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

Кафедра Прикладная математика и информатика
(наименование)

09.04.03 Прикладная информатика
(код и наименование направления подготовки)

Управление корпоративными информационными процессами
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Методы и модели интеграции веб-приложений с корпоративной информационной системой предприятия»

Обучающийся М.Ю. Сибирцев
(Инициалы Фамилия) (личная подпись)

Научный руководитель д.т.н., доцент, С.В. Мкртычев
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Оглавление

Введение

«Интеграция веб-приложений с корпоративной информационной системой (КИС) организации – это процесс, позволяющий различным приложениям обмениваться данными и функциональностью через API или коннекторы. Интеграция улучшает эффективность, безопасность и качество работы приложений, а также снижает затраты на их поддержку.

Интеграция веб-приложения с КИС предприятия означает связывание веб-приложения с ядром, которое образует ERP-система, которая обеспечивает управление основными бизнес-процессами и данными в разных отделах и функциональных областях. Интеграция веб-приложения с ERP позволяет обеспечить единый доступ к информации, улучшить производительность, снизить издержки и повысить удовлетворенность клиентов.

При эффективной интеграции веб-приложений повышается доступность данных КИС, их согласованность и общая эффективность работы. В конечном счете, это позволяет организациям максимально использовать преимущества и возможности своей КИС, обеспечивая им важное конкурентное преимущество в своей отрасли» [10].

Разработка эффективной модели интеграции внешнего веб-приложения и КИС предприятия, в основу которой положены современные методы интеграции компонентов КИС, представляет актуальность и научно-практический интерес.

Объектом настоящего исследования является процесс интеграции веб-приложений с КИС предприятия.

Предметом исследования являются методы и модели интеграции веб-приложений с КИС предприятия.

Целью работы является исследование методов и разработка моделей интеграции веб-приложений с КИС предприятия.

Для достижения цели необходимо решить следующие задачи:

- произвести анализ состояния исследований и разработок в области интеграции веб-приложений с КИС предприятия;
- произвести анализ методологических подходов к интеграции веб-приложений с КИС предприятия;
- разработать модели интеграции веб-приложений с КИС предприятия;
- выполнить апробацию и оценить эффективность предлагаемых проектных решений.

Гипотеза исследования: применение модели интеграции, в основу которой положены современные методы интеграции компонентов КИС, обеспечит повышение эффективности интеграции веб-приложения с КИС предприятия.

Методы исследования. В процессе исследования использованы следующие подходы и методы: методы интеграции веб-приложений с КИС предприятия, методы и технологии построения КИС предприятий.

Новизна исследования заключается в разработке эффективных моделей интеграции веб-приложений с КИС предприятия.

Практическая значимость исследования заключается в возможности практического применения предлагаемых моделей для интеграции веб-приложений с КИС предприятия. Теоретической основой диссертационного исследования являются научные труды российских и зарубежных ученых и специалистов, занимающихся проблемами интеграции веб-приложений с КИС предприятия.

Основные этапы исследования: исследование проводилось с 2022 по 2024 год в несколько этапов.

На первом (констатирующем) этапе формулировалась тема исследования, выполнялся сбор информации по теме исследования из различных источников, проводилась формулировка гипотезы, определялись постановка цели, задач, предмета исследования, объекта исследования и выполнялось определение проблематики данного исследования.

Второй этап – теоретический. В ходе проведения данного этапа

осуществлялся анализ методологических подходов к интеграции веб-приложений с КИС предприятия, разработана модель интеграции веб-приложений с КИС предприятия, опубликована статья по теме исследования в научном сборнике.

Третий этап – практический. В ходе проведения данного этапа проводилась апробация предлагаемых проектных решений, произведена оценка их эффективности, сформулированы выводы о полученных результатах по проведенному исследованию.

На защиту выносятся:

- модели интеграции веб-приложений с КИС предприятия;
- результаты апробации и оценки эффективности предлагаемых проектных решений.

По теме исследования опубликована 1 статья:

Сибирцев М.Ю. Модель интеграции веб-приложения с корпоративной информационной системой предприятия // Вестник научных конференций. 2024. №3-1 (103). С. 87-88.

Диссертация состоит из введения, четырех глав, заключения и списка литературы.

В первой главе дан анализ современного состояния исследований и разработок в области интеграции веб-приложений с КИС предприятия.

Во второй главе дан анализ методологических подходов к интеграции веб-приложений с КИС предприятия.

Третья глава посвящена разработке моделей интеграции веб-приложений с КИС предприятия.

В четвертой главе выполнены апробация предлагаемых проектных решений и оценка их эффективности.

В заключении приводятся результаты исследования.

Работа изложена на 72 страницах и включает 35 рисунков, 8 таблиц и 43 источника.

Глава 1 Анализ состояния исследований и разработок в области интеграции веб-приложений с корпоративной информационной системой предприятия

Взрыв Интернета повлиял на многие аспекты стиля работы в производственной отрасли. Однако все указывает на то, что революция все еще находится в зачаточном состоянии. Производственным предприятиям необходимо перестроить свою рабочую модель, чтобы сохранить конкурентоспособность в эпоху глобального производства.

Планирование ресурсов предприятия (ERP) с поддержкой Интернета, основанной на интеграции веб-приложения с КИС предприятия, имеет решающее значение для непрерывного развития производственного предприятия.

«Целью интеграции является создание единого информационного пространства. Это означает, что в обеих интегрируемых системах должна содержаться непротиворечивая информация (то есть данные одной системы не должны вступать в конфликт с данными другой, не должно быть ситуаций, когда что-то «не сходится»)» [6].

Проблематика интеграции веб-приложений с КИС предприятия широко рассмотрена в работах ученых и специалистов таких ИТ-вендоров, как Microsoft, Amazon, SAP, 1С и др.

Как показал анализ, основными областями применения интеграции веб-приложений с КИС предприятия являются:

- «устранение информационного дисбаланса: единая база ERP-системы, охватывающая весь поток информации из разных структур организации, исключает возможность информационной несовместимости системы, что существенно повышает качество информации и дает преимущества при принятии решений;
- интеграция интернет-магазина с КИС;
- отслеживание заказов через Интернет» [6];

– создания клиентских порталов;

– пользовательские инструменты управления данными;

– пользовательские системы отчетности и др.

Существует несколько методов интеграции веб-приложения с ERP в зависимости от архитектуры, технологии и требований к безопасности.

Так, в работе [24] авторы предлагают модель внедрения, называемую веб-ориентированной объектно-ориентированной моделью (WOOM), для внедрения системы ERP в глобальной перспективе, которая обеспечивает структуру, канал и интерфейс, необходимые для создания веб-системы ERP в глобальной бизнес-среде.

Предлагаемая модель направлена на то, чтобы предоставить производителям основу для систематического внедрения ERP.

Используя передовые методы разработки, была реализована сложная, но простая в использовании модель построения, которая обеспечивает комплексное решение для всех областей глобальной системы ERP (рисунок 1).

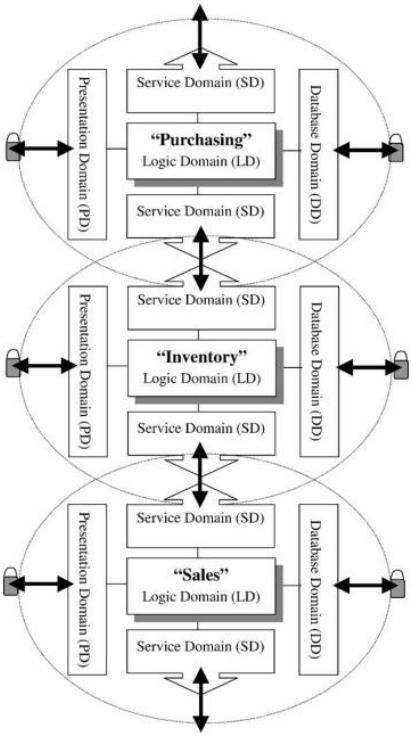


Рисунок 1 – Модель взаимодействия между WOOM-объектами

«Одним из наиболее распространенных и современных способов является использование API (Application Programming Interface) – интерфейсов прикладного программирования, которые позволяют веб-приложениям взаимодействовать и передавать данные друг другу через HTTP-протокол.

Этот способ интеграции отлично подойдет компаниям, использующим популярные продукты с API. Интеграция ERP API с платформами электронной коммерции обеспечивает лучший доступ ко всем необходимым данным с онлайн-рынков. Системы ERP обычно отвечают за несколько операций, которые помогают розничным торговцам автоматизировать свои бизнес-процессы. Для этого им необходимо уметь работать с различными типами данных и реализовывать их с помощью своих систем» [6].

Существуют два основных стиля API – SOAP (Simple Object Access Protocol) и REST (Representational State Transfer), которые имеют различные преимущества и недостатки [3].

SOAP – это стандартизированный и надежный протокол, который использует формат XML для обмена данными. Он подходит для сложных транзакций и высоких требований к безопасности, но он также более тяжеловесный и сложный в разработке и поддержке.

REST – это гибкий и легкий стиль, который использует разные форматы данных, такие как JSON, XML или HTML. Он подходит для простых и частых запросов, но он также менее стандартизированный и надежный.

Выбор между SOAP и REST зависит от конкретных целей и характеристик веб-приложения и ERP-системы, а также от наличия готовых коннекторов или библиотек для интеграции. В любом случае, интеграция веб-приложения с ERP требует тщательного планирования, анализа, тестирования и поддержки, чтобы обеспечить эффективность и надежность обмена данными между системами.

В работе [34] отмечено, что современный деловой мир сталкивается с проблемами доступа к своим фрагментированным данным для создания точного и последовательного представления своих основных

информационных активов по всему предприятию для принятия бизнес-решений и осуществления операций. «Чтобы предоставлять своевременную, надежную и целостную информацию различным заинтересованным сторонам, предприятие должно использовать перспективную архитектуру интеграции данных. Подход веб-сервисов, основанный на сервисно-ориентированной архитектуре, обеспечивает гибкость бизнеса за счет слабой связанности и возможности повторного использования активов данных. Он позволяет приложениям и бизнес-процессам получать доступ к соответствующим данным последовательно и точно, когда бы они ни были нужны, независимо от того, где и в какой форме они находятся. В этой статье реализуются веб-сервисы на основе SOAP для интеграции данных между веб-порталом и системой ERP. Тестирование производительности также проводится для изучения поведения веб-сервисов при различных нагрузках» [34].

Архитектура программной системы состоит из ее структуры, разложения на компоненты, их интерфейсов и взаимосвязей. Архитектура реализации показывает поток вызова веб-сервисов от клиента (VIOLA) к серверу (VERP) и ответ сервера в виде запросов и ответов (рисунок 2).

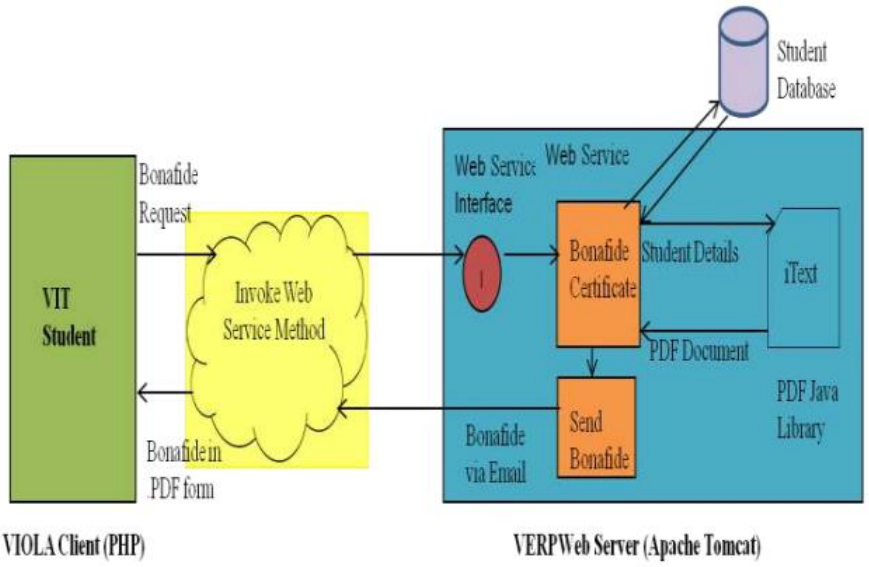


Рисунок 2 – Архитектура программной системы на основе сервисно-ориентированной архитектуры

В работе [33] описана интеграция OpenAI с веб-приложениями React для ускорения процессов компании и повышении точности прогнозов продаж.

Интеграция OpenAI с ERP-системой включает в себя несколько шагов, обеспечивающих плавную и эффективную интеграцию (рисунок 3).

Рисунок 3 – Процесс интеграции OpenAI с ERP-системой

Шаг 1. Проконсультируйтесь с компанией по разработке ReactJS ИЛИ компанией по разработке AI. Когда вы рассматриваете возможность интеграции OpenAI в веб-приложение React для улучшения вашей ERP-системы, важно сотрудничать с первоклассной компанией по разработке программного обеспечения.

Шаг 2. Подготовка набора данных для интеграции API OpenAI

Успех модели прогнозирования OpenAI зависит от наличия высококачественных данных. Подготовка данных гарантирует, что OpenAI имеет доступ к актуальным и точным данным, необходимым для точного прогнозирования продаж. Правильно подготовленные данные могут привести

к снижению количества ошибок, принятию обоснованных решений и положительному возврату инвестиций.

Шаг 3. Выберите модель OpenAI для прогнозирования продаж. Выбор подходящего алгоритма машинного обучения или статистической модели для прогнозирования продаж имеет важное значение.

Шаг 4. Точная настройка модели для интеграции в ERP-систему. Точная настройка важна для прогнозирования продаж, поскольку модель требует оптимизации для конкретной задачи. Прогнозирование продаж включает в себя прогнозирование будущих продаж на основе исторических данных, которые могут отличаться от других типов данных и задач, на которых обучалась модель.

Шаг 5. Интеграция модели OpenAI для улучшения прогнозирования продаж. После точной настройки модели ее можно легко интегрировать в вашу ERP-систему для повышения функциональности и производительности.

Шаг 6. Тестирование и развертывание интегрированного OpenAI в ERP-систему. Чтобы обеспечить правильную интеграцию и использование модели OpenAI в вашей ERP-системе, решающее значение имеет тщательный этап тестирования. Команда контроля качества проводит тесты, чтобы убедиться, что прогнозы модели правильно интегрированы в ERP-систему. Это предполагает сравнение прогнозов модели с фактическими данными о продажах и оценку точности прогнозов.

В работе [40] описано решение, позволяющее улучшить дизайн и удобство использования данных инвентаризации с помощью обновленного интерфейса веб-сайта, интегрированного в ERP-систему. Этот интерфейс соединяет данные с Интернетом в режиме реального времени. Этот интерфейс также работает рука об руку с сайтом WordPress, ориентированным на потребителя, который авторы также переработали в рамках этого проекта.

В работе [43] предлагается модель интеграции веб-портала с ERP-системой, которая состоит из следующих компонентов (рисунок 4):

− ERP-система;

- ETL;
- хранилище данных «Одноверсионное хранилище данных»;
- витрины данных;
- веб-портал.

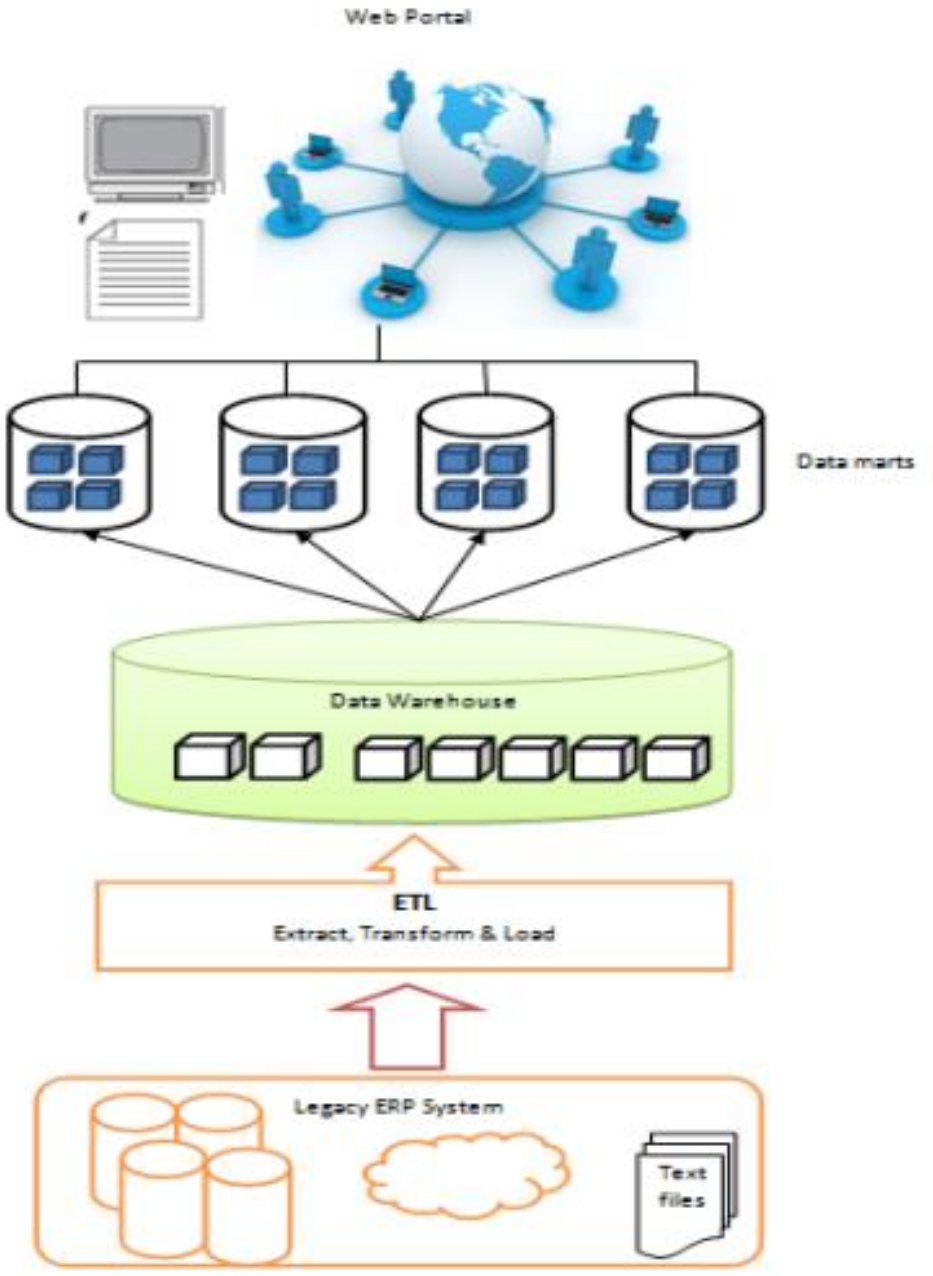


Рисунок 4 – Модель интеграции веб-портала с ERP-системой

Веб-портал используется для поддержки принятия решений на различных организационных уровнях путем предоставления таких функций,

как настройка, персонализация, поддержка совместной работы и уведомлений, которые помогают менеджерам принимать правильные решения с помощью Интернета.

В работе [10] рассматривается подход к веб-интеграции с КИС для создания единого информационного пространства (рисунок 5).

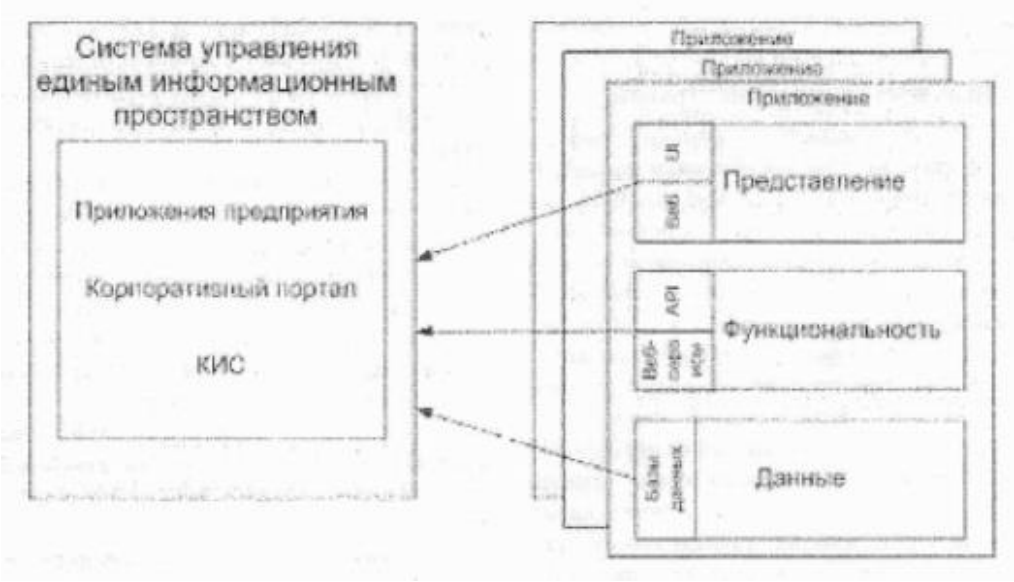


Рисунок 5 – Схема интеграции веб-портала с КИС

Сформулированы принципы веб-интеграции, среди которых выделяется использование технологии SOA. Отмечается, что веб-портал становится удобной платформой для организации работы веб-сервисов.

В работе [6] представлены модели интеграции веб-приложения с системой товарного учета (СТУ), которая образует ядро КИС:

- западная модель. «Для обеспечения гибкой и надежной интеграции в этих случаях применяется т.н. Middleware, промежуточный слой, обеспечивающий обмен данными между разрозненными подсистемами, такими как eCommerce, ERP, Web-службами и т.д. Объединяя многие форматы и протоколы передачи данных, предоставляя возможности преобразования одних форматов в другие, middleware также обеспечивает высокую степень надежности и

гибкости интегрированной системы, поскольку изменение одного ее компонента повлечет за собой лишь изменение правил взаимодействия с ним (рисунок 6)» [6].

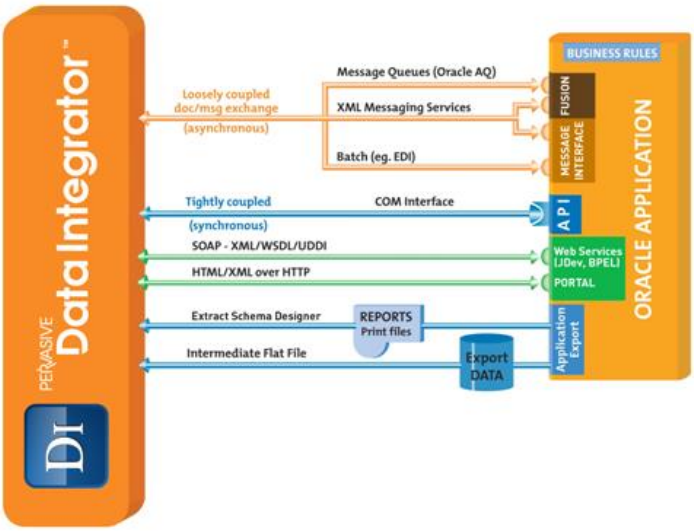


Рисунок 6 – Архитектура взаимодействия Pervasive Data Integrator Middleware с приложениями

Данная модель используется для интеграции веб-приложений с зарубежными ERP-системами;

– отечественная модель. «Для обеспечения взаимодействий на всех трех уровнях мы используем т.н. интерфейсный слой. Этот слой позволяет выгрузить необходимые для синхронизации данные в необходимом формате, отвечающем требованиям бизнес-логики. При этом мы не используем раздутые XML-стандарты вроде CommerceML, а адаптируем протоколы под нужды конкретной модели (рисунок 7)» [6].

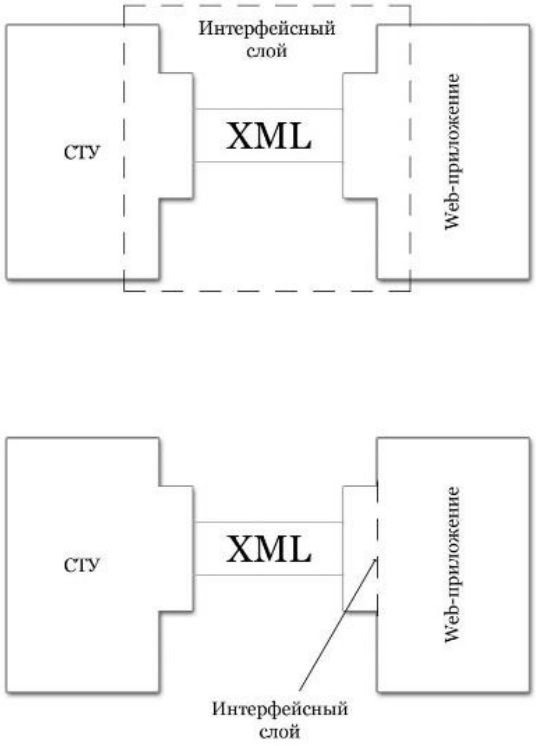


Рисунок 7 – Схема интеграции с помощью интерфейсного слоя

Данная модель используется для интеграции веб-приложений с ERP-системами на платформе 1С: Предприятие 8 [1].

В работе [2] описан механизм интеграции веб-сайта с 1С: Предприятие 8.

«Отмечено, что итоговый метод подбирается профессиональным разработчиком, который обязан учесть все специфические детали и нюансы. Нередко окончательный результат зависит от квалификации и профессиональных знаний исполнителя, а также мощностных возможностей используемых серверов.

Масштаб баз данных, количество и уровень занятости работников также выступают важными составляющими всего процесса.

Наиболее распространенный вариант – обмен в формате CommerceML, предполагающий применение встроенных во многие конфигурации опций 1С. Делает возможным настраивание интеграции по различным схемам. Полная

автоматизация процесса исключена» [2].

На рисунке 8 показана схема интеграции веб-приложения на основе 1С-Битрикс с КИС медучреждения, ядром которой является решение на платформе «1С: Медицина» [1].

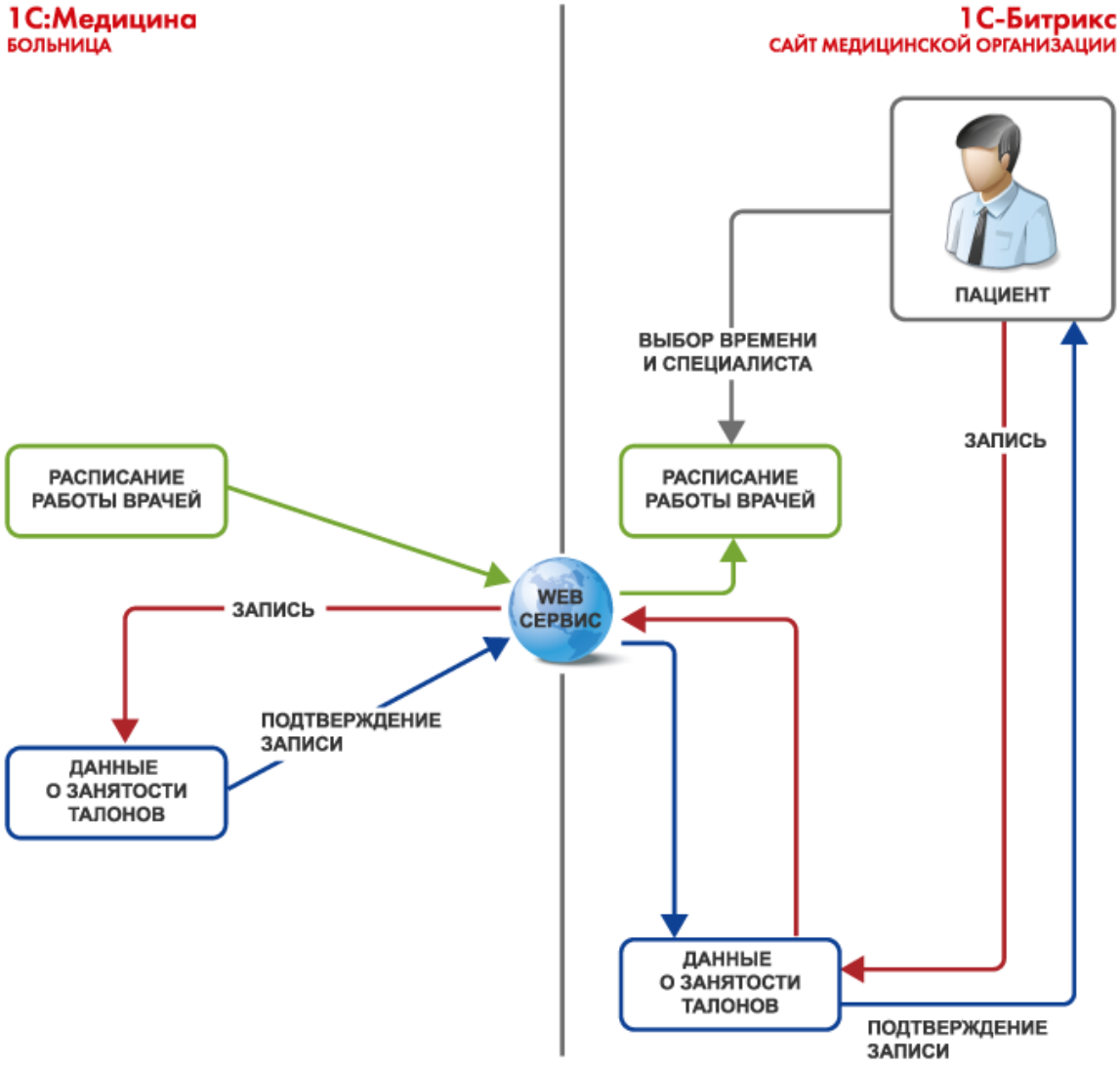


Рисунок 8 – Схема интеграции ИТ-решения «1С:Медицина» с веб-приложением на основе 1С-Битрикс

«Обмен данными происходит в формате MedML, который используется для передачи сведений о деятельности медицинской организации на сайт и для самостоятельной записи пациентов на прием в режиме онлайн» [8].

Внедрение систем ERP является высоко-рискованной деятельностью

для каждой компании.

Все внедрения порождают множество открытых вопросов, касающихся двух основных проблем данных и процесса.

В статье [28] основное внимание уделяется проблеме данных и исследуется, как автоматизировать некоторые ключевые действия по внедрению.

«Авторы статьи предлагают решение выявленных проблем в виде веб-приложения для управления данными.

Хотя основное внимание авторов сосредоточено на академической перспективе внедрения систем ERP, полученные результаты также применимы в промышленности для автоматизации ключевых действий по внедрению настроек базы данных, что делает научный вклад этой статьи более ценным» [28].

Вместе с тем, необходимо констатировать недостаточность современных исследований в области интеграции веб-приложений с КИС предприятия, что подтверждает актуальность темы настоящего исследования.

Выводы по главе 1

В результате проделанной работы по главе 1 были сделаны следующие выводы:

- планирование ресурсов предприятия (ERP) с поддержкой Интернета, основанной на интеграции веб-приложения с КИС предприятия, имеет решающее значение для непрерывного развития производственного предприятия. Веб-приложение используется для поддержки принятия решений на различных организационных уровнях путем предоставления таких функций, как настройка, персонализация, поддержка совместной работы и уведомлений, которые помогают менеджерам принимать правильные решения с помощью Интернета;
- проблематика интеграции веб-приложений с КИС предприятия

широко рассмотрена в работах отечественных и зарубежных ученых и специалистов ведущих вендоров ПО;

- одним из наиболее распространенных и современных способов является использование API (Application Programming Interface) – интерфейсов прикладного программирования, которые позволяют веб-приложениям взаимодействовать и передавать данные друг другу через HTTP-протокол;
- выбор между SOAP и REST зависит от конкретных целей и характеристик веб-приложения и ERP-системы, а также от наличия готовых коннекторов или библиотек для интеграции. В любом случае, интеграция веб-приложения с ERP требует тщательного планирования, анализа, тестирования и поддержки, чтобы обеспечить эффективность и надежность обмена данными между системами.

Таким образом, анализ литературы и источников подтвердил интерес ученых и специалистов к проблеме формирования интеграции веб-приложений с КИС предприятия на основе современных ИТ.

Глава 2 Анализ подходов и методов интеграции веб-приложений с корпоративной информационной системой предприятия

«В 1990-х годах компании сосредоточились на внедрении систем планирования ресурсов предприятия (ERP) для решения проблем интеграции» [41].

Однако ERP-системы автоматизируют основные бизнес-процессы, не решая базовые бизнес-структуры и процессы. В результате ряд разнородных приложений часто сосуществуют с системами ERP.

Как показывает практика, ERP-системы усилили необходимость интеграции, поскольку существующие системы должны быть объединены с приложениями ERP.

Рассмотрим методы интеграции веб-приложений с КИС предприятия.

2.1 Методы системной интеграции

Системная интеграция надежно объединяет функциональность разнородных приложений и, как показало исследование, приводит к разработке новых стратегических бизнес-решений для предприятий [36].

Цели системной интеграции:

- снижение ручного труда: интеграции позволяют автоматизировать рутинные задачи, которые ранее выполнялись вручную;
- ускорение бизнес-процессов: обмен данными между системами ускоряет процессы и делает их более эффективными;
- сокращение ошибок: автоматическая передача данных уменьшает вероятность ошибок, связанных с ручным вводом.

Интеграция веб-приложений с корпоративными информационными системами – важный аспект для эффективного функционирования предприятий.

Интеграция «точка-точка» (Point-to-point, P2P) – это способ соединения

двух или более программных приложений или систем для беспрепятственного обмена данными. Это помогает организациям устранить разрозненность данных и обеспечивает обмен важной информацией между различными системами.

Интеграция P2P устраняет необходимость в сложном промежуточном программном обеспечении или специальном коде, обеспечивая более эффективный и оптимизированный подход к интеграции данных [20].

Интеграция P2P – это простой подход, при котором две системы напрямую взаимодействуют друг с другом посредством специального кода или API.

Хотя интеграция P2P подходит для небольших сценариев, она становится менее эффективной по мере увеличения сложности и объема интеграции.

Пример схемы интеграция по типу «точка-точка» показан на рисунке 9.

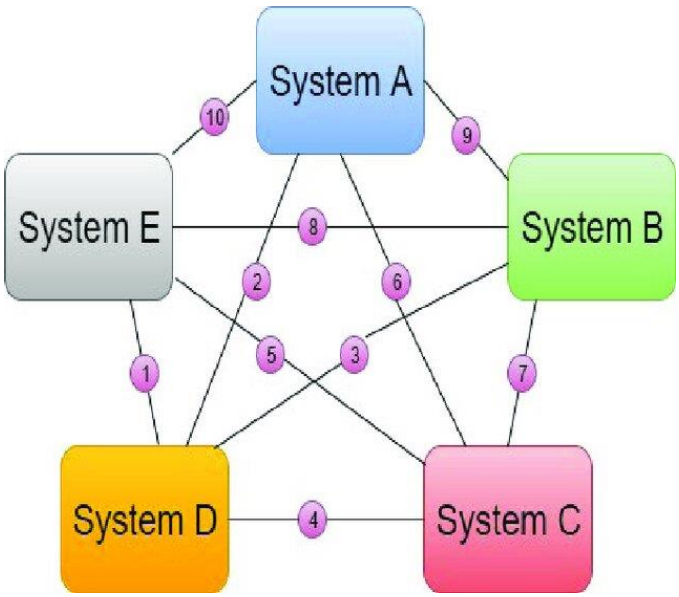


Рисунок 9 – Архитектура интеграции по типу «точка-точка»

Рассмотрим концентраторно-спицевую модель (Hub-and-spoke) на основе единой сервисной шины (ESB).

ESB – это централизованная платформа для обмена данными между

системами. Обеспечивает более гибкую архитектуру [19].

Как показано на рисунке 10, приложения косвенно связаны через ESB, а не напрямую друг с другом. ESB отвечает за всю встроенную логику, необходимую для взаимодействия/интеграции структур.

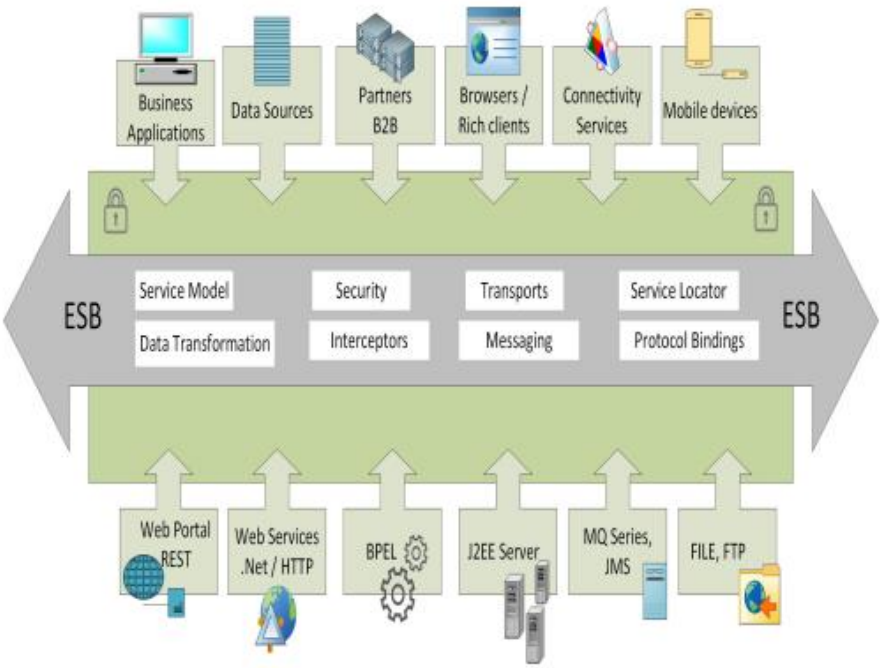


Рисунок 10 – Архитектура единой сервисной шины

Рассмотрим интеграцию через ESM (Enterprise Service Management)-систему. В рамках этого метода происходит интеграция приложений, данных и людей, вовлечённых в бизнес-процессы. Здесь также используется единая сервисная шина (ESB), которая обеспечивает гибкую интеграцию.

В основе концепции ESM лежит сервисный подход, по которому все подразделения компании рассматриваются как поставщики услуг для внутренних или внешних клиентов [7]:

- ESM-платформа позволяет применить лучшие практики управления услугами (ITIL, VeriSM) во всех подразделениях компании: ИТ-отдел, отдел кадров, административно-хозяйственный отдел, финансовой, юридической службах и др.;

- «платформа SimpleOne помогает выстроить бизнес-процессы в компании и систематизировать работу разных подразделений в единой цифровой среде» [7];
- благодаря внедрению ESM-платформы растет удовлетворённость потребителей услуг, минимизируются риски простоя основных бизнес-процессов (рисунок 11).

Рисунок 11 – Архитектура концепции ESM

«Системная интеграция – это способ обеспечить обмен информацией между двумя или более программами. Путем интеграции можно добиться того, чтобы функциональность одной системы стала доступной в другой» [36].

2.2 Метод интеграции на основе сервисно-ориентированной архитектуры

Рассмотрим интеграцию на основе сервисно-ориентированной архитектуры (рисунок 12) [23].

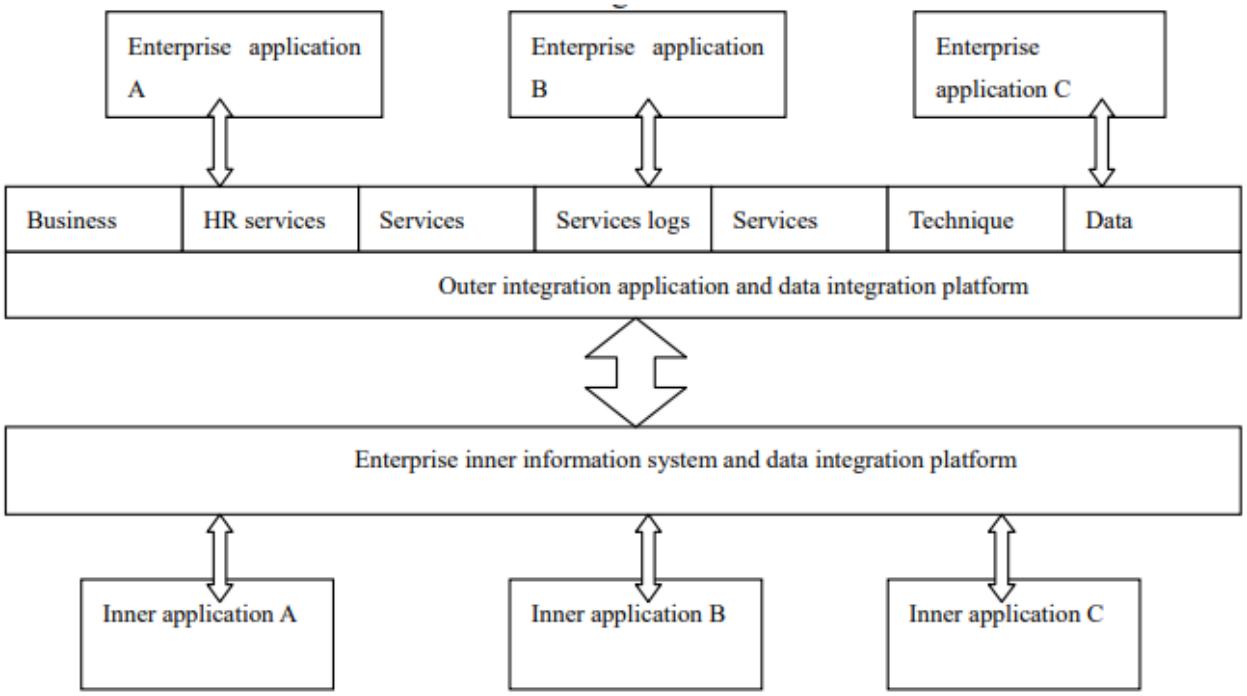


Рисунок 12 – Пример интеграции основе сервисно-ориентированной архитектуры

Внутренние прикладные модули предприятия интегрированы в прозрачную платформу для устранения различий внутри модуля, между модулями и методами хранения данных.

Основные функции следующие:

Приложение А. Общность взаимодействия: поскольку бизнес предприятия постоянно меняется и часто между различными информационными системами модулей, взаимодействие хорошей общности является гарантией того, что преобразование старого и нового интерфейса не повлияет на использование информационных систем. В конечном итоге

реализуется взаимосвязь всей системы на основе существующих информационных систем различных модулей для расширения.

Приложение B. Передача данных: система интеграции приложений может надежно взаимодействовать с внутренними или внешними прикладными системами для достижения безопасного обмена информационными данными подчиненных модулей.

Приложение C. Универсальный интерфейс: информационная система каждого модуля внутри предприятия удобно и быстро подключается к платформе интеграции приложений, удобна для обмена данными между ними. Принят принцип единого стандарта данных и предоставляются унифицированные интерфейсы планирования, и каждый модуль может реализовать соединение и проверку между предприятиями в разных местах.

Приложение D. Публичное управление: включая изолированную связь частных данных и обмен публичными данными, генерацию журнала, совместное использование данных, систему журналов запросов, запросы записей и т. д.

2.3 Методы облачной интеграции

«Платформа интеграции как услуга (iPaaS) – это набор автоматизированных инструментов, которые интегрируют программные приложения, развернутые в различных средах. Крупные предприятия, использующие системы корпоративного уровня, часто используют iPaaS для интеграции приложений и данных, которые находятся локально, а также в общедоступных и частных облаках.

Обычно инструмент iPaaS предоставляет готовые соединители, бизнес-правила, карты и преобразования, которые облегчают разработку приложений и координируют потоки интеграции» [21].

«Некоторые поставщики iPaaS предлагают специальные комплекты разработки для модернизации устаревших приложений и добавления таких

возможностей, как поддержка мобильных устройств, интеграция с социальными платформами и управление бизнес-данными» [35].

Пример интеграции приложений на основе iPaaS показан на рисунке 13.

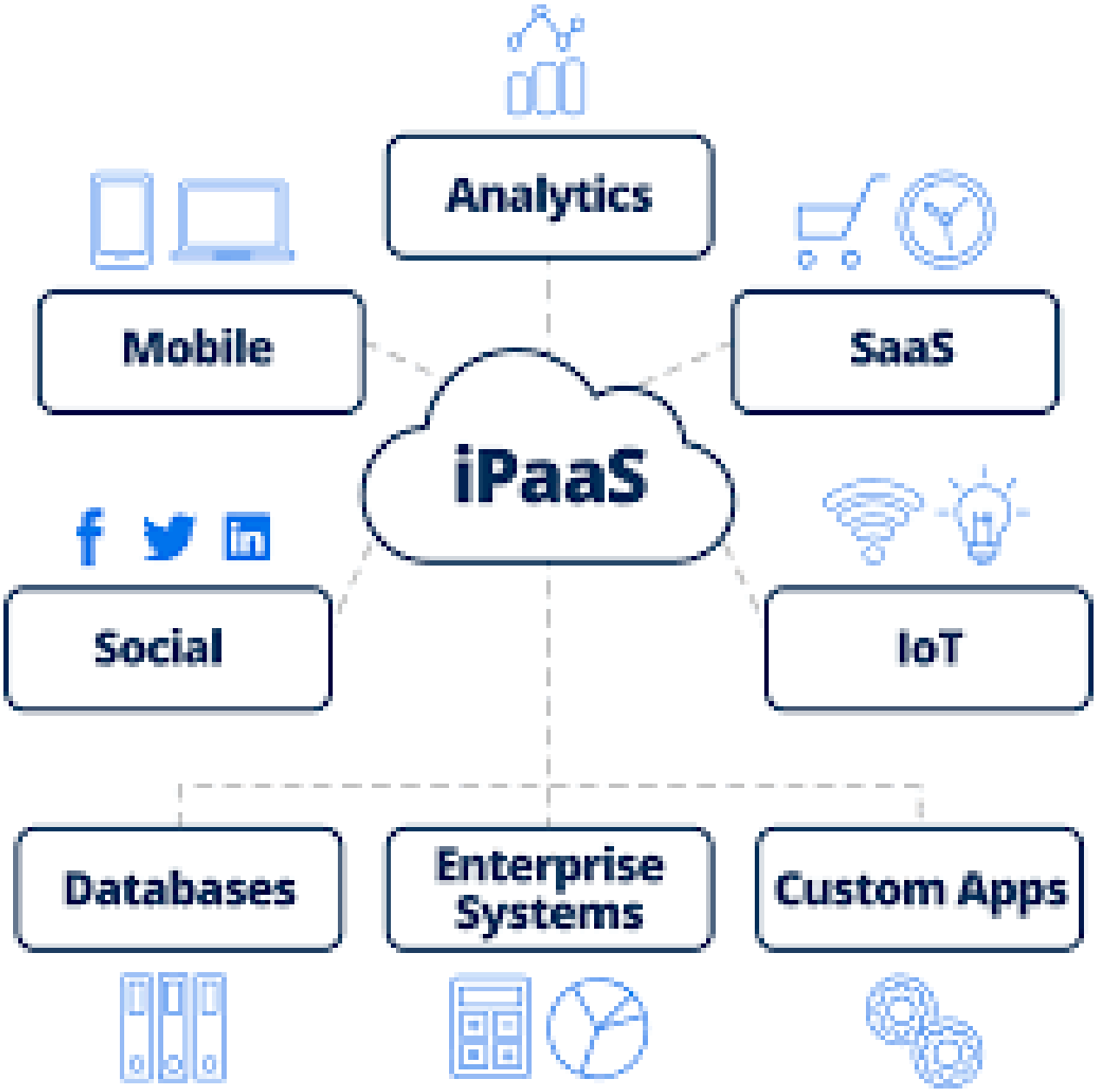


Рисунок 13 – Пример интеграции приложений на основе iPaaS

«SaaS-приложения, как правило, разработаны так, чтобы их было легко настраивать, использовать и развертывать, что делает их привлекательным вариантом для компаний, стремящихся решать конкретные деловые и

административные задачи. Однако простота их использования также побуждает бизнес-группы и отделы покупать SaaS-приложения для удовлетворения потребностей групп и отделов, что может создать часто громоздкую экосистему облачных бизнес-приложений» [42].

Рассмотрим интеграционные решения, построенные по модели iPaaS.

SAP Integration Suite – это iPaaS-платформа интеграции. Она предлагает набор облачных сервисов, позволяющих организациям и группам разработчиков создавать потоки интеграции производственного уровня. Это помогает соединить разнородные системы, интерфейсы, сервисы, облачные или локальные приложения и данные.

SAP Integration Suite предоставляет множество замечательных возможностей [35]:

- облачная интеграция (CI). Возможность облачной интеграции обеспечивает рабочую среду и студию разработки, где вы можете обнаружить существующие потоки интеграции, предоставляемые SAP, а также разработать новые потоки интеграции на основе бизнес-требований. Это также место, где вы можете контролировать интеграцию и управлять арендатором;
- управление API. Управление API дает вам возможность связывать потоки интеграции, службы ODATA или открытые экземпляры соединителей с внешним миром. Вы также можете настроить сертификаты, ключи API. Используя управление API, вы можете монетизировать свои API;
- Open Connectors. SAP Open Connectors – это набор из более чем 150 соединителей для интеграции с приложениями, не относящимися к SAP. Например – TWILIO, STRIPE и т. д.

Пример архитектуры интеграции веб- и мобильных приложений с КИС предприятия на основе решения SAP Integration Suite показан на рисунке 14.

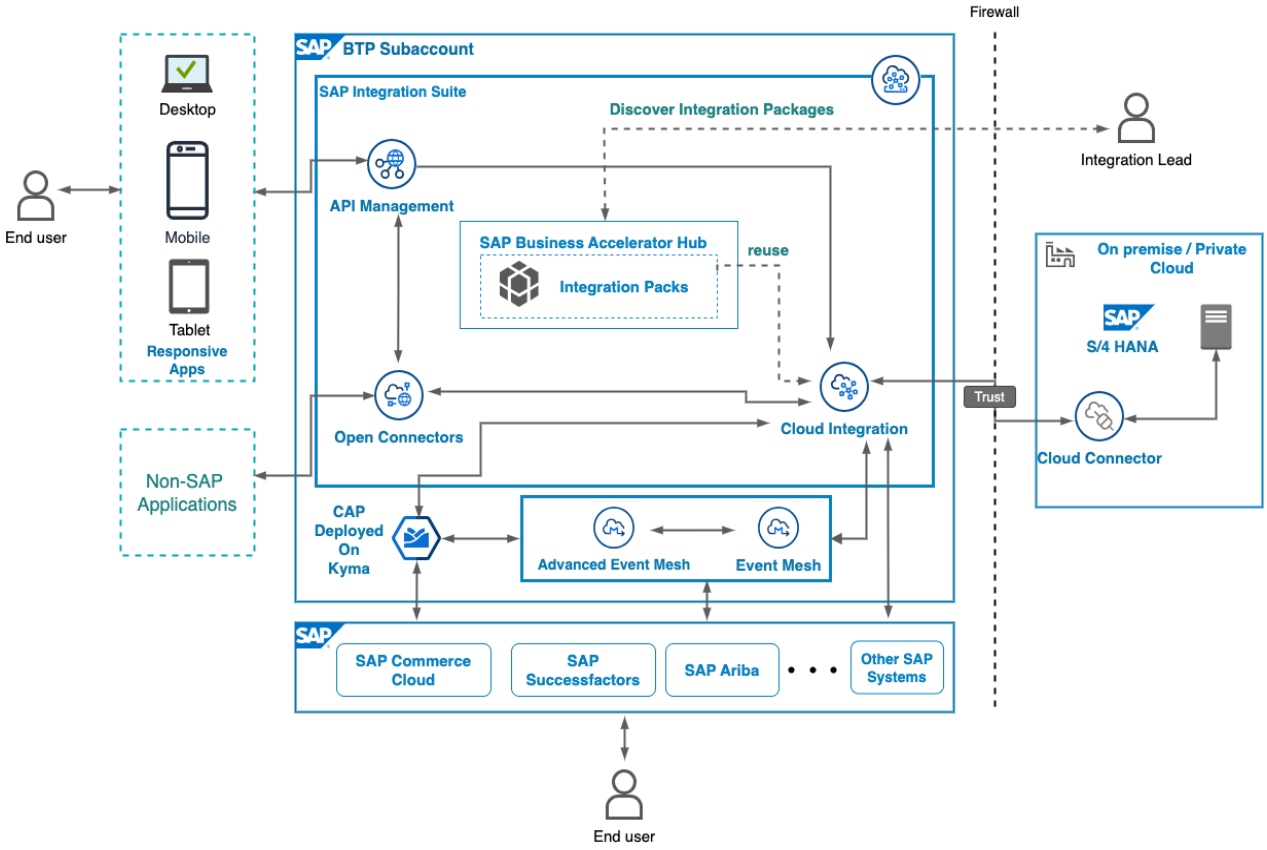


Рисунок 14 – Архитектура интеграции веб- и мобильных приложений с КИС предприятия на основе решения SAP Integration Suite

Celigo iPaas – одна из лучших iPaaS-платформ интеграции данных для подключения приложений, консолидации данных и компьютеризации трудоемких ручных процессов.

Она содержит набор мастеров интеграции, готовых шаблонов и других средств автоматизации, позволяющих значительно упростить процесс принятия решений на основе данных [26].

Вот некоторые из наиболее примечательных особенностей Celigo:

– предоставляет интеграционные приложения платформы — набор готовых полностью управляемых интеграционных приложений, ориентированных на популярные облачные приложения, такие как Amazon, Salesforce, Shopify, NetSuite и Zendesk;

- доступ к собственным шаблонам автоматизации бизнес-процессов, которые позволяют пользователям полностью автоматизировать бизнес-процессы.
- позволяет создавать собственные потоки в рабочей области разработчика Celigo. Имена полей автоматически заполняются из подключенных каналов, таких как Salesforce, поэтому нетехническим пользователям никогда не придется иметь дело с непонятными или запутанными именами идентификаторов при настройке потоков;
- простой пользовательский интерфейс в рабочей области разработчика для извлечения, фильтрации, преобразования и сопоставления данных, а также функцию автоматического сопоставления, предлагающую сопоставление полей на основе сотен отзывов клиентов Celigo. Один щелчок открывает дополнительные инструменты разработчика, такие как JavaScript, для более сложной интеграции;
- функциональность в реальном времени. Такие элементы, как обновление запасов в реальном времени на веб-сайтах электронной коммерции, помогают ограничить перепродажу;
- автоматическое отслеживание потоков позволяет пользователям планировать их на определенные периоды времени для максимальной производительности и аналитической информации.

Архитектура Celigo iPaas показана на рисунке 15.

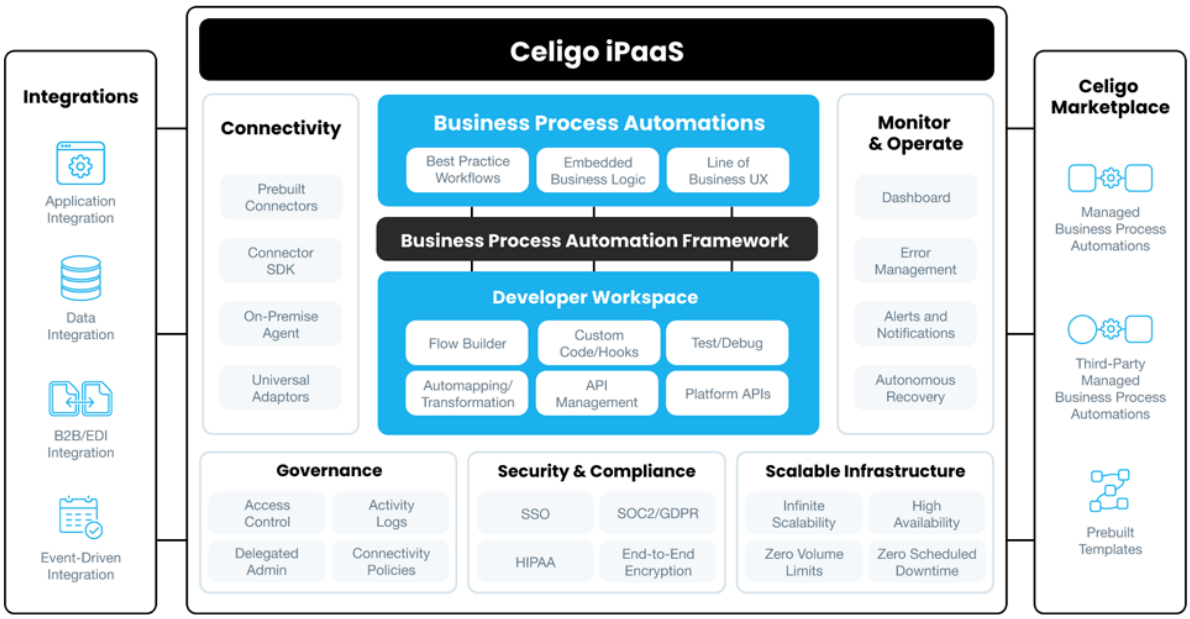


Рисунок 15 – Архитектура Celigo iPaas

Informatica Intelligent Cloud Services – это облачная интеграционная платформа, обеспечивающая плавную интеграцию между ERP-системами и другими приложениями, базами данных и облачными сервисами. Он предлагает широкий спектр соединителей, возможностей преобразования данных и инструментов мониторинга для оптимизации процессов интеграции данных.

Это известное и признанное предприятие iPaaS.

Предлагает интеллектуальную интеграцию на основе AI/ML.

Предлагает ключевые функции управления основными данными и интеграцию с облачной платформой.

Использует модель ценообразования на основе IPU.

Пример архитектуры интеграции веб-приложений с КИС предприятия на основе решения Informatica показан на рисунке 16.

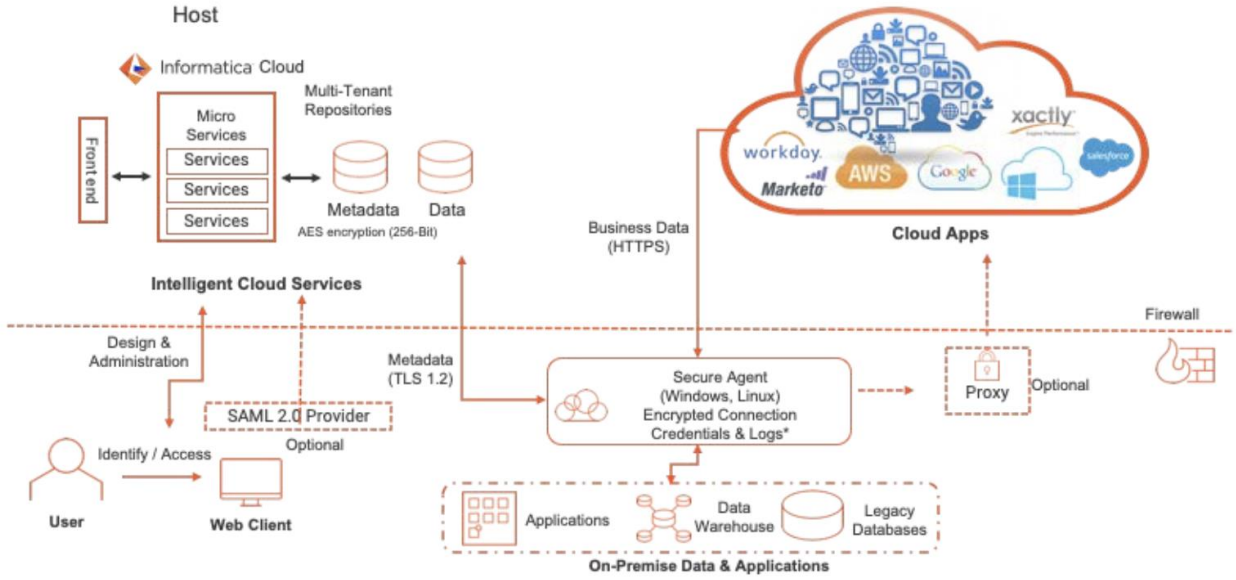


Рисунок 16 – Архитектура интеграции веб-приложений с КИС предприятия на основе решения Informatica

Платформа Entaxy ION – отечественная Low-code платформа для создания интеграционных маршрутов обмена данными между информационными системами [12].

Возможности решения:

– универсальный интерфейс;
– поддержка стандартов (OData);
– расширение с внутренним API;
– множество точек расширения;
– полная поддержка маршрутов Camel;
– поддержка множества протоколов и коннекторов (контейнеры, Kafka, и т.д.);
– наличие шаблонов интеграции корпоративных приложений.

Архитектура решения показана на рисунке 17.

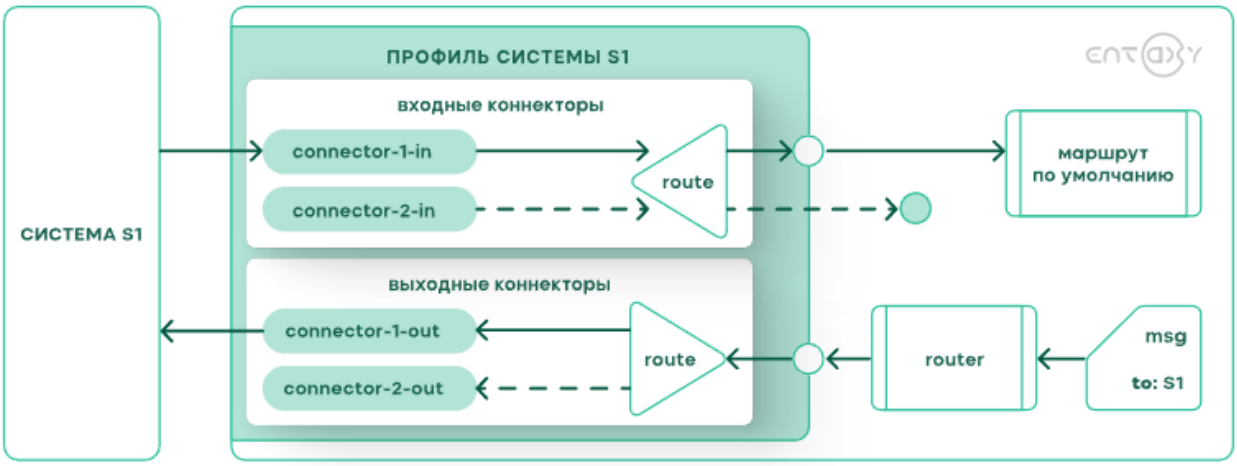


Рисунок 17 – Архитектура платформы Entaxy ION

Функциональная схема построения интеграционного маршрута показана на рисунке 18.

Рисунок 18 – Функциональная схема построения интеграционного маршрута Entaxy ION

Таким образом решения iPaaS действуют как промежуточное программное обеспечение, облегчая интеграцию и связь между различными программными компонентами.

Они устраняют необходимость в сложных интеграциях с индивидуальным кодом, предлагая готовые соединители, API и инструменты, упрощающие процесс интеграции.

«Гибридная интеграционная платформа (Hybrid Integration Platform, HIP) – это усовершенствованная программная платформа, которая обеспечивает плавную интеграцию между различными приложениями, системами и источниками данных независимо от того, находятся ли они локально, в облаке или и там, и там (рисунок 19)» [22].

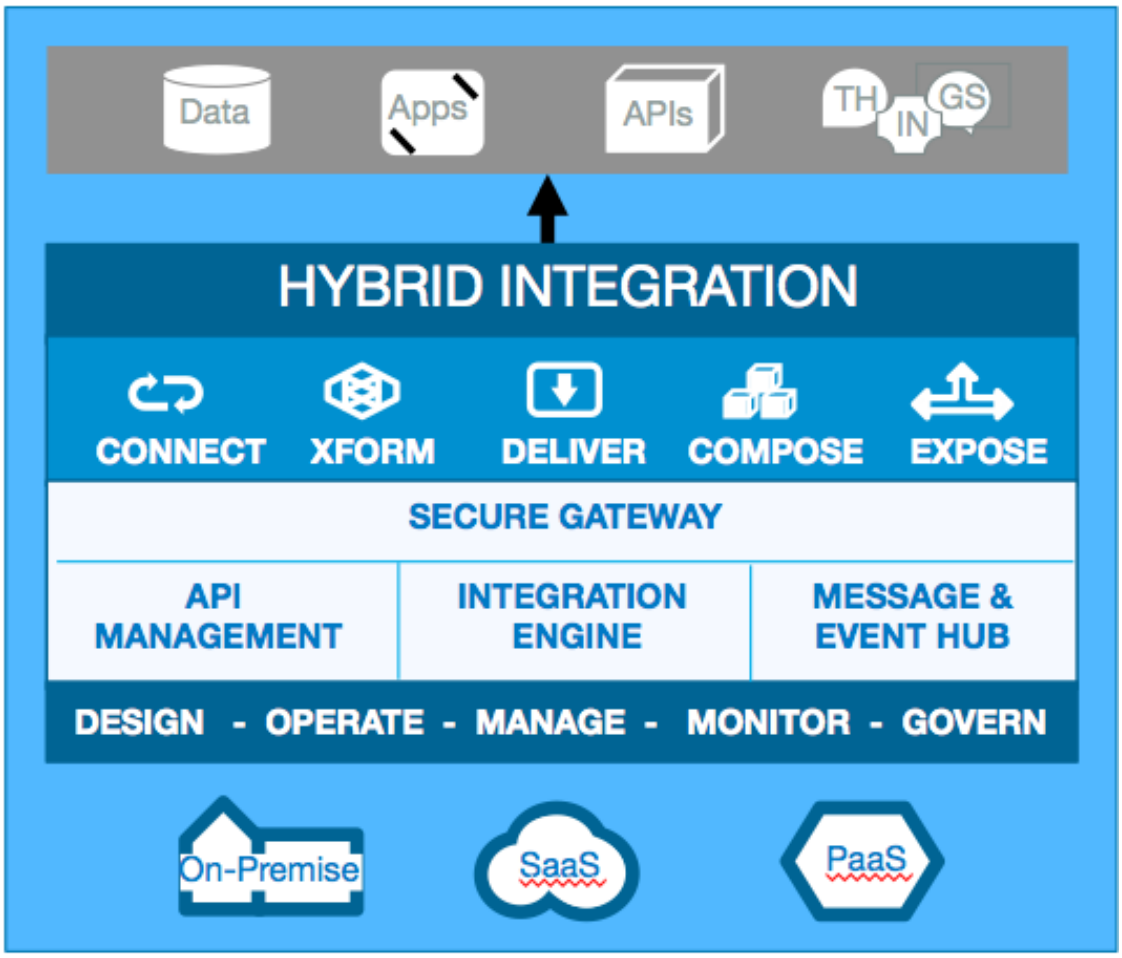


Рисунок 19 – Пример гибридной интеграции приложений

С помощью платформы гибридной интеграции (HIP) организации могут создавать сложные интеграции и управлять ими, автоматизировать бизнес-процессы и получать ценную информацию из своих данных.

2.4 Метод интеграции на основе API баз данных

Метод Database API или API баз данных (интерфейс прикладного программирования) предоставляет набор определенных правил, которые позволяют различным приложениям взаимодействовать с базой данных [29].

Долгое время организации использовали базы данных для хранения критически важных фрагментов данных.

Но по мере изменения экосистемы данных все больше данных перемещалось из баз данных в сами сервисы. Эти сервисы регулярно имеют интерфейсы прикладного программирования API, которые обеспечивают связь между другими программными приложениями и сервисами.

«Сейчас для работы с API создано больше приложений для обработки данных, чем когда-либо, но только потому, что количество сервисов постоянно растет, не означает, что организации по-прежнему не используют базы данных.

API-интерфейсы баз данных появились для устранения разрыва между приложениями, созданными для работы со службами и базами данных.

Метод API баз данных обеспечивает эффективную связь между программными приложениями и базой данных, позволяя приложениям запрашивать данные, находящиеся в базах данных, и манипулировать ими через стандартизированный интерфейс.

API-интерфейсы баз данных работают путем установления соединения между приложением и базой данных» [25].

Это соединение обычно осуществляется с помощью набора стандартизированных инструкций или команд, понятных API. Когда приложение отправляет запрос на доступ к данным или их изменение, API

преобразует этот запрос в формат, понятный базе данных.

Затем база данных обрабатывает этот запрос и возвращает соответствующий ответ обратно API, который, в свою очередь, доставляет его приложению.

Этот процесс гарантирует эффективное извлечение данных, манипулирование ими и их хранение при сохранении целостности и безопасности базы данных (рисунок 20) [39].

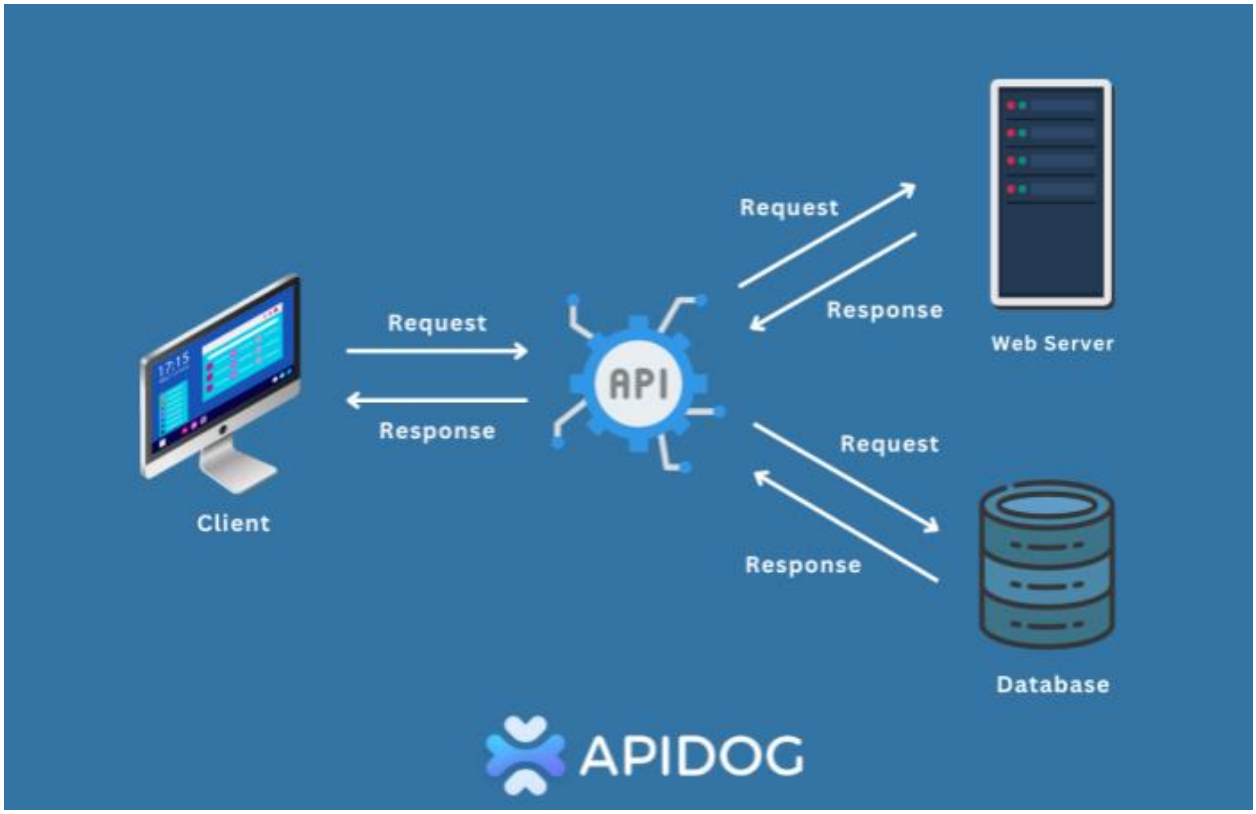


Рисунок 20 – Схема подключения API к БД

Несколько API-интерфейсов баз данных приобрели популярность благодаря своим надежным функциям и возможностям:

– «JDBC (подключение к базе данных Java). Независимость от платформы: JDBC API позволяет Java-приложения взаимодействовать с любой базой данных, совместимой с SQL» [18]. Его платформенно-независимый характер делает его

идеальным для предприятий, работающих в разных системах;

- ODBC (открытое подключение к базе данных). Его широкое использование и поддержка основными поставщиками баз данных гарантируют, что ODBC остается надежным выбором для подключения к базам данных;
- MongoDB Data API. Этот API разработан для обеспечения масштабируемости, поддерживая высокопроизводительный распределенный характер MongoDB;
- Notion API. Интеграция с инструментами повышения производительности: Notion API позволяет разработчикам подключать свои приложения к Notion, универсальной платформе рабочего пространства, обеспечивающей бесперебойный обмен данными между различными инструментами повышения производительности. API Notion разработан с упором на удобство использования, что делает его доступным для более широкого круга разработчиков, включая тех, кто может не специализироваться в управлении базами данных;
- API объектно-реляционного сопоставления (ORM). API-интерфейсы ORM применяют объектно-ориентированную абстракцию при взаимодействии с базой данных (БД) [18].

Подключение API к базам данных означает, что приложения могут извлекать, обрабатывать и хранить данные в режиме реального времени. Это подключение гарантирует, что пользователи получают самую актуальную информацию, а приложения могут предоставлять динамический контент на основе взаимодействий и предпочтений пользователя.

2.5 Выбор метода интеграции веб-приложения с КИС предприятия

Для сравнения характеристик рассмотренных методов интеграции веб-приложений с КИС предприятия составим таблицу 1.

Таблица 1 – Сравнение характеристик методов интеграции веб-приложений с КИС предприятия

Метод	Преимущества	Недостатки
SOA	Сервисы SOA могут быть повторно использованы в разных приложениях и бизнес-процессах. Это сокращает время разработки и уменьшает дублирование функционала. Сервисы могут быть развернуты на различных платформах и масштабироваться независимо друг от друга. Это позволяет распределять нагрузку и эффективно управлять ресурсами. «SOA позволяет легко модифицировать или добавлять новые сервисы без необходимости переделывать всю систему. Это особенно полезно в условиях постоянно изменяющихся требований бизнеса. Сервисы взаимодействуют друг с другом через стандартизированные протоколы (например, HTTP, SOAP, REST), что делает их доступными для различных платформ и языков программирования. Сервисы могут быть реализованы на разных языках программирования и работать на разных платформах, что позволяет выбирать наиболее подходящие технологии для каждой задачи. SOA поддерживает концепцию интеграции различных бизнес-систем и их унификацию. Это способствует лучшему управлению и оптимизации бизнес-процессов» [21].	С увеличением количества сервисов растет сложность их управления, мониторинга и поддержки. Необходимы дополнительные инструменты для оркестрации, балансировки и безопасности сервисов. Внедрение SOA требует значительных вложений в архитектуру, разработку и интеграцию существующих систем. Также может потребоваться перепроектирование существующих приложений для соответствия стандарта SOA. Из-за того, что взаимодействие между сервисами происходит через сеть (часто через HTTP), это может вызывать дополнительные задержки и увеличивать затраты на передачу данных. Так как сервисы взаимодействуют друг с другом по сети, возрастает риск уязвимостей и атак на систему. Необходимо обеспечивать дополнительную защиту данных и управление доступом к сервисам. В распределенной системе, построенной на SOA, может быть сложно тестировать взаимодействие между сервисами, особенно при интеграции с внешними системами. Различные сервисы могут развиваться с разной скоростью, что требует управления версиями сервисов и их совместимостью.

Продолжение таблицы 1

P2P	Простота и эффективность: интеграция P2P проста и эффективна, если нужно интегрировать всего несколько систем. Это позволяет напрямую общаться без посредников. «Повышенная точность и целостность данных: поскольку данные передаются напрямую между системами, снижается вероятность ошибок или несогласованности данных. Гибкость и масштабируемость: интеграцию P2P можно настроить в соответствии с конкретными требованиями, что подходят для небольших сценариев» [21]. Настройка и контроль: разработчики могут контролировать процесс интеграции, адаптируя его к своим потребностям. Экономичность: интеграция P2P часто требует минимальных инвестиций в дополнительные инструменты или платформы.	Повышение сложности и обслуживания: по мере роста числа интегрированных систем управление несколькими двухточечными соединениями становится сложным. Каждое новое добавленное приложение требует дополнительных усилий по интеграции. Отсутствие централизованного управления: в P2P-интеграциях отсутствует центральный узел, что затрудняет соблюдение согласованных стандартов или мониторинг потока данных. Ограниченная масштабируемость и адаптируемость: P2P-интеграции с трудом справляются с эффективной обработкой больших объемов данных. Масштабирование становится затруднительным. Более высокая зависимость от технических знаний: обслуживание и устранение неисправностей требуют специальных навыков. Потенциальные угрозы безопасности: прямые соединения могут привести к уязвимостям, если они не защищены должным образом.

Продолжение таблицы 1

Метод	Преимущества	Недостатки
iPaaS	Синхронизация данных между системами: Ручной ввод или обновление данных в разных системах может привести к ошибкам и снижению эффективности. iPaaS обеспечивает плавную синхронизацию данных между приложениями, уменьшая количество человеческих ошибок и обеспечивая своевременный доступ к точной информации. Возможность избежать проблемы с внутренней интеграцией. Создание интеграций собственными силами — это сложно и ресурсоемко. iPaaS устраняет необходимость в специальной интеграции, предоставляя готовые соединители и API. Разработчики могут сосредоточиться на основной бизнес-логике, а не на обслуживании API. «Масштабируемость и гибкость: iPaaS поддерживает масштабируемость по мере роста бизнеса. Это позволяет легко добавлять новые приложения и сервисы без серьезных изменений инфраструктуры» [21].	Кривая обучения: внедрение iPaaS требует понимания его особенностей и возможностей. Командам необходимо научиться создавать коннекторы, обрабатывать преобразования данных и управлять рабочими процессами. Кастомизация против стандартизации: Хотя iPaaS предлагает гибкость, найти правильный баланс между настройкой и стандартизацией может быть непросто. Собственные интеграции могут не охватывать все уникальные бизнес-требования. Роботизированная автоматизация процессов (RPA): RPA может улучшить iPaaS за счет автоматизации повторяющихся задач. Однако поддержка сценариев RPA и адаптация к изменениям пользовательского интерфейса могут оказаться сложной задачей.

Продолжение таблицы 1

Метод	Преимущества	Недостатки
HIP	Масштабируемость. Традиционные локальные приложения могут быть сложными для масштабирования. Гибридная платформа интеграции позволяет сосуществовать различным системам, обеспечивая гибкость и отказоустойчивость. Оно поддерживает рост и увеличивает доходы за счет эффективного использования новых возможностей для бизнеса. Безопасность и соответствие требованиям. Использование облачных провайдеров часто приводит к повышению безопасности благодаря передовым практикам и большой емкости хранилища. Предприятия получают выгоду от простого доступа к данным, снижения затрат и снижения риска сбоя системы при хранении данных извне1. Скорость. Платформы гибридной интеграции ускоряют процессы передач данных за счет оптимизации сетей, уменьшения задержек и оптимизации обмена данными. Экономическая эффективность: гибридная интеграция сокращает время разработки, необходимое для совместно работы приложений. Такие функции, ка сокращение кода и возможности перетаскивания, экономят время, позволяя разработчикам сосредоточитьс на инициативах по цифровой трансформации.	Ограничения однородного варианта. Выбор однородной гибридной платформы интеграции может привести к ограничениям, которые не отвечают всем потребностям предприятия. Кроме того, некоторые компоненты могут остаться неиспользованными. Рекомендации по использованию лучших в своем классе решений. Хотя лучшие в своем классе решения обеспечивают гибкость, для их навигации может потребоваться больше усилий из-за различных интерфейсов. Командам интеграции необходимо оценить, перевешивают ли преимущества сложность.

Продолжение таблицы 1

Метод	Преимущества	Недостатки
API баз данных	Абстрагирует базовую сложность базы данных, позволяя разработчикам сосредоточиться на логике приложения, не беспокоясь о деталях реализации базы данных. Предоставляет стандартизированный способ взаимодействия с базой данных, упрощая переключение между различными базами данных или интеграцию с другими системами. Может обеспечить дополнительный уровень безопасности, контролируя доступ к базе данных и гарантируя, что только авторизованные приложения могут взаимодействовать с данными. Может помочь улучшить масштабируемость приложения, предоставляя гибкий и эффективный способ взаимодействия с базой данных. Можно повторно использовать в нескольких приложениях, что сокращает время разработки и усилия, необходимы для взаимодействия с базой данных.	Может добавить дополнительный уровень сложности в систему, что може быть сложно в управлении и обслуживании. Может привести к накладным расходам на производительность, поскольку уровень API может добавить задержку и снизить общую производительность системы. Может привести к привязке к поставщику, что затрудняет переключение на другого поставщика или технологию базы данных. Может иметь крутую кривую обучения, требующую от разработчиков вкладывать время и усилия в изучение API и его нюансов. Приложение может стать зависимым от API базы данных, что затрудняет поддержку или обновление приложения если API изменится или станет недоступным.

На основе результатов анализа табличных данных выбран комплексный поход к интеграции веб-приложения с КИС предприятия, предусматривающий возможность применения различных методов интеграции в зависимости от конкретной задачи.

Выводы по главе 2

В результате проделанной работы по главе 2 были сделаны следующие выводы:

- рассмотрены следующие методы интеграции веб-приложений с КИС предприятия: метод системной интеграции, метод интеграции на основе СОА, метод облачной интеграции, метод интеграции на основе API баз данных;
- системная интеграция надежно объединяет функциональность

разнородных приложений и, как показало исследование, приводит к разработке новых стратегических бизнес-решений для предприятий;

- ESB – это централизованная платформа для обмена данными между системами. Обеспечивает более гибкую архитектуру;
- в основе концепции ESM лежит сервисный подход, по которому все подразделения компании рассматриваются как поставщики услуг для внутренних или внешних клиентов;
- большинство корпоративных платформ iPaaS имеют надежный набор инструментов, который практически не требует программирования;
- подключение API к базам данных означает, что приложения могут извлекать, обрабатывать и хранить данные в режиме реального времени. Это подключение гарантирует, что пользователи получают самую актуальную информацию, а приложения могут предоставлять динамический контент на основе взаимодействий и предпочтений пользователя.

На основе результатов анализа данных таблицы 1 выбран комплексный поход к интеграции веб-приложения с КИС предприятия, предусматривающий возможность применения различных методов интеграции в зависимости от конкретной задачи.

Глава 3 Разработка моделей интеграции веб-приложений с корпоративной информационной системой предприятия

Комплексный подход к интеграции веб-приложения с ERP-системой включает в себя несколько этапов и аспектов.

Во-первых, необходимо определить цели и задачи интеграции, а также выбрать подходящий метод интеграции, который зависит от целей, задач и особенностей компании.

Для реализации комплексного подход разработаны модели моделей интеграции веб-приложений с корпоративной информационной системой предприятия, представленные ниже.

Модели построены с помощью онлайн-сервиса Visual Paradigm и CASE-средства Rational Rose [32], [38].

Диаграммы последовательностей UML – это мощный визуальный инструмент, используемый в бизнес-анализе для представления взаимодействий между различными компонентами или субъектами в системе.

Они предоставляют пошаговое описание того, как эти сущности взаимодействуют и сотрудничают, помогая бизнес-аналитикам лучше понимать сложные процессы [17].

По своей сути диаграммы последовательностей демонстрируют хронологический порядок, в котором действия происходят в системе. Каждый компонент или субъект представлен «линией жизни», а стрелки указывают сообщения, которыми они обмениваются.

Эти сообщения могут быть простыми вызовами методов, сигналами или асинхронными сообщениями.

Диаграмма последовательности сценария интеграции веб-приложения с КИС предприятия для REST API показана на рисунке 21 [5].

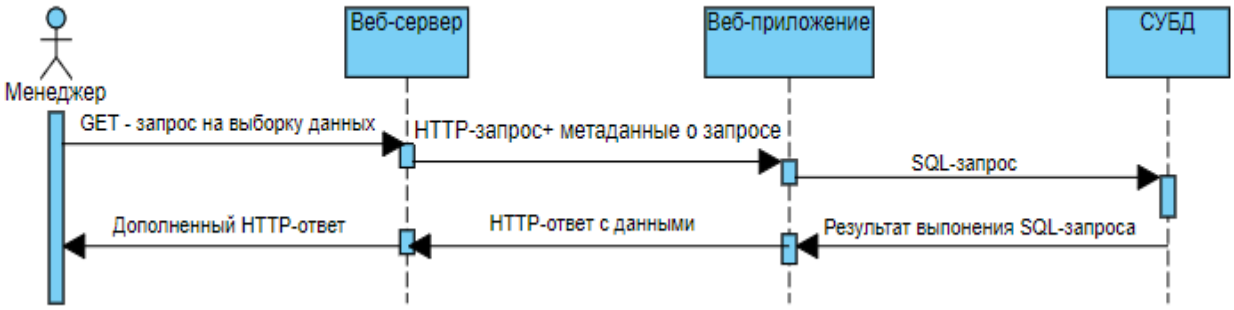


Рисунок 21 – Диаграмма последовательности сценария интеграции веб-приложения для REST API

На рисунке 22 представлена модель интеграции веб-приложения с КИС предприятия на основе iPaaS.

Модель построена в виде диаграммы компонентов UML [37].

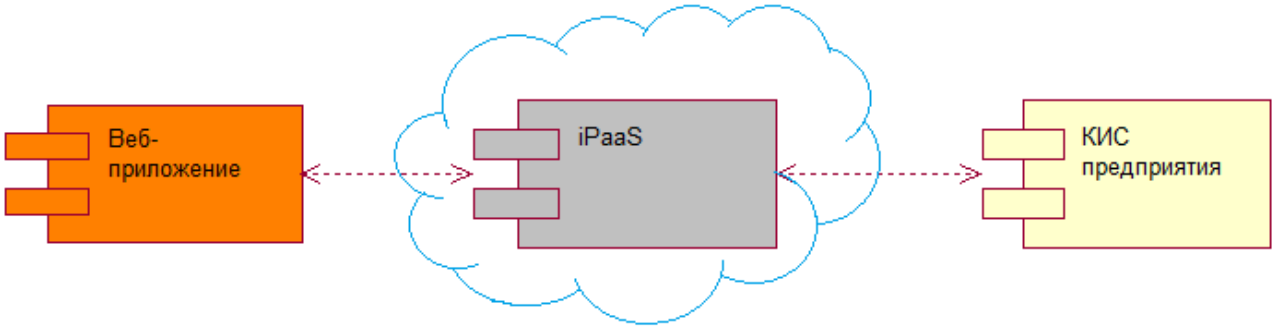


Рисунок 22 – Модель интеграция веб-приложения с КИС предприятия на основе iPaaS

Здесь используется метод облачной интеграции.

«Облачная интеграция объединяет несколько гибридных или облачных сред в единую ИТ-инфраструктуру для унификации процессов, систем и приложений. Она поддерживает построение и выполнение различных потоков интеграции, интеграции процессов и обмена сообщениями» [42].

На рисунке 23 представлена блочная модель интеграции веб-приложения (BI) с ERP-системой предприятия.

Модель интеграции состоит из следующих компонентов:

– ERP-система;
– ETL;
– хранилище данных;
– витрины данных;
– веб-приложение (BI).

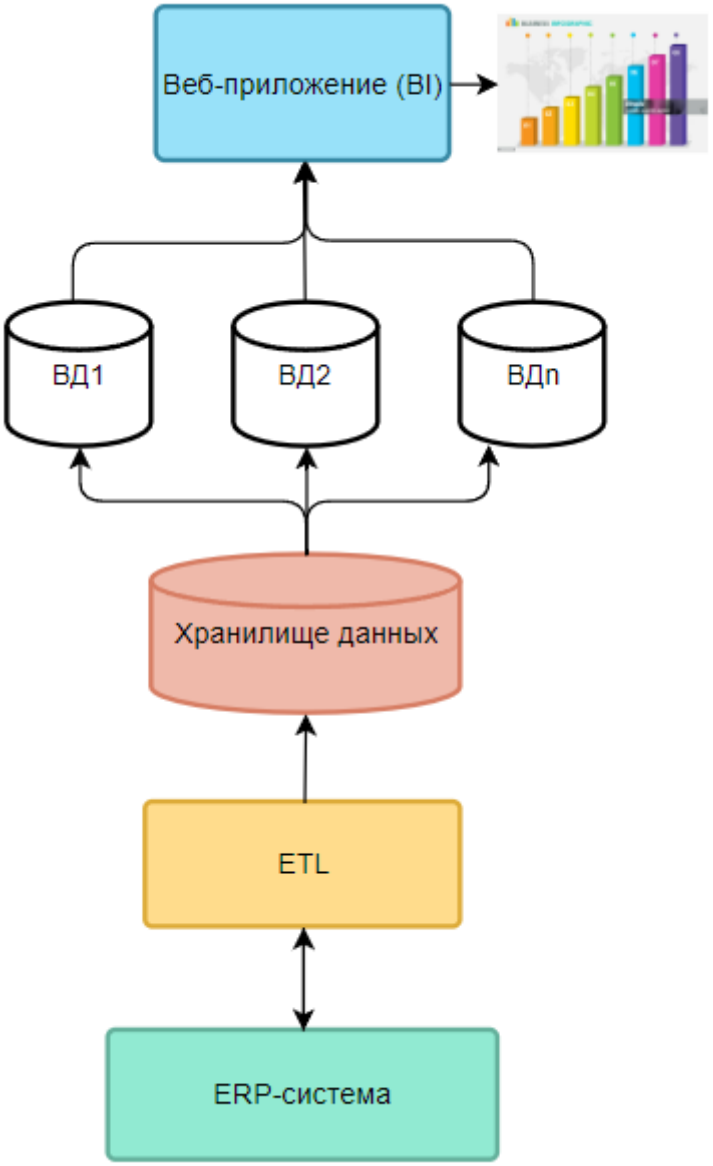


Рисунок 23 – Модель интеграции веб-приложения (BI) с ERP-системой предприятия

Инструменты ETL (выгрузка, трансформация и загрузка) снабжают хранилище данных подходящими форматированными данными. Для этой модели используется специальное хранилище данных, специально предназначенное для ERP.

Из хранилища данных извлекаются несколько витрин данных, чтобы обеспечить хороший интерактивный ответ на запросы по подразделениям организации.

Веб-приложение (BI) производит аналитическую обработку данных из витрин данных и визуализацию результатов анализа для поддержки принятия управляющих решений.

Рассмотрим процесс построения диаграммы вариантов использования UML.

Вариант использования (обозначение: овал/эллипс) представляет системную транзакцию с внешним системным пользователем, называемым актором. Варианты использования иногда считаются функциональными требованиями высокого уровня.

Диаграмма вариантов использования показывает коммуникации между системными транзакциями (вариантами использования) и внешними пользователями (акторами) в контексте границы системы (обозначение: прямоугольник).

Акторы могут представлять программное обеспечение (персоны, организации, объекты), программные системы или аппаратные системы. Определение отношений между субъектом системы и акторами системы является эффективным неформальным способом определения области действия системы [37].

Цель диаграмм вариантов использования – предоставить высокоуровневое представление субъектной системы и передать системные требования верхнего уровня в нетехнических терминах для всех заинтересованных сторон, включая клиентов и менеджеров проектов, а также архитекторов и инженеров. Для указания масштабируемой и имитируемой

модели архитектуры системы необходимы дополнительные более строгие диаграммы UML [37].

Акторами интеграции веб-приложения по методу API баз данных являются: API, Клиент, Веб-сервер и СУБД.

Описание вариантов использования интеграции веб-приложения по методу API баз данных в таблицах 2-6.

Таблица 2 – Описание прецедента: Запрос Клиента к API

«Элемент диаграммы	Описание
Прецедент	Запрос Клиента к API
ID	1
Краткое описание	Запрос клиента веб-приложения к API
Главный актор	Клиент
Второстепенный актор	API
Предусловие	Нет
Основной поток	Клиент обращается с запросом на подключение к API
Постусловие	Нет
Альтернативные потоки	Нет» [9]

Таблица 3 – Описание прецедента: Ответ API Клиенту

«Элемент диаграммы	Описание
Прецедент	Ответ API Клиенту
ID	2
Краткое описание	Ответ API Клиенту
Главный актор	API
Второстепенный актор	Клиент
Предусловие:	Запрос Клиента к API
Основной поток	API отвечает Клиенту на запрос
Постусловие	Нет
Альтернативные потоки	Нет» [9]

Таблица 4 – Описание прецедента: Запрос Веб-севера к API

«Элемент диаграммы	Описание
Прецедент	Запрос Веб-сервера к API
ID	3» [9]

Продолжение таблицы 4

«Элемент диаграммы	Описание
Краткое описание	Запрос Веб-сервера к API
Главный актор	Веб-сервер
Второстепенный актор	API
Предусловие	Нет
Основной поток	Веб-сервер обращается с запросом на подключение к API
Постусловие	Нет
Альтернативные потоки	Нет» [9]

Таблица 5 – Описание прецедента: Ответ API Веб-серверу

«Элемент диаграммы	Описание
Прецедент	Ответ API Веб-серверу
ID	4
Краткое описание	Ответ API Веб-серверу
Главный актор	API
Второстепенный актор	Веб-сервер
Предусловие:	Запрос Веб-сервера к API
Основной поток	API отвечает Веб-серверу на запрос
Постусловие	Нет
Альтернативные потоки	Нет» [9]

Таблица 6 – Описание прецедента: Запрос СУБД к API

«Элемент диаграммы	Описание
Прецедент	Запрос СУБД к API
ID	5
Краткое описание	Запрос СУБД к API
Главный актор	СУБД
Второстепенный актор	API
Предусловие	Нет
Основной поток	СУБД обращается с запросом на подключение к API
Постусловие	Нет
Альтернативные потоки	Нет» [9]

Таблица 7 – Описание прецедента: Ответ API СУБД

«Элемент диаграммы	Описание
Прецедент	Ответ API СУБД
ID	6
Краткое описание	Ответ API СУБД
Главный актор	API
Второстепенный актор	СУБД
Предусловие:	Запрос СУБД к API
Основной поток	API отвечает СУБД на запрос
Постусловие	Нет
Альтернативные потоки	Нет» [9]

Разработанная диаграмма вариантов использования интеграции веб-приложения с КИС предприятия по методу API баз данных представлена на рисунке 24.

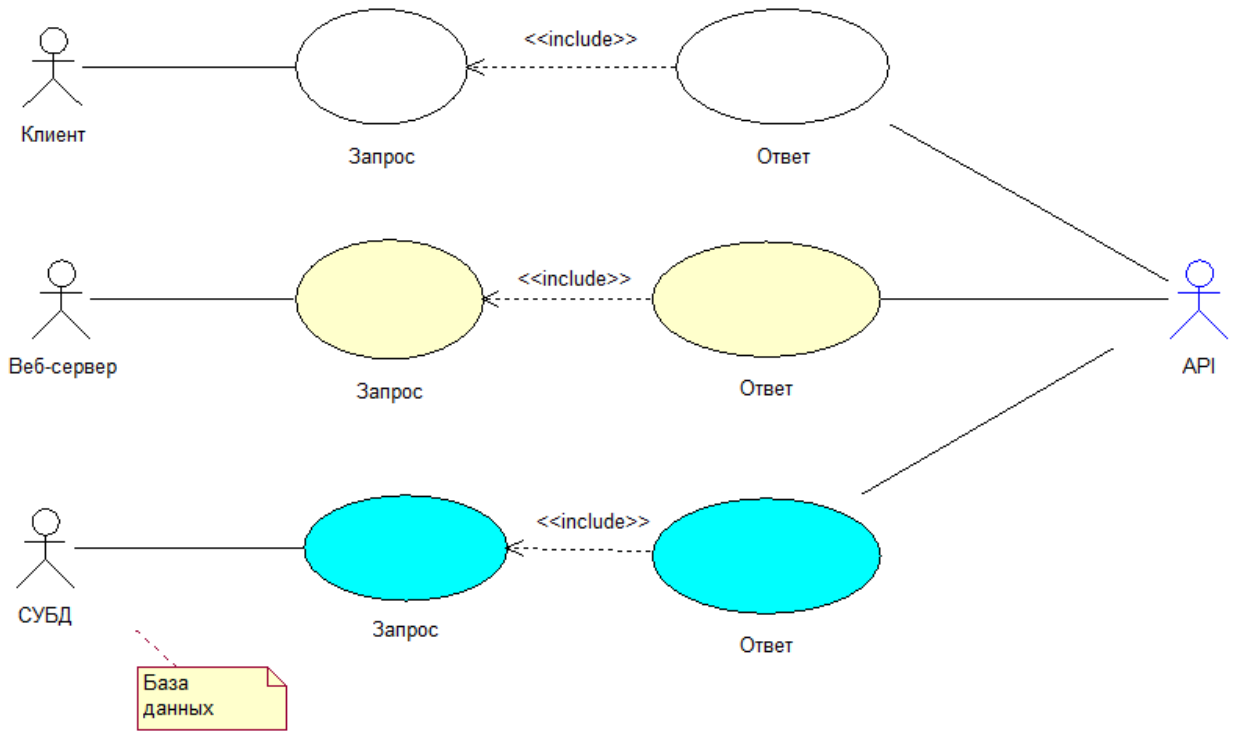


Рисунок 24 – Диаграмма вариантов использования интеграции веб-приложения с КИС предприятия по методу API баз данных

Разработанные модели позволяют реализовать комплексный подход к интеграции веб-приложения с КИС предприятия в зависимости от задачи, которые решает интегрируемое веб-приложение.

Выбор метода интеграции зависит от масштаба компании, количества ее бизнес-процессов, а также изменений, которых хочет достичь организация.

В одних случаях автоматизация распространяется лишь на некоторые отделы, в других – на все предприятие.

К тому же переходить к новому формату можно постепенно.

Выводы по главе 3:

В результате проделанной работы по главе 3 были сделаны следующие выводы:

- комплексный подход к интеграции веб-приложения с ERP-системой включает в себя несколько этапов и аспектов;
- выбор метода интеграции зависит от масштаба компании, количества ее бизнес-процессов, а также изменений, которых хочет достичь организация.

Разработанные модели позволяют реализовать комплексный подход к интеграции веб-приложения с КИС предприятия в зависимости от задачи, которые решает интегрируемое веб-приложение.

Глава 4 Апробация проектных решений и оценка их эффективности

4.1 Апробация проектных решений

Для апробации проектных решений использован метод интеграции веб-приложений с КИС предприятия на основе API баз данных.

Диаграмма развертывания решения показан на рисунке 25.

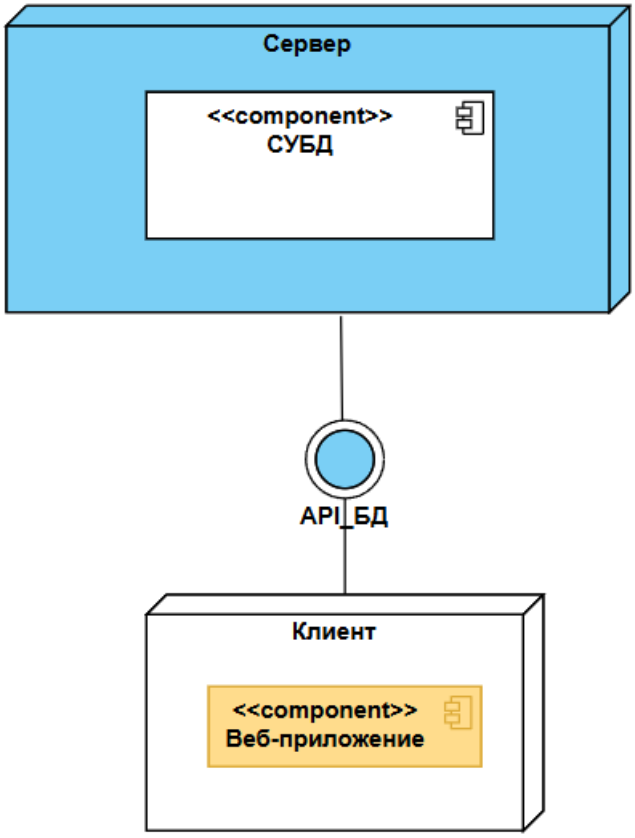


Рисунок 25 – Диаграмма развертывания интеграции веб-приложений с КИС предприятия на основе API баз данных

В качестве языка разработки выбран язык Python [31].

В качестве среды разработки выбрана среда Jupyter Notebook [30].

Jupyter Notebook – это популярная веб-интерактивная вычислительная среда, которая предлагает ряд преимуществ для науки о данных, научных вычислений и образования.

Вот некоторые из преимуществ использования Jupyter Notebook:

- интерактивные вычисления;
- выполнение ячеек кода: Jupyter Notebook позволяет вам выполнять ячейки кода одну за другой, что упрощает эксперименты и визуализацию результатов;
- немедленная обратная связь: вы получаете немедленную обратную связь по вашему коду, что помогает в отладке и лучшем понимании кода;
- визуализация данных;
- поддержка мультимедийных данных: Jupyter Notebook поддерживает мультимедийные данные, включая изображения, видео и интерактивные визуализации, что упрощает передачу идей и результатов;
- интеграция Matplotlib и Seaborn: Jupyter Notebook поставляется со встроенной поддержкой популярных библиотек визуализации данных, таких как Matplotlib и Seaborn;
- совместно используемые блокноты: Jupyter Notebook можно легко совместно использовать с другими, что делает его отличным инструментом для совместной работы и обмена знаниями;
- контроль версий: Jupyter Notebook поддерживает контроль версий, что позволяет отслеживать изменения и сотрудничать с другими;
- интерактивное обучение: Jupyter Notebook предоставляет интерактивную среду обучения, что упрощает изучение и преподавание концепций программирования;
- учебники и руководства: Jupyter Notebook широко используется в учебных пособиях и руководствах, что упрощает изучение и обучение у других;
- гибкость и настройка;
- поддержка нескольких языков: Jupyter Notebook поддерживает несколько языков программирования, включая Python, R, Julia и

MATLAB;

- настраиваемый интерфейс: интерфейс Jupyter Notebook можно настроить в соответствии с вашими потребностями, с поддержкой тем, расширений и плагинов;
- воспроизводимость: Jupyter Notebook позволяет легко воспроизводить результаты, что необходимо в научных вычислениях и науке о данных;
- доступность: Jupyter Notebook доступен из любой точки мира, что упрощает удаленную работу над проектами.

Рассмотрим модели интеграции веб-приложений на Python с популярными СУБД.

На рисунках 26,27 представлены программный код и результат подключения веб-приложения к СУБД SQLite, соответственно [16].

```
import sqlite3 as sl
#sqlite3.api level
#sqlite3.sqlite_version
```

```
db1=sl.connect('3_5.db')
```

```
#from google.colab import drive
#drive.mount('/content/drive')
```

```
crs=db1.cursor()
crs.execute("""CREATE TABLE IF NOT EXISTS doctor(doctor_id serial NOT NULL PRIMARY KEY,
    surname varchar(50) NOT NULL,
    firstname varchar(50) NOT NULL,
    patronymic varchar(20) NULL,
    dep_id int NOT NULL,
    spec_id int NOT NULL)""")
crs.execute("DELETE FROM doctor" )
sql_insert="""INSERT INTO doctor VALUES (1,'Федоров', 'Илья', 'Васильевич', 1, 1),(2, 'Мозгунова',

try:
    crs.execute(sql_insert)
except sl.DatabaseError as err:
    print('Error: ', err)
else:
    db1.commit()
    print('Запрос успешно выполнен')

crs.close()
```

```
Запрос успешно выполнен
```

```
crs=db1.cursor()

crs.executescript("""CREATE TABLE IF NOT EXISTS deps (dep_id serial NOT NULL PRIMARY KEY, name var
CREATE TABLE IF NOT EXISTS spec (spec_id serial NOT NULL PRIMARY KEY, name varchar(100) NOT NULL);
DELETE FROM deps;
DELETE FROM spec;
INSERT INTO deps VALUES (1, 'Кардиологическое отделение'), (2, 'Пульмонологическое отделение'), (3
INSERT INTO spec  VALUES (1, 'Кардиолог'), (2, 'Пульмонолог'), (3, 'Гастроэнтеролог'),(4, 'Врач фу
db1.commit()
```

Рисунок 26 – Код подключения веб-приложения к СУБД SQLite

```
crs=db1.cursor()
print('\nПросмотр всех записей таблицы doctor')
for r in crs.execute('SELECT * FROM doctor ORDER BY doctor_id '):
  print(r)
```

```
Просмотр всех записей таблицы doctor
(1, 'Федоров', 'Илья', 'Васильевич', 1, 1)
(2, 'Мозгунова', 'Светлана', 'Яковлевна', 2, 3)
(3, 'Крапивова', 'Екатерина', 'Игоревна', 3, 2)
```

```
crs=db1.cursor()
st=[(4,'Смирнов', 'Алексей', 'Михайлович', 1, 2),(5,'Арутян', 'Карен', 'Альбертович', 2, 3),

try:
    crs.executemany('INSERT INTO doctor VALUES (?,?,?,?,?,?)',st)
except sqlite3.DatabaseError as err:
    print('Error: ', err)
    print('Последний идентификатор вставленной строки: ', crs.lastrowid)
else:
    db1.commit()
    print('Запрос успешно выполнен')
crs.close()
```

```
Запрос успешно выполнен
```

```
r=[]
crs=db1.cursor()
c=crs.execute('SELECT * FROM deps ORDER BY name')
for i in c:
  r.append(i)
r
```

```
[(3, 'Гастроэнтерологическое отделение'),
 (1, 'Кардиологическое отделение'),
 (2, 'Пульмонологическое отделение')]
```

```
crs.close()
```

Рисунок 27 – Результат подключения веб-приложения к СУБД SQLite

Рассмотрим использование библиотеки pandas для управления данными.

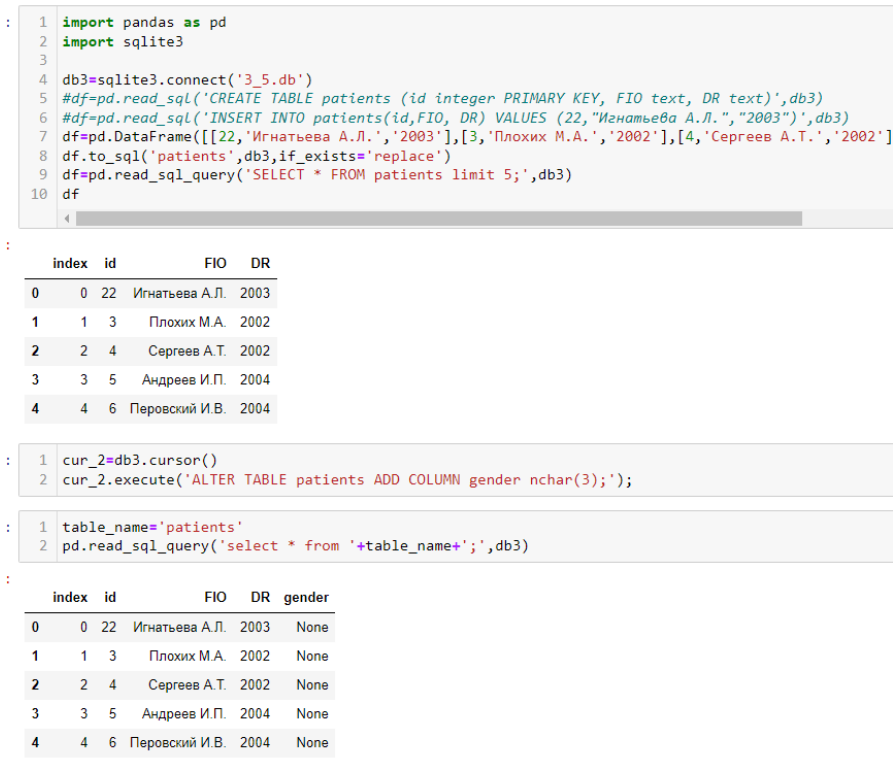
Pandas – популярная библиотека с открытым исходным кодом на Python для обработки и анализа данных. Она предоставляет структуры данных и функции для эффективной обработки структурированных данных, включая табличные данные, такие как электронные таблицы и таблицы SQL.

Библиотека pandas является частью дистрибутива Anaconda и может быть установлен вместе с Anaconda или Miniconda.

Основные характеристики Pandas:

– структуры данных: Pandas предлагает две основные структуры данных: Series (одномерный маркированный массив) и DataFrame (двумерная маркированная структура данных со столбцами потенциально разных типов);
– «операции с данными: Pandas предоставляет различные методы для обработки данных, такие как фильтрация, сортировка, группировка и слияние данных;
– анализ данных: Pandas предлагает функции для анализа данных, включая статистические функции, визуализацию данных и очистку данных;
– ввод/вывод данных: Pandas поддерживает чтение и запись данных из различных форматов, включая базы данных CSV, Excel, JSON и SQL» [27].

На рисунках 28,29 представлены программный код и результат примера применения библиотеки pandas, соответственно.



```
import pandas as pd
import sqlite3

db3=sqlite3.connect('3_5.db')
#df=pd.read_sql('CREATE TABLE patients (id integer PRIMARY KEY, FIO text, DR text)',db3)
#df=pd.read_sql('INSERT INTO patients(id,FIO, DR) VALUES (22,"Игнатьева А.Л.","2003")',db3)
df=pd.DataFrame([[22,'Игнатьева А.Л.','2003'],[3,'Плохих М.А.','2002'],[4,'Сергеев А.Т.','2002']
df.to_sql('patients',db3,if_exists='replace')
df=pd.read_sql_query('SELECT * FROM patients limit 5;',db3)
df
```

	index	id	FIO	DR
0	0	22	Игнатьева А.Л.	2003
1	1	3	Плохих М.А.	2002
2	2	4	Сергеев А.Т.	2002
3	3	5	Андреев И.П.	2004
4	4	6	Перовский И.В.	2004

```
cur_2=db3.cursor()
cur_2.execute('ALTER TABLE patients ADD COLUMN gender nchar(3);');
```

```
table_name='patients'
pd.read_sql_query('select * from '+table_name+';',db3)
```

	index	id	FIO	DR	gender
0	0	22	Игнатьева А.Л.	2003	None
1	1	3	Плохих М.А.	2002	None
2	2	4	Сергеев А.Т.	2002	None
3	3	5	Андреев И.П.	2004	None
4	4	6	Перовский И.В.	2004	None

Рисунок 28 – Код применения библиотеки pandas

```
cur_2=db3.cursor()
cur_2.execute('ALTER TABLE patients ADD COLUMN gender nchar(3);');
```

```
table_name='patients'
pd.read_sql_query('select * from '+table_name+';',db3)
```

	index	id	FIO	DR	gender
0	0	22	Игнатьева А.Л.	2003	None
1	1	3	Плохих М.А.	2002	None
2	2	4	Сергеев А.Т.	2002	None
3	3	5	Андреев И.П.	2004	None
4	4	6	Перовский И.В.	2004	None

```
df=pd.read_sql('SELECT * FROM  patients',db3)
df['gender']=None
df.to_sql('patients',db3,if_exists='replace')
```

```
5
```

```
from datetime import datetime
db2=sqlite3.connect('3_5.db')
df=pd.DataFrame([[1,'Sergeev BB','Пульмонологическое отделение',datetime(2023,11,29,12,20)
                  columns=['id','FIO_prep','subject','Lec_beg','Lec_end'])
df.to_sql('Report1',db2,if_exists='replace')
pd.read_sql_query('select * from Report1;',db2)
table_name='Report1'
#crs.execute('alter table Report1 add column room integer')
pd.read_sql_query('select * from '+table_name+';',db2)


```

	index	id	FIO_prep	subject	Lec_beg	Lec_end
0	0	1	Sergeev BB	Пульмонологическое отделение	2023-11-29 12:20:00	2023-12-29 13:55:00

Рисунок 29 – Результат применения библиотеки pandas

Рассмотрим модель подключения к СУБД MySQL с помощью модуля Getpass [14].

Getpass – это модуль Python, который позволяет безопасно запрашивать у пользователя пароль или другую конфиденциальную информацию без повтора ввода в консоль. Он является частью стандартной библиотеки Python.

Основные характеристики Getpass:

- безопасный ввод: Getpass предоставляет способ ввода конфиденциальной информации, такой как пароли, без отображения ввода на экране;
- независимость от платформы: Getpass работает на различных

платформах, включая Windows, macOS и Linux;

– простой API: Getpass имеет простой и удобный в использовании API, что упрощает интеграцию в ваши приложения Python.

На рисунке 30 представлен код подключения к СУБД MySQL с помощью модуля Getpass.

```
#Доступ к базам данных MySQL
```

```
pip install mysql-connector-python
```

```
Requirement already satisfied: mysql-connector-python in d:\anaconda3\lib\site-packages (8.3.0)
Note: you may need to restart the kernel to use updated packages.
```

```
from getpass import getpass
from mysql.connector import connect, Error

try:
    with connect(
        host="localhost",
        user=input("Имя пользователя: "),
        password=getpass("Пароль: "),
    ) as connection:
        create_db_query = "CREATE DATABASE IF NOT EXISTS test_1"
        with connection.cursor() as cursor:
            cursor.execute(create_db_query)
except Error as e:
    print(e)
```

```
Имя пользователя: root
Пароль: ········
```

```
#просмотр всех таблиц, хранящихся в базе данных
try:
    with connect(
        host="localhost",
        user=input("Введите имя пользователя: "),
        password=getpass("Введите пароль: "),
    ) as connection:
        print(connection)
        show_db_query = "SHOW DATABASES"
        with connection.cursor() as cursor:
            cursor.execute(show_db_query)
            for db in cursor:
                print(db)
except Error as e:
    print(e)
```

```
Введите имя пользователя: root
Введите пароль: ········
<mysql.connector.connection_cext.CMySQLConnection object at 0x00000257FEAD5310>
('information_schema',)
('mysql',)
('performance_schema',)
```

Рисунок 30 – Код и результат подключения к СУБД MySQL с помощью модуля Getpass

На рисунке 31 представлен код управления данными в СУБД MySQL.

```
#подключение к существующей БД
import requests
import mysql.connector

cnx = mysql.connector.connect(user ='root', password= 'mysql', host = '127.0.0.1',port='3306', database='test_1')
print(cnx)
cursor = cnx.cursor()
sql_create = """CREATE TABLE IF NOT EXISTS students (id integer PRIMARY KEY, FIO text, DR text)"""
cursor.execute(sql_create)
sql_show = "DESCRIBE students"
with cnx.cursor() as cursor:
    cursor.execute(sql_show)
    res = cursor.fetchall()
    for row in res:
        print(row)

sql_del="DELETE FROM students"
with cnx.cursor() as cursor:
    cursor.execute(sql_del)
    cnx.commit()

sql_insert="INSERT INTO students VALUES (1,'Ivanov I','2004'),(2,'Smirnov S','2002')"
with cnx.cursor() as cursor:
    cursor.execute(sql_insert)
    cnx.commit()

sql_select = "SELECT * FROM students"
with cnx.cursor() as cursor:
    cursor.execute(sql_select)
    result = cursor.fetchall()
    for row in result:
        print(row)

```

```
<mysql.connector.connection_cext.CMySQLConnection object at 0x00000257FF6282D0>
('id', 'int', 'NO', 'PRI', None, '')
('FIO', 'text', 'YES', '', None, '')
('DR', 'text', 'YES', '', None, '')
(1, 'Ivanov I', '2004')
(2, 'Smirnov S', '2002')
```

Рисунок 31 – Код и результат управления данными в СУБД MySQL

Рассмотрим модель подключения веб-приложения к СУБД PostgreSQL [15].

Psycopg2 – это адаптер базы данных PostgreSQL для языка программирования Python. Это расширение Python, которое позволяет подключаться к базам данных PostgreSQL и выполнять SQL-запросы из скриптов или приложений Python.

На рисунке 32 представлены код и результат подключения к СУБД PostgreSQL.

#Работа с СУБД PostgreSQL

```
pip install psycopg2-binary
```

```
^C
Note: you may need to restart the kernel to use updated packages.
```

```
import psycopg2
psycopg2.__libpq_version__
```

```
160000
```

```
# Подключение к серверу БД, просмотр информации о сервере PostgreSQL
import psycopg2
from psycopg2 import Error

try:
    connection = psycopg2.connect(user="postgres",
                                  # пароль, который указали при установке PostgreSQL
                                  password="123",
                                  host="127.0.0.1",
                                  port="5432",
                                  database="OrdersDB") #as cn: with cn.cursor() as cur: ...

    # Курсор для выполнения операций с БД
    cur = connection.cursor()
    # Распечатать сведения о PostgreSQL
    print("Информация о сервере PostgreSQL")
    print(connection.get_dsn_parameters(), "\n")
    # Выполнение SQL-запроса
    cur.execute("SELECT version();")
    # Получить результат
    record = cur.fetchone()
    print("Вы подключены к - ", record, "\n")

except (Exception, Error) as error:
    print("Ошибка при работе с PostgreSQL", error)
finally:
    if connection:
        cur.close()
        connection.close()
        print("Соединение с PostgreSQL закрыто")
```

```
Информация о сервере PostgreSQL
{'user': 'postgres', 'channel_binding': 'prefer', 'dbname': 'OrdersDB', 'host': '127.0.0.1', 'port': '5432', 'options': '', 'ss
lmode': 'prefer', 'sslcompression': '0', 'sslcertmode': 'allow', 'sslsni': '1', 'ssl_min_protocol_version': 'TLSv1.2', 'gssencm
ode': 'disable', 'krbsrvname': 'postgres', 'gssdelegation': '0', 'target_session_attrs': 'any', 'load_balance_hosts': 'disabl
e'}

Вы подключены к -  ('PostgreSQL 10.22, compiled by Visual C++ build 1925, 64-bit',)
```

Рисунок 32 – Код и результат подключения к СУБД PostgreSQL

На рисунке 33 представлены код и результат управления данными в СУБД PostgreSQL.

```python
# Подключение к существующей БД test_db
import psycopg2
from psycopg2 import Error
from psycopg2.extensions import ISOLATION_LEVEL_AUTOCOMMIT
try:
    # Подключение к существующей базе данных
    connection = psycopg2.connect(user="postgres",
                                  # пароль, который указали при установке PostgreSQL
                                  password="123",
                                  host="127.0.0.1",
                                  port="5432")
    connection.set_isolation_level(ISOLATION_LEVEL_AUTOCOMMIT)
    # Курсор для выполнения операций с базой данных
    cursor = connection.cursor()
    sql_create_database = 'create database ik_db'
    cursor.execute(sql_create_database)
except (Exception, Error) as error:
    print("Ошибка при работе с PostgreSQL", error)
finally:
    if connection:
        cursor.close()
        connection.close()
        print("Соединение с PostgreSQL закрыто")
```

```
Соединение с PostgreSQL закрыто
```

```python
import psycopg2
conn = psycopg2.connect(dbname='ik_db' ,
                        user='postgres',
                       password='123',
                      host='127.0.0.1' ,
                     port=5432)
cur = conn.cursor()
cur.execute("CREATE TABLE IF NOT EXISTS students (id integer PRIMARY KEY, FIO text, DR text)")
cur.execute("INSERT INTO students VALUES (1,'Иванов И.Р.','2004'),(2,'Смирнов В.С.','2002')",("postgres", "123"))
#cur.close()
conn.commit()
```

Рисунок 33 – Код и результат управления данными в СУБД PostgreSQL

Рассмотрим модель подключения веб-приложения к СУБД MS SQL Server.

Для этого используем библиотеку Pymssql.

Pymssql – это простой интерфейс базы данных для Python, который построен на основе FreeTDS и обеспечивает интерфейс Python DB-API (PEP-249) для MS SQL Server.

На рисунке 34 представлены код и результат управления данными в СУБД MS SQL Server.

```
!pip install pymssql
import pymssql
conn = pymssql.connect(
    server='DESKTOP-M3LJ3BU',
    user=None,
    password=None,
    database='INSURANCE_DB',
    as_dict=True
)
```

```
Requirement already satisfied: pymssql in d:\anaconda3\lib\site-packages (2.3.1)
```

```
#Подключение к MS SQL Server
cursor = conn.cursor()
cursor.execute("SELECT * FROM Agents")
```

```
records = cursor.fetchall()
for r in records:
    print(f"{r['AgentID']}\t{r['AgentFullName']}")
```

```
1       Anderson A
2       Brown B
3       Clark C
```

Рисунок 34 – Код и результат управления данными в СУБД MS SQL Server

Результаты апробация подтвердили работоспособность предлагаемых проектных решений по интеграции веб-приложений с КИС предприятия.

4.2 Оценка эффективности проектных решений

«Для оценки экономической эффективности предлагаемых проектных решений используем методику сравнения затрат на разработку модуля интеграции веб-приложения внешним программистом по договору аутсорсинга (базовый вариант) и программистом образовательной организации (проектный вариант), соответственно.

В калькуляцию себестоимости заказной разработки модуля интеграции веб-приложения включаются следующие статьи затрат:

– зарплата исполнителя проекта по трудовому договору ($ЗБ_1$);

– социальные страховые взносы ($ЗБ_2$);

– прочие прямые расходы ($ЗБ_3$);

– накладные расходы ($ЗБ_4$).

В заказной доработке задействован внешний программист» [11].

«Средняя стоимость часа работы программиста на языке Python по договору составляет 1500 руб» [13].

«Ориентировочное время разработки составляет 100 час.

Итого затраты базового варианта $С_{баз}$ составят (1):

$$С_{баз} = ЗБ_1 + ЗБ_2 + ЗБ_3 + ЗБ_4 \quad (1)$$

$С_{баз}$ = 1500*100 +0,271*1500*100+0+0 =190650 руб

В самостоятельной разработке модуля интеграции веб-приложения задействованы программист и аналитик предприятия.

В калькуляцию себестоимости самостоятельной разработки модуля интеграции веб-приложения включаются следующие статьи затрат:

– зарплата исполнителей проекта с учетом затраченного времени 100 час ($ЗП_1$);

– социальные страховые взносы ($ЗП_2$);

– прочие прямые расходы ($ЗП_3$);

– накладные расходы ($ЗП_4$).

Итого затраты проектного варианта $С_{пр}$ составят (2):

$$С_{пр} = ЗП_1 + ЗП_2 + ЗП_3 + ЗП_4 \quad (2)$$

$С_{пр}$ = (50000+40000) руб+0,3*(50000+40000)+0+0 = 117000 руб

Сформируем таблицу и график показателей экономической эффективности (таблица 8, рисунок 35)» [11].

Таблица 8 – Показатели эффективности проекта разработки модуля интеграции веб-приложения

«Затраты		Абсолютное изменение затрат	Коэффициент относительного снижения затрат	Индекс снижения затрат
Базовый вариант	Проектный вариант			
$С_{баз}$ (руб.)	$С_{пр}$ (руб.)	$\Delta С = С_{баз} - С_{пр}$ (руб.)	$К_С = \Delta С / С_{баз} \times 100\%$	$У_С = С_{баз} / С_{пр}$
190650	117000	73650	39	1,6» [11]

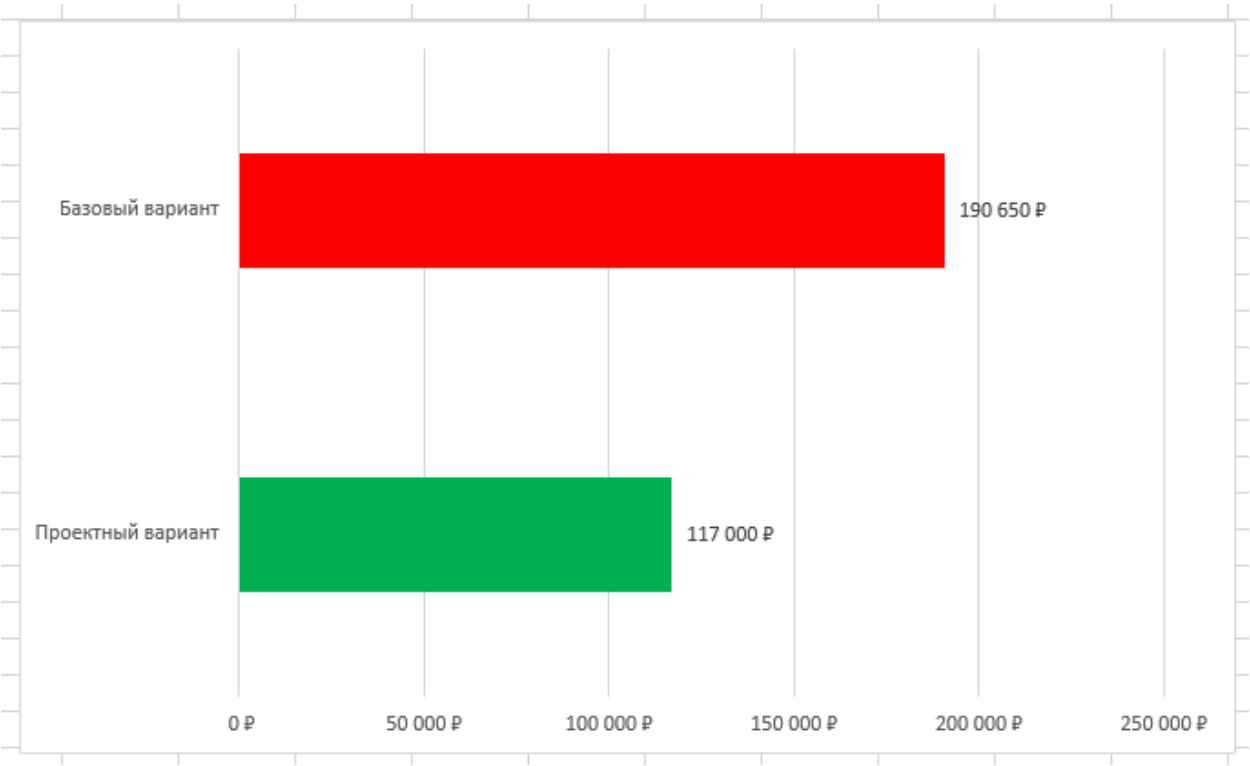


Рисунок 35 – Гистограмма сравнения затрат на разработку модуля интеграции веб-приложения

Таким образом, затраты при проектном варианте разработки модуля интеграции веб-приложения сократились в 1,6 раза.

«Срок окупаемости затрат на внедрение проектного решения ($Т_{ок}$) определяется по формуле (3):

$$T_{ок} = {K_п}/{\Delta С} \text{ (мес.)}, \quad (3)$$

где $К_П$ – затраты на реализацию проектных решений (проектирование и внедрение модуля интеграции веб-приложения).

Следовательно, срок окупаемости модуля интеграции веб-приложения равен:

$$Т_{ок} = 117000/73650 \approx 1{,}6 \text{ мес.}$$

Представленные расчеты подтвердили существенное снижение затрат на проектирование и эффективность проектного решения» [11].

«Для оценки эффективности управления модуля интеграции веб-приложения используем формулу расчета показателя функциональной эффективности управления (4):

$$К_{эу} = \frac{\sum_{i=1}^{n} Р_{yi}}{n} \quad (4)$$

где n - количество функций управления, реализуемых модулем интеграции веб-приложения;

$Р_{yi}$ - вероятность выработки модулем интеграции веб-приложения эффективного управляющего воздействия при реализации *i*-й функции управления» [4].

«Для управления данными используются следующие функции:

- подключение к СУБД;
- управления данными в СУБД.

Как показывает практика, эти функции успешно выполняются предлагаемым модулем интеграции веб-приложения.

В этом случае значение показателя функциональной эффективности управления модуля интеграции веб-приложения будет равно:

$$К_{эу} = 2/2 = 1$$

Таким образом, коэффициент эффективности управления предлагаемого модуля интеграции веб-приложения Кэу> 0,5, что свидетельствует о высокой функциональной эффективности процесса интеграции веб-приложения с КИС предприятия» [4].

Выводы по главе 4

Результаты проделанной работы позволили сделать следующие выводы:

- для апробации проектных решений использован метод интеграции веб-приложений на основе API баз данных;
- использование библиотек pandas и getpass обеспечивает эффективное подключение и управление данными при интеграции веб-приложения с КИС предприятия.

Расчеты подтвердили экономическую эффективность предлагаемых проектных решений и высокую функциональную эффективность управления предлагаемого модуля интеграции веб-приложения с КИС предприятия.

Заключение

Интеграция веб-приложений с корпоративной информационной системой (КИС) организации – это процесс, позволяющий различным приложениям обмениваться данными и функциональностью через API или коннекторы. Интеграция улучшает эффективность, безопасность и качество работы приложений, а также снижает затраты на их поддержку.

Интеграция веб-приложения с КИС предприятия означает связывание веб-приложения с ее ядром, которое образует ERP-система, которая обеспечивает управление основными бизнес-процессами и данными в разных отделах и функциональных областях. Интеграция веб-приложения с ERP позволяет обеспечить единый доступ к информации, улучшить производительность, снизить издержки и повысить удовлетворенность клиентов.

При эффективной интеграции веб-приложений повышается доступность данных КИС, их согласованность и общая эффективность работы. В конечном счете, это позволяет организациям максимально использовать преимущества и возможности своей КИС, обеспечивая им важное конкурентное преимущество в своей отрасли.

Магистерская диссертация посвящена актуальной проблеме разработки эффективной модели интеграции внешнего веб-приложения и КИС предприятия, в основу которой положены современные методы интеграции компонентов КИС, представляет актуальность и научно-практический интерес.

В процессе работы над магистерской диссертацией решены следующие задачи:

- дан анализ современного состояния исследований и разработок в области интеграции веб-приложений с КИС предприятия. Как показал анализ, планирование ресурсов предприятия (ERP) с поддержкой Интернета, основанной на интеграции веб-приложения

с КИС предприятия, имеет решающее значение для непрерывного развития производственного предприятия. Веб-приложения используется для поддержки принятия решений на различных организационных уровнях путем предоставления таких функций, как настройка, персонализация, поддержка совместной работы и уведомлений, которые помогают менеджерам принимать правильные решения с помощью Интернета. Проблематика интеграции веб-приложений с КИС предприятия широко рассмотрена в работах отечественных и зарубежных ученых и специалистов ведущих вендоров ПО. Одним из наиболее распространенных и современных способов является использование API-интерфейсов прикладного программирования, которые позволяют веб-приложениям взаимодействовать и передавать данные друг другу через HTTP-протокол. Выбор между SOAP и REST зависит от конкретных целей и характеристик веб-приложения и ERP-системы, а также от наличия готовых коннекторов или библиотек для интеграции. В любом случае, интеграция веб-приложения с ERP требует тщательного планирования, анализа, тестирования и поддержки, чтобы обеспечить эффективность и надежность обмена данными между системами. Анализ литературы и источников подтвердил интерес ученых и специалистов к проблеме формирования интеграции веб-приложений с КИС предприятия на основе современных ИТ;

– рассмотрены следующие методы интеграции веб-приложений с КИС предприятия: метод системной интеграции, метод интеграции на основе СОА, метод облачной интеграции, метод интеграции на основе API баз данных. Системная интеграция надежно объединяет функциональность разнородных приложений и, как показало исследование, приводит к разработке новых стратегических бизнес-решений для предприятий. ESB – это централизованная платформа

для обмена данными между системами. Обеспечивает более гибкую архитектуру. основе концепции ESM лежит сервисный подход, по которому все подразделения компании рассматриваются как поставщики услуг для внутренних или внешних клиентов. Большинство корпоративных платформ iPaaS имеют надежный набор инструментов, который практически не требует программирования. Подключение API к базам данных означает, что приложения могут извлекать, обрабатывать и хранить данные в режиме реального времени. Это подключение гарантирует, что пользователи получают самую актуальную информацию, а приложения могут предоставлять динамический контент на основе взаимодействий и предпочтений пользователя. На основе результатов анализа табличных данных выбран комплексный поход к интеграции веб-приложения с КИС предприятия, предусматривающий возможность применения различных методов интеграции в зависимости от конкретной задачи;

- разработаны модели, которые позволяют реализовать комплексный подход к интеграции веб-приложения с КИС предприятия в зависимости от задачи, которые решает интегрируемое веб-приложение;
- выполнена успешная апробация проектных решений. Расчеты подтвердили экономическую эффективность предлагаемых проектных решений и высокую функциональную эффективность управления предлагаемого модуля интеграции веб-приложения с КИС предприятия.

Гипотеза исследования подтверждена.

Список используемой литературы и используемых источников

1. Архитектура платформы 1С: Предприятие (веб-клиент) [Электронный ресурс]. URL: https://v8.1c.ru/platforma/web-klient/ (дата обращения: 20.10.2024).

2. Бобков О. Интеграция сайта с 1С: как подключить и связать веб-ресурс с ПО — инструкция подключения и синхронизации [Электронный ресурс]. URL: https://www.cleverence.ru/articles/elektronnaya-kommertsiya/integratsiya-sayta-s-1s-kak-podklyuchit-i-svyazat-veb-resurs-s-po-instruktsiya-podklyucheniya-i-sinkh/ (дата обращения: 20.09.2024).

3. В чем разница между SOAP и REST? [Электронный ресурс]. URL: https://aws.amazon.com/ru/compare/the-difference-between-soap-rest/ (дата обращения: 20.09.2024).

4. Вдовин В.М., Суркова Л.Е., Шурупов А.А. Предметно-ориентированные экономические информационные системы. М.: Дашков и К, 2016. 388 с.

5. Вичугова А. UML-диаграмма последовательности для REST API: практический пример [Электронный ресурс]. URL: https://babok-school.ru/blog/rest-api-example-web-server-and-web-app-interaction-on-uml-sequence/ (дата обращения: 20.09.2024).

6. Интеграция Web-приложений с системами товарного учета [Электронный ресурс]. URL: https://habr.com/ru/articles/120984/ (дата обращения: 20.11.2024).

7. Интеграция корпоративных приложений с помощью ESM-системы [Электронный ресурс]. URL: https://simpleone.ru/blog/integracziya-korporativnyh-prilozhenij-s-pomoshhyu-esm-sistemy/ (дата обращения: 20.09.2024).

8. Интернет запись на прием и обмен данными с сайтами [Электронный ресурс]. URL: https://solutions.1c.ru/catalog/clinic/internet_site (дата обращения: 21.09.2024).

9. Леоненков А. В. Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose : учебное пособие. М. : ИНТУИТ, Ай Пи Ар Медиа, 2020. 317 с. [Электронный ресурс]. URL: https://www.iprbookshop.ru/97554.html (дата обращения: 20.09.2024).

10. Подобрий А.Н. Web-интеграция для создания единого информационного пространства // Вестник УлГТУ. 2012. №4 (60). URL: https://cyberleninka.ru/article/n/web-integratsiya-dlya-sozdaniya-edinogo-informatsionnogo-prostranstva (дата обращения: 20.09.2024).

11. Поршкевич Н.Ю., Огнева А.Ю., Чинчукова Е.П. Оценка экономической эффективности применения информационных систем // Экономика и социум. 2016. №7 (26). URL: https://cyberleninka.ru/article/n/otsenka-ekonomicheskoy-effektivnosti-primeneniya-informatsionnyh-sistem (дата обращения: дата обращения: 20.09.2024).

12. Сайт Entaxy.ru [Электронный ресурс]. URL: https://entaxy.ru/product (дата обращения: 20.09.2024).

13. Сколько стоят услуги программистов? Цены студий и фрилансеров Источник: https://www.kadrof.ru/articles/46641 (дата обращения: дата обращения: 20.09.2024).

14. СУБД MySQL [Электронный ресурс]. URL: https://www.mysql.com/ (дата обращения: 20.09.2024).

15. СУБД Postgresql [Электронный ресурс]. URL: https://www.postgresql.org/ (дата обращения: 20.09.2024).

16. СУБД SQLite [Электронный ресурс]. URL: https://www.sqlite.org/index.html (дата обращения: 20.09.2024).

17. Сунгатуллина А. Т. Системный анализ и проектирование информационных систем на основе объектно-ориентированного подхода : учебно-методическое пособие по дисциплине «Методы и средства проектирования информационных систем». М. : Российский университет транспорта (МИИТ), 2020. 118 с. [Электронный ресурс]. URL:

https://www.iprbookshop.ru/115990.html (дата обращения: дата обращения: 20.09.2024).

18. Хасан Хан У. Что такое API базы данных? Почему и как они используются? [Электронный ресурс]. URL: https://www.astera.com/ru/type/blog/database-api/ (дата обращения: 20.09.2024).

19. Aziz O. et al. Research Trends in Enterprise Service Bus (ESB) Applications: A Systematic Mapping Study, IEEE Access, Vol. 8, 2020, P. 31180 – 31197.

20. Bennett T. The Pros and Cons of Point-to-Point Integration [Электронный ресурс]. URL: https://www.integrate.io/blog/pros-cons-of-point-to-point-integration (дата обращения: 20.09.2024).

21. Bigelow S.J. What is iPaaS? Guide to integration platform as a service [Электронный ресурс]. URL: https://www.techtarget.com/searchcloudcomputing/definition/iPaaS-integration-platform-as-a-service (дата обращения: 20.09.2024).

22. Brooks A. What Is a Hybrid Integration Platform (HIP)? [Электронный ресурс]. URL: https://www.adeptia.com/blog/what-is-a-hybrid-integration-platform (дата обращения: 20.09.2024).

23. Feng Zhou and Xiaoping Feng “Research and Design of the Web Service Application”, 6th International Conference on Electronic, Mechanical, Information and Management (EMIM 2016).

24. Johnny K.C. Nga, W.H. Ipa “Web-ERP: the new generation of enterprise resources planning”, Journal of Materials Processing Technology 6700 (2003) 1–5.

25. Johnson J. Database API: What Is It, How Does It Work & 6 Best APIs [Электронный ресурс]. URL: https://www.cdata.com/blog/what-is-database-api (дата обращения: 20.09.2024).

26. Pallares A. iPaaS: Examples, Benefits and Use Cases [Электронный ресурс]. URL: https://www.dckap.com/blog/ipaas-examples-benefits-and-use-cases/ (дата обращения: 20.09.2024).

27. Pandas documentation [Электронный ресурс]. URL:

https://pandas.pydata.org/pandas-docs/stable/index.html (дата обращения: 20.09.2024).

28. Picek R., Grcic M. ERP System Database Environment Configuration Using A Web Application, 2012 [Электронный ресурс]. URL: https://dblp.org/rec/conf/iti/PicekG12.html (дата обращения: 20.09.2024).

29. Prakash A. From API to Database: A Step-by-Step Guide on Efficient Data Integration [Электронный ресурс]. URL: https://airbyte.com/data-engineering-resources/api-to-database (дата обращения: 20.09.2024).

30. Project Jupyter [Электронный ресурс]. URL: https://jupyter.org/ (дата обращения: дата обращения: 20.09.2024).

31. Python: Работа с базой данных, часть 1/2: Используем DB-API https://habr.com/ru/articles/321510/ (дата обращения: 20.09.2024).

32. Rational Rose model [Электронный ресурс]. URL: https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=migration-rational-rose-model (дата обращения: 20.09.2024).

33. Sanghvi H. How to Integrate OpenAI into React Web App to Boost Your ERP System [Электронный ресурс]. URL: https://syndelltech.com/how-to-integrate-openai-into-erp-system/ (дата обращения: 20.09.2024).

34. Solanki M. et al. Integration of a web portal and an erp through web service based implementation and testing approach, International Journal of Research in Engineering and Technology (IJRET), Vol. 3, Iss. 11, P. 269-278.

35. Technology Blogs by SAP [Электронный ресурс]. URL: https://community.sap.com/t5/technology-blogs-by-sap/integration-architecture-for-enterprise-agility-powered-by-sap-btp/ba-p/13551112 (дата обращения: 20.09.2024).

36. Themistocleous M. et al. “ERP and application integration Business Process Management Journal”, Vol. 7, Iss. 3, P. 195 – 204.

37. Unified Modeling Language (UML) Diagrams [Электронный ресурс]. URL: https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/ (дата обращения: 20.09.2024).

38. Visual Paradigm online [Электронный ресурс]. URL: https://online.visual-paradigm.com/ (дата обращения: 20.09.2024).

39. Waheed A. How to Set Up a Database API in Minutes [Электронный ресурс]. URL: https://apidog.com/blog/set-up-database-api/ (дата обращения: 20.09.2024).

40. Web Application ERP Integration [Электронный ресурс]. URL: https://www.buink.com/web-application-erp-integration/ (дата обращения: 20.09.2024).

41. What Does a Web-Based ERP System Mean? https://wezom.com/blog/what-does-a-web-based-erp-system-mean (дата обращения: 20.10.2024).

42. What is iPaaS? Guide to integration platform as a service [Электронный ресурс]. URL: https://www.techtarget.com/searchcloudcomputing/definition/iPaaS-integration-platform-as-a-service (дата обращения: 20.09.2024).

43. Yehia M. Helmy, Mohamed I. Marie, Sara M. Mosaad, “An Integrated ERP with Web Portal”, Advanced Computing: An International Journal (ACIJ), Vol.3, No.5, 2012.