платформы представлена на рис. 29–31.

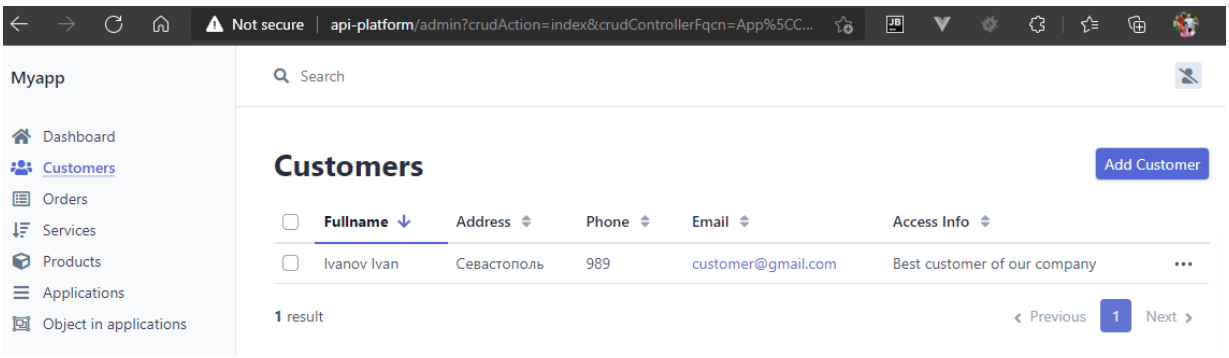

Рисунок 29 – Работа с компонентом административной панели

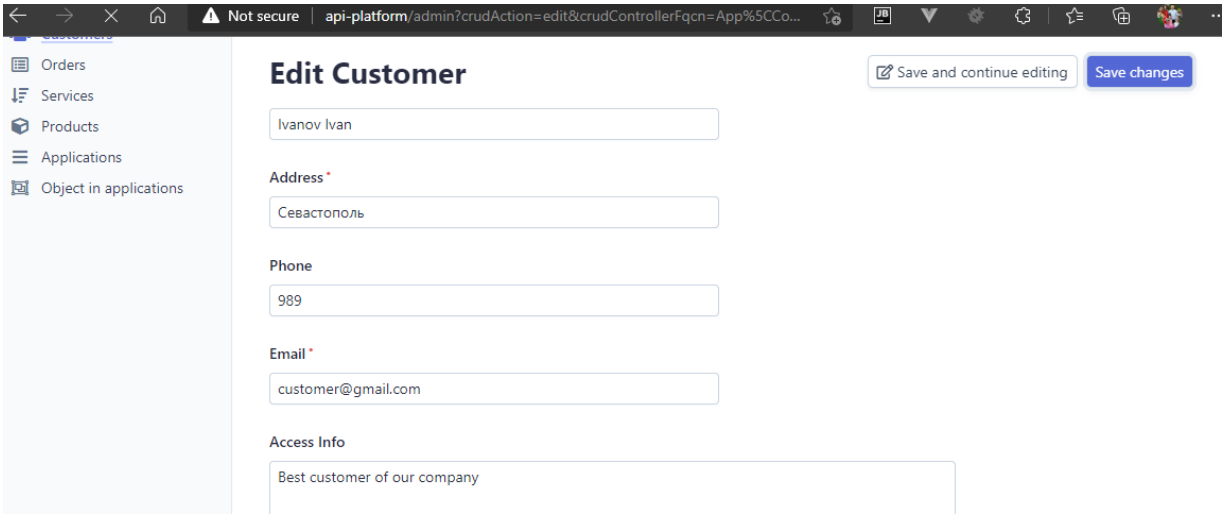

Рисунок 30 – Управление записями в административной панели

Исходный код модулей API-платформы приведен в приложении.

3.6 Описание функциональности API-платформы

Функциональность платформы описана на рис. 31–36 и отображает набор API, которые предназначены для работы с платформой.

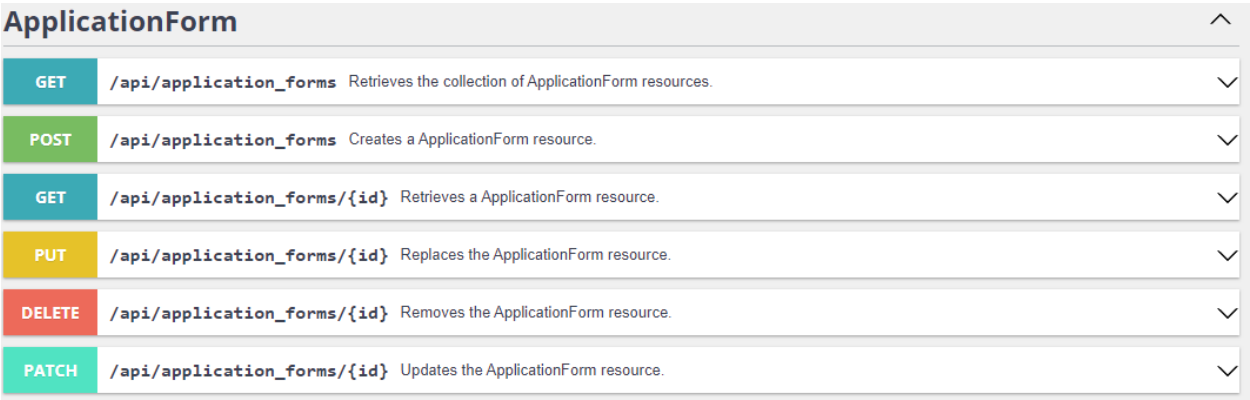

Рисунок 31 - Модуль «ApplicationForm»

– GET - /api/application_forms - Retrieves the collection of ApplicationForm resources.
– POST - /api/application_forms - Creates a ApplicationForm resource.
– GET - /api/application_forms/{id} - Retrieves a ApplicationForm resource.
– PUT - /api/application_forms/{id} - Replaces the ApplicationForm resource.
– DELETE - /api/application_forms/{id} - Removes the ApplicationForm resource.
– PATCH - /api/application_forms/{id} - Updates the ApplicationForm resource.

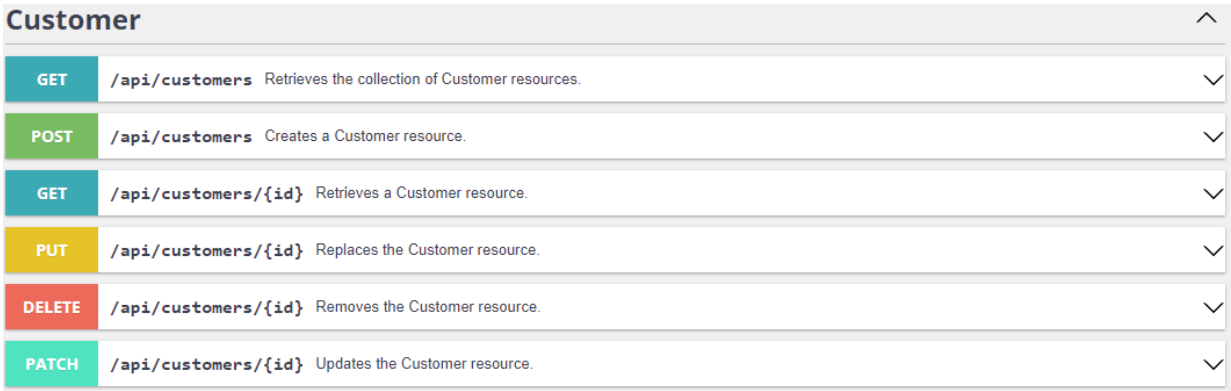

Рисунок 32 - Модуль «Customer»

GET /api/customers - Retrieves the collection of Customer resources.

POST - /api/customers - Creates a Customer resource.

GET - /api/customers/{id} - Retrieves a Customer resource.

PUT - /api/customers/{id} - Replaces the Customer resource.

DELETE - /api/customers/{id} - Removes the Customer resource.

PATCH - /api/customers/{id} - Updates the Customer resource.

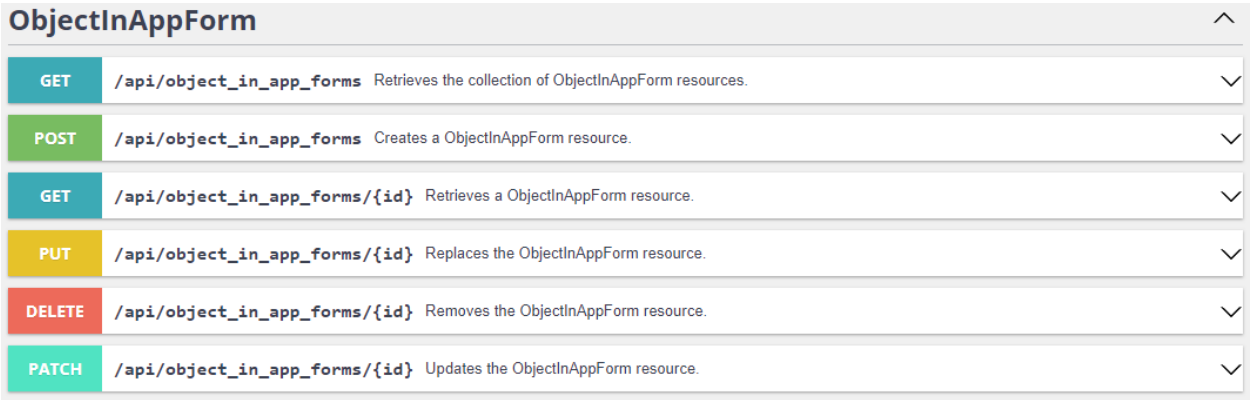

Рисунок 33 - Модуль «ObjectInAppForm»

GET - /api/object_in_app_forms - Retrieves the collection of ObjectInAppForm resources.

POST - /api/object_in_app_forms - Creates a ObjectInAppForm resource.

GET - /api/object_in_app_forms/{id} - Retrieves a ObjectInAppForm resource.

PUT - /ap/object_in_app_forms/{id} - Replaces the ObjectInAppForm resource.

DELETE - /api/object_in_app_forms/{id} - Removes the ObjectInAppForm resource.

PATCH - /api/object_in_app_forms/{id} - Updates the ObjectInAppForm resource.

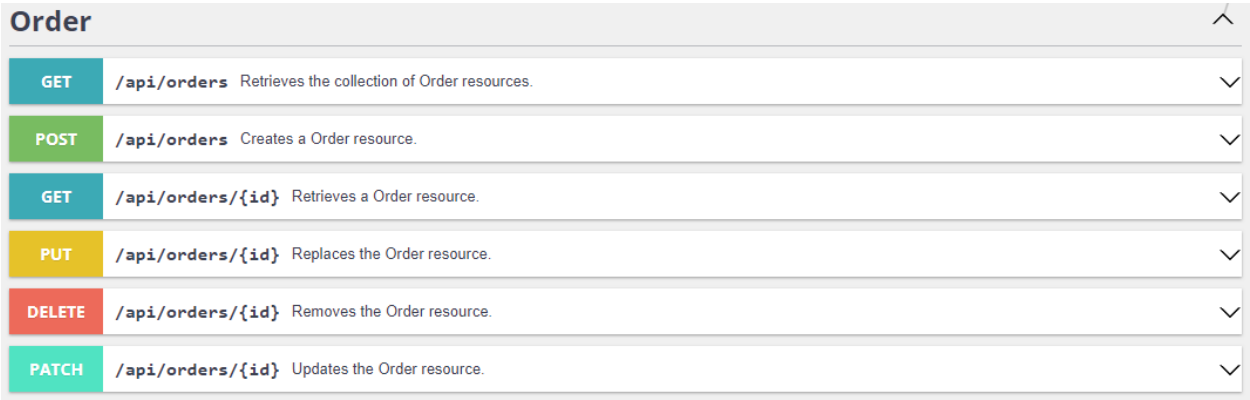

Рисунок 34 - Модуль «Order»

GET /api/orders - Retrieves the collection of Order resources.

POST - /api/orders - Creates a Order resource.

GET - /api/orders/{id} - Retrieves a Order resource.

PUT - /api/orders/{id} - Replaces the Order resource.

DELETE - /api/orders/{id} - Removes the Order resource.

PATCH - /api/orders/{id} - Updates the Order resource.

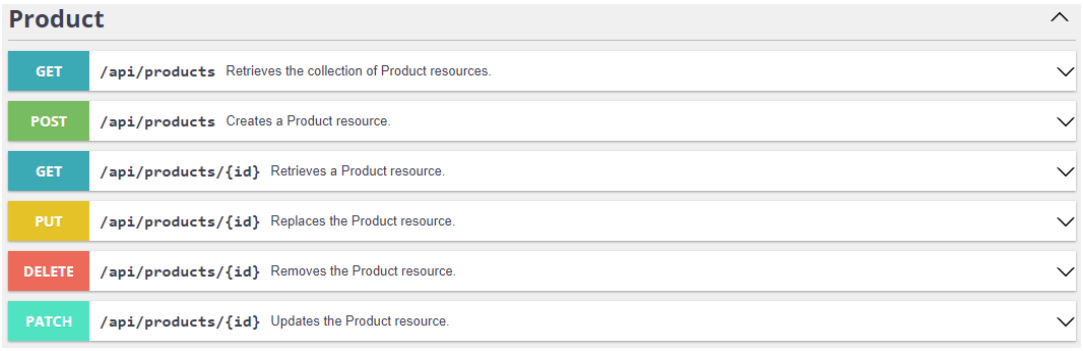

Рисунок 35 - Модуль «Product»

GET /api/products - Retrieves the collection of Product resources.

POST - /api/products - Creates a Product resource.

GET - /api/products/{id} - Retrieves a Product resource.

PUT - api/products/{id} - Replaces the Product resource.

DELETE - /api/products/{id} - Removes the Product resource.

PATCH - /api/products/{id} - Updates the Product resource.

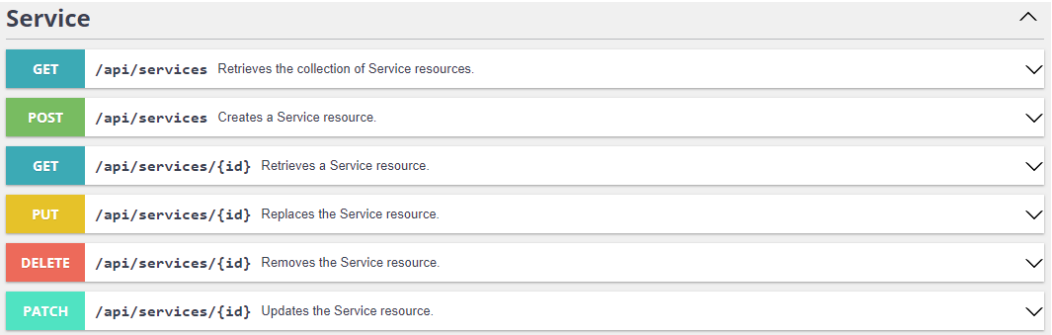

Рисунок 36 - Модуль «Service»

GET - /api/services - Retrieves the collection of Service resources.

POST - /api/services - Creates a Service resource.

GET - /api/services/{id} - Retrieves a Service resource.

PUT - /api/services/{id} - Replaces the Service resource.

DELETE - /api/services/{id} - Removes the Service resource.

PATCH - /api/services/{id} - Updates the Service resource.

3.7 Тестирование программного проекта

В проектах по разработке программного обеспечения помимо основной задачи по реализации заявленной функциональности существует не менее важная задача по обеспечению качества ПО. Процесс тестирования позволит определить скорость загрузки контента, качество адаптивности верстки, отказоустойчивость ресурсов под нагрузкой, метрику кода и корректность работы информационной системы.

Для проверки работоспособности API использована программа Postman. Результаты выполнения запроса к API созданного сервисам показаны на рис. 37.

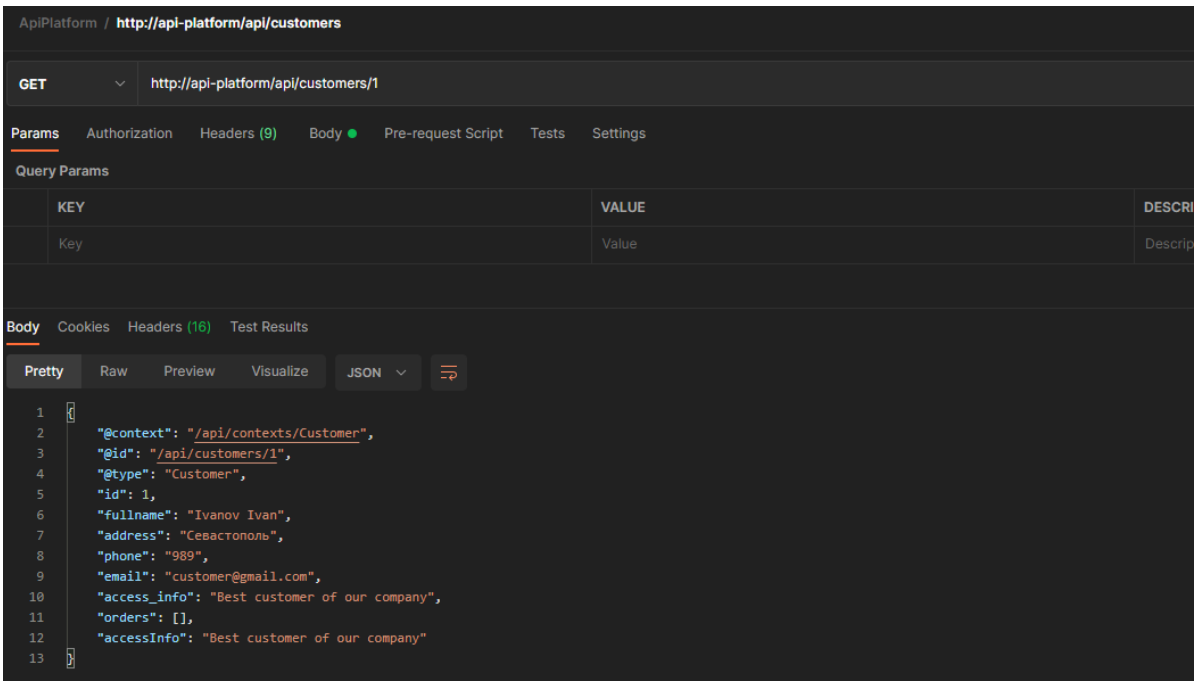

Рисунок 37 – Проверка работы API с помощью Postman

В таблице 5 представлены результаты тестов проверки работы разработанного мобильного приложения.

Таблица 5 - Проверка работоспособности системы

ID	Модуль	Описание теста	Ожидаемый результат	Прохождение теста
1	Модуль «Заявки»	Проверка API работы с заявками	Функционирование методов GET, POST, PUT, PATCH, DELETE	Тест пройден
2	Модуль «Заказы»	Проверка API работы с заказами	Функционирование методов GET, POST, PUT, PATCH, DELETE	Тест пройден
3	Модуль «Услуги»	Проверка API работы с услугами	Функционирование методов GET, POST, PUT, PATCH, DELETE	Тест пройден

На этом этапе выявляются возможные ошибки, допущенные на предыдущих этапах. Синтаксические ошибки в тексте программы

автоматически идентифицируются еще на этапе трансляции, и пользователь вносит соответствующие изменения в текст. Может случиться так, что текст программы записано, верно, а ошибка допущена при составлении алгоритма – программа работает, но выдает неправильные результаты. Если же ошибка допущена на этапе постановки задачи, то программа работает правильно, но решает другую задачу.

Итак, целью отладки является получение правильной программы, результатам работы которой можно было бы доверять.

Суть отладки заключается в том, что пользователь разрабатывает систему тестов, с помощью которой проверяется работа программы в различных возможных режимах. Каждый тест имеет набор входных данных, для которых известен результат. Тест стараются выбрать так, чтобы не только установить сам факт ошибки, но и локализовать ее, то есть выявить и сузить часть программы, содержащей ошибку.

В состав систем программирования включают специальные возможности отладки программ. Пользователь формулирует задачу, а система выполняет эту задачу и выдает пользователю необходимую информацию о том, как ведет себя программа. Такая информация значительно облегчает поиск и исправление ошибок.

В результате проведения тестирования приложения были проверены его функции, при этом было определено, что все спроектированные функции работают.

Выводы по главе

В результате выполнения 3 главы работы было проведено физическое проектирование API-платформы, разработана физическая модель базы данных, выбрана архитектура платформы, выбраны технологии реализации и СУБД системы, разработана схема взаимодействия модулей, описаны возможности системы, проведена оценка экономической эффективности и тестирование программного проекта.

Заключение

В результате работы была достигнута поставленная цель – разработана единая API-платформа для автоматизации бизнес-процессов компании ООО «IT Consult». При достижении цели были решены следующие задачи:

- проведено функциональное моделирование предметной области;
- проведено логическое моделирование предметной области;
- осуществлена физическая реализацию единой API-платформы для автоматизации бизнес-процессов компании ООО «IT Consult».

При функциональном моделировании предметной области проведено: концептуальное моделирование предметной области; выбор технологии моделирования бизнес-процессов ООО «IT Consult»; моделирование бизнес-процессов ООО «IT Consult» для постановки задачи автоматизированного варианта решения; разработку и анализ модели бизнес-процесса «Как есть»; обоснование необходимости разработки API-платформы; анализ существующих API-платформ для автоматизации бизнес-процессов. После этого была поставлена задача на разработку проекта API-платформы и разработана модель бизнес-процесса «Как должно быть». В процессе логического проектирования было осуществлено логическое проектирование API-платформы для ООО «IT Consult». При этом был создан проект структуры и функционала API-платформы для ООО «IT Consult», определено информационной обеспечение и создана концептуальная и логическая модель базы данных для API-платформы для ООО «IT Consult», определены требования к аппаратно-программному обеспечению проектируемой системы. В результате физического проектирования было проведено физическое проектирование API-платформы, разработана физическая модель базы данных, выбрана архитектура платформы, выбраны технологии реализации и СУБД системы, разработана схема взаимодействия модулей, описаны возможности системы, проведено тестирование программного проекта.

Список используемой литературы

1. Автоматизация управления предприятием- М.: ИНФРА - М. 2000. - 239 с.
2. Анализ бизнес-процессов деятельности оптовой фирмы (на примере ЧП 1000 мелочей). Финансовый университет, Кемерово, 2017. 65 с.
3. Аткинсон MySQL. Библиотека профессионала / Аткинсон, Леон. - М.: Вильямс, 2014. - 624 с.
4. Введение в Rest API: что это простыми словами [Электронный ресурс] / Режим доступа https://mcs.mail.ru/blog/vvedenie-v-rest-api (дата обращения: 18.09.2021).
5. Долганова, О.И. Моделирование бизнес-процессов: Учебник и практикум для академического бакалавриата / О.И. Долганова, Е.В. Виноградова, А.М. Лобанова. - Люберцы: Юрайт, 2016. - 289 с.
6. Информационные технологии. Процессы жизненного цикла программных средств: СТБ ИСО/МЭК 12207–2003 [Электронный ресурс]. – Режим доступа: www.nbrb.by/payment/TechCode Pract/pdf/tcp_135_2008.pdf. – Дата доступа: 17.08.2021.
7. Кожевников А. В. Тарасов А. Г. Исследование процессов взаимодействия продавца с клиентами и разработка CRM-системы с открытым API для интеграции с интернет-магазином, анализ структуры и требований стандартов к компонентам CRM-систем /
8. Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли. - М.: Вильямс И.Д., 2017. - 1440 с.
9. Кузнецов Максим, Симдянов Игорь Самоучитель PHP 5; БХВ-Петербург - М., 2017. - 560 с.
10. Локхарт Джош Современный PHP. Новые возможности и передовой опыт; ДМК Пресс - М., 2016. - 304 с.

11. Мартишин, С.А. Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench: Методы и средства проектирования информационных систем и техноло / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. - М.: Форум, 2018. - 61 с.

12. Методические указания к лабораторным работам «Методы проектирования схем баз данных» по дисциплине «Распределенные информационно-аналитические системы» для студентов специальностей 122 – Компьютерные науки и информационные технологии,124 – Системный анализ/ Сост. Ю.Н. Кожин, О.Н. Малых, В.Ф. Прокопенков.– Х.: НТУ “ХПИ”, 2017.–32 с

13. Поллис, Г. Разработка программных проектов на основе Rational Unified Process (RUP) / Г. Поллис, Л. Огастин, К. Лоу. – М.: Бином-Пресс, 2009. – 346 с.

14. Программная среда для веб-разработки Open Server Panel [Электронный ресурс] / Режим доступа: https://ospanel.io/ (дата обращения: 18.08.2021).

15. Репин В. В. Бизнес-процессы. Моделирование, внедрение, управление / В. В. Репин. – М.: Манн, Иванов и Фербер, 2013. – 512 с.

16. Стельмашонок Е.В., Стельмашонок В.Л. Основы компьютерного моделирования бизнес-процессов. Учебное пособие. — Санкт-Петербург: Санкт-петербургский государственный экономический университет, 2019. — 88 с. — ISBN 978-5-7310-4573-5.

17. Стружкин, Н.П. Базы данных: проектирование: Учебник для академического бакалавриата / Н.П. Стружкин, В.В. Годин. - Люберцы: Юрайт, 2016. - 477 с.

18. Шёнталер, Ф. Бизнес-процессы. Языки моделирования, методы, инструменты / Ф. Шёнталер. - М.: Альпина Паблишер, 2019. - 264 с.

19. Яргер, Р.Дж. MySQL и mSQL: Базы данных для небольших предприятий и Интернета / Р.Дж. Яргер, Дж. Риз, Т. Кинг. - М.: СПб: Символ-Плюс, 2015. - 560 с.

20. 6 инструментов CRM с открытым кодом [Электронный ресурс] / Режим доступа: https://te-st.ru/2016/07/11/6-open-source-crm/ (дата обращения: 18.06.2021).

21. Rational Rose, методика RUP и язык UML [Электронный ресурс]. – Режим доступа: www.rational.com. – Дата доступа: 14.08.2021.

22. Rest API [Электронный ресурс] / Режим доступа: https://www.ibm.com/ru-ru/cloud/learn/rest-apis/ (дата обращения: 18.09.2021).

23. Rest-api [Электронный ресурс] / Режим доступа: https://mcs.mail.ru/blog/vvedenie-v-rest-api (дата обращения: 19.08.2021).

24. Rest-api [Электронный ресурс] / Режим доступа: https://www.ibm.com/ru-ru/cloud/learn/rest-apis (дата обращения: 19.08.2021).

25. Symfony [Электронный ресурс] / Режим доступа: https://symfony.com (дата обращения: 18.08.2021).

26. The technological benefits of Symfony in 6 easy lessons [Электронный ресурс]/symfony.com. — Режим доступа: http://symfony.com/six-good-technical-reasons (дата обращения: 09.09.2021).

27. Zurmo [Электронный ресурс] / Режим доступа: https://hellip.com/ru/product/zurmo.html (дата обращения: 18.06.2021).

Приложение А

Исходный код приложения

```
<?php
namespace App\Entity;
use ApiPlatform\Core\Annotation\ApiResource;
use App\Repository\CustomerRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
/**
 * @ORM\Entity(repositoryClass=CustomerRepository::class)
 * @ApiResource()
 */
class Customer
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;
    /**
     * @ORM\Column(type="string", length=255)
     */
    private $fullname;
    /**
     * @ORM\Column(type="string", length=255)
     */
    private $address;
```

```
        /**
         * @ORM\Column(type="string", length=20, nullable=true)
         */
        private $phone;
        /**
         * @ORM\Column(type="string", length=100)
         */
        private $email;
        /**
         * @ORM\Column(type="string", length=255, nullable=true)
         */
        private $access_info;
        /**
         *@ORM\OneToMany(targetEntity=Order::class,
mappedBy="customer")
         */
        private $orders;
        public function __construct()
        {
            $this->orders = new ArrayCollection();
        }
        public function getId(): ?int
        {
            return $this->id;
        }
        public function getFullname(): ?string
        {
            return $this->fullname;
```

```
}
public function setFullname(string $fullname): self
{
   $this->fullname = $fullname;
   return $this;
}
public function getAddress(): ?string
{
   return $this->address;
}
public function setAddress(string $address): self
{
   $this->address = $address;
   return $this;
}
public function getPhone(): ?string
{
   return $this->phone;
}
public function setPhone(?string $phone): self
{
   $this->phone = $phone;
  return $this;
}
public function getEmail(): ?string
{
   return $this->email;
}
```

```
public function setEmail(string $email): self
{
    $this->email = $email;
    return $this;
}
public function getAccessInfo(): ?string
{
    return $this->access_info;
}
public function setAccessInfo(?string $access_info): self
{
    $this->access_info = $access_info;
    return $this;
}
/**
 * @return Collection|Order[]
 */
public function getOrders(): Collection
{
    return $this->orders;
}
public function addOrder(Order $order): self
{
    if (!$this->orders->contains($order)) {
        $this->orders[] = $order;
        $order->setCustomer($this);
    }
    return $this;
```

```
    }
    public function removeOrder(Order $order): self
    {
        if ($this->orders->removeElement($order)) {
            // set the owning side to null (unless already changed)
            if ($order->getCustomer() === $this) {
                $order->setCustomer(null);
            }
        }
        return $this;
    }
}
<?php
namespace App\Entity;
use ApiPlatform\Core\Annotation\ApiResource;
use App\Repository\OrderRepository;
use Doctrine\ORM\Mapping as ORM;
/**
 * @ORM\Entity(repositoryClass=OrderRepository::class)
 * @ORM\Table(name="`order`")
 * @ApiResource()
 */
class Order
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
```

```
     */
    private $id;
    /**
     * @ORM\Column(type="datetime")
     */
    private $date;
    /**
     * @ORM\Column(type="float")
     */
    private $summ;
    /**
     * @ORM\Column(type="boolean")
     */
    private $paystatus;
    /**
     *                          @ORM\ManyToOne(targetEntity=Customer::class,
inversedBy="orders")
     * @ORM\JoinColumn(nullable=false)
     */
    private $customer;
    /**
     *                  @ORM\OneToOne(targetEntity=ApplicationForm::class,
mappedBy="orders", cascade={"persist", "remove"})
     */
    private $application_form_id;
    /**
     *                              @ORM\ManyToOne(targetEntity=User::class,
inversedBy="orderitem")
```

```
 */
private $user;
public function getId(): ?int
{
   return $this->id;
}

public function getDate(): ?\DateTimeInterface
{
   return $this->date;
}
public function setDate(\DateTimeInterface $date): self
{
   $this->date = $date;
   return $this;
}
public function getSumm(): ?float
{
   return $this->summ;
}
public function setSumm(float $summ): self
{
   $this->summ = $summ;
   return $this;
}
public function getPaystatus(): ?bool
{
   return $this->paystatus;
```

```
    }
    public function setPaystatus(bool $paystatus): self
    {
        $this->paystatus = $paystatus;
        return $this;
    }
    public function getCustomer(): ?Customer
    {
        return $this->customer;
    }
    public function setCustomer(?Customer $customer): self
    {
        $this->customer = $customer;
        return $this;
    }
    public function getApplicationFormId(): ?ApplicationForm
    {
        return $this->application_form_id;
    }
    public function setApplicationFormId(ApplicationForm
$application_form_id): self
    {
        // set the owning side of the relation if necessary
        if ($application_form_id->getOrders() !== $this) {
            $application_form_id->setOrders($this);
        }
        $this->application_form_id = $application_form_id;
        return $this;
```

```
    }
    public function getUser(): ?User
    {
        return $this->user;
    }
    public function setUser(?User $user): self
    {
        $this->user = $user;
        return $this;
    }
}
```