**Министерство общего и профессионального образования**

**Ростовской области**

**государственное бюджетное профессиональное образовательное учреждение**

**Ростовской области**

**«Волгодонский техникум информационных технологий, бизнеса и дизайна**

**имени В.В. Самарского»**

ПРОЕКТ ДОПУЩЕН К ЗАЩИТЕ

(подпись, дата)

**КУРСОВОЙ ПРОЕКТ**

По МДК.02.01 Технология разработки программного обеспечения

Специальность 09.02.07 Информационные системы и программирование

На тему Анализ требований разработки программного обеспечения

автоматизированного учета по поставке продуктов питания в магазин.

Выполнил Д.Е.Сидора

(подпись)

Руководитель А.И. Аксенов

(подпись)

Работа защищена

(оценка, дата, подпись)

**Министерство общего и профессионального образования**

**Ростовской области**

**государственное бюджетное профессиональное образовательное учреждение**

**Ростовской области**

**«Волгодонский техникум информационных технологий, бизнеса и дизайна**

**имени В.В. Самарского»**

**ОДОБРЕНО: УТВЕРЖДАЮ:**

цикловой комиссией профессионального

информационного цикла И.о.. зам директора по УР

Протокол № \_\_ от «\_\_» \_\_\_\_\_\_\_\_ 202\_ г. \_\_\_\_\_\_\_\_ /Н.В.Погорелова/

Председатель ЦК \_\_\_\_\_\_\_\_ /И.Н.Власенко/

**ЗАДАНИЕ**

**НА КУРСОВОЙ ПРОЕКТ**

Студента Сидора Дмитрия Евгеньевича

Группы ИСП-22

Специальность 09.02.07 Информационные системы и программирование

**ТЕМА ЗАДАНИЯ**: Анализ требований разработки программного обеспечения автоматизированного учета по поставке продуктов питания в магазин.

**СОДЕРЖАНИЕ ЗАДАНИЯ**:

1. Графическая часть в объеме \_\_\_\_ листов формата А1:

2. Пояснительная записка в объеме 27 листов, содержащая разделы:

 разработка технического задания для проектируемого программного продукта;

 анализ требований и определений спецификаций к эскизному проекту соответствия с предметной областью;

 проектирование эскизного проекта при структурном анализе;

 разработка схемы базы данных;

 заключение.

3. Руководитель: преподаватель, Аксенов А.И.

(должность, фамилия, инициалы, подпись)

Задание выдано: «18» сентября 2024 г.

Срок выполнения: «28» апреля 2025 г.

**СОДЕРЖАНИЕ**

Введение 5

1 Разработка технического задания для проектируемого программного продукта 5

1.1 Разработка технического задания 6-9

1.2 Построение систем на основе информационных технологий 9

1.2.1 Функционально-модульная и объектно-ориентированная технология проектирования информационной системы 10

1.3 Структурный подход к проектированию информационной системы 10-12

2 Анализ требований и определение спецификаций к эскизному проекту в соответствии с предметной областью 12-16

2.1 Диаграмма потоков данных 12-13

2.2 Диаграмма «сущность-связь» 13-14

2.3 Диаграмма переходов состояния 14-15

2.4 Функциональная диаграмма 15-16

3 Проектирование эскизного проекта при структурном анализе 16

3.1 Структурная схема разрабатываемой информационной системы 16-17

4 Разработка схемы базы данных 17-21

4.1 Проектирование реляционной базы данных на основе логической и инфологической моделей данных 17-19

4.2 Реализация базы данных в СУБД Microsoft SQL Server 19-21

Заключение 21

Список использованных источников 22-23

ПРИЛОЖЕНИЕ А 23

ПРИЛОЖЕНИЕ Б 23 - 34

Введение

Актуальность темы курсового проекта связан с тем, что в настоящее время процесс автоматизации учета поставки различных продуктов питания в разные магазины. Цель моего курсового проекта заключается в том, чтобы отслеживать в какие магазины поставлять какие продукты питания. Ожидаемые результаты в данной проекте заключатся в том, чтобы автоматизировать работу складского учета поставки продуктов питания, а также написание базы данных, построение диаграмм.

Анализ существующих решений состоит из описания программного продукта, используемого в автоматизированном отслеживании поставки продуктов питания в любой из магазинов. В сравнительном анализе существующих решений необходимы: программный продукт, его база данных, и код программы. В выборе наиболее подходящих решений должно присутствовать не менее двух решений. А заключение должно состоять из результата работы, выводов и рекомендаций. Результат работы должен быть оформлен без сбоев. Выводы должны быть оформлены и начинаться со слов: таким образом, моя программа помогает отслеживать поставки продуктов питания в любой магазин.

Основные цели курсового проекта заключаются в:

− разработка системы автоматизации;

− учета поставки продуктов питания в магазины;

− деятельность магазинов по учету поступления продуктов питания.

Задачи курсового проекта:

− проведение анализа предметной области;

− произвести учет продуктов питания в магазины;

− написание базы данных;

− построение диаграмм и схем;

− разработка технического задания.

1 Разработка технического задания для проектируемого программного продукта

Разработка технического задания для проектируемого программного продукта является важным этапом в создании любой программной системы. Оно позволяет четко определить требования к продукту, что повышает эффективность и качество его разработки, а также позволяет избежать проблем в процессе эксплуатации [3].

Техническое задание должно содержать информацию о целях создания продукта, его функциональности, технических требованиях и условиях использования. Важно также учесть факторы, которые могут повлиять на результат разработки, такие как бюджет, сроки выполнения работ и возможности команды разработчиков.

Техническое задание должно быть разработано с учетом специфики конкретного проекта и предъявляемых к нему требований. Оно является основным документом, на основе которого проводится оценка выполнения работ, а также контроль за соблюдением сроков и бюджета.

1.1 Разработка технического задания

1) Введение

− описание предметной области;

− цель и задачи курсового проекта;

− ожидаемые результаты.

2) Требования к программному обеспечению

− функциональные требования;

− требования к пользовательскому интерфейсу;

− требования к базе данных.

3) Анализ существующих решений

− описание программных продуктов, используемых в автоматизированном учете по поставке продуктов питания в магазины;

− сравнительный анализ существующих решений;

− выбор наиболее подходящего решения.

4) Заключение

− результаты работы;

− выводы и рекомендации.

Ожидаемые результаты:

− анализ потребностей продуктовых магазинов в программном обеспечении для автоматизации учета по поставке продуктов питания и их характеристик;

− определение потенциальных вариантов реализации функциональности программного обеспечения для автоматизации поддержки торговых точек;

− анализ полученных результатов.

1) Требования к программному обеспечению

Для разрабатываемого программного обеспечения автоматизации учета поставки продуктов питания, необходимо предусмотреть следующие функции:

1) регистрация товара – возможность добавления информации о товарах в базу данных с основными данными, такими как наименования, дата производства, срок годности, производитель, цена;

2) создание лицевого счета магазина – возможность создать лицевой счет с помощью автоматической поддержки путем оформления заявки, после оформления заявки каждому клиенту с лицевым счетом должен присваиваться автоматически сгенерированный идентификационный номер.

3) управление лицевым счетом – возможность осуществлять пополнения товарами магазинов.

4) запрос информации о клиенте – возможность запросить информацию о клиенте такие как идентификационный номер клиента, реквизиты лицевого счета, баланс лицевого счета.

5) помощь в решении вопросов – помощь в решении вопросов клиентов с помощью анализа вопроса, каждое слово в вопросе должно анализироваться

программой на наличие ключевых слов, например, таких как «уточнить данные о наличии товара», «проблема с доставкой товара в магазин» и тому подобное, в зависимости от ключевого слова ПО должно выбирать пути дальнейшего действия.

Программное обеспечение должно иметь простой, интуитивно понятный и удобный пользовательский интерфейс. В нём должны быть предусмотрены все необходимые элементы управления и информации, позволяющие максимально облегчить получение той или иной помощи и информации о клиенте. Интерфейс также должен быть адаптивным и подходить для работы на различных устройствах.

Для хранения и управления информацией о клиентах и запросах необходимо использовать базу данных. База данных должна быть надёжной, безопасной, и иметь быстрый доступ к данным. База данных должна обеспечивать возможность эффективного хранения, обработки и анализа данных, а также возможность выгрузки и импорта информации.

Таким образом, разработанное программное обеспечение должно удовлетворять всех предъявляемых требований в функциональных возможностях, пользовательском интерфейсе и базе данных для успешного управления автоматизированной поддержкой клиентов банка.

2) Анализ существующих решений.

Созданием программного обеспечения для учета поставки продуктов питания в магазины в мире занимаются большое количество IT. Примером является программа «Мой Склад», которая имеет большие возможности управления учетом и функциями. Программа написана для мобильных устройств. Вся информация о клиентах хранится в безопасном хостинге. Программное обеспечение было разработано для небольших магазинов и организаций. Бесплатная версия данного ПО, подходит для магазинов с одним сотрудником. Платные версии рассчитаны на любое количество точек и персонала.

3) Сравнительный анализ существующих решений.

Сравнительный анализ существующих решений показал, что все они имеют свои преимущества и недостатки. EKAM – для небольших магазинов, прост в эксплуатации, удобный интерфейс. «Мой Склад» также для небольших магазинов, имеющих одного продавца у него больше возможностей чем у EKAM и он может работать на мобильных устройствах.

После проведения сравнительного анализа мы выбрали систему автоматизированной поддержки клиентов как наиболее подходящее. Она имеет достаточный функционал для поддержки клиентов.

4) Заключение

Мы провели анализ существующих решений, описали программные продукты, уже используемые в организациях, и сравнили их функционал. Выбор разработки программного обеспечения для автоматизированной поддержки учета поступления продуктов питания в магазины, был основан на ее достаточном функционале и простоте в использовании.

Выводы:

− необходимость использования, специализированного ПО для автоматизированной поддержки клиентов, является актуальной;

− в организациях уже используются несколько программных продуктов для автоматизированной поддержки клиентов, но они имеют свои преимущества и недостатки;

− после проведения сравнительного анализа существующих решений мы выбрали разработки программного автоматизированной поддержки клиентов как наиболее подходящее решение на основе ее функциональности, доступности и простоты в использовании.

Рекомендации:

− при выборе программного обеспечения для поддержки клиентов, необходимо учитывать его функциональные возможности и цену;

− рекомендуется проводить обучение клиента перед использованием новой системы поддержки клиентов;

− необходимо следить за обновлением системы и ее адаптацией к изменяющимся требованиям и потребностям организации.

1.2 Построение систем на основе информационных технологий

Построение систем основывается на принципах и обеспечивает эффективность и устойчивость:

 иерархия управление задачами и использование источников данных должны подчиняться иерархическому принципу;

− принцип агрегированности данных: система должна учитывать запросы на разных уровнях иерархии, обеспечивая агрегацию данных;

− избыточность (построение с учетом не только текущих, но и будущих задач);

− конфиденциальность: обеспечение защиты конфиденциальных данных пользователей;

− адаптивность: к изменяющимся запросам система должна быть гибкой и легко адаптируемой к изменениям в требованиях;

− согласованность и информационное единство использование системы показателей для предотвращения несогласованных действий и вывода неправильной информации;

− открытость системы возможность расширения данных для пополнения информации.

1.2.1 Функционально-модульная и объектно-ориентированная технология проектирования информационной системы

Функционально-модульная технология проектирования информационной системы основана на разделении системы на функциональные модули, каждый из которых выполняет конкретную задачу и взаимодействует с другими модулями.

Простая структура облегчает разработку, но могут возникнуть проблемы при масштабировании.

Объектно-ориентированная технология проектирования информационной системы основана на понятии объекта, обладающего состоянием и поведением.

Объекты взаимодействуют и наследуют свойства друг от друга. Позволяет создавать сложные и легко масштабируемые системы, но требует более высокого уровня абстракции.

Функционально-модульная технология проектирования информационной системы может быть применена в маломасштабных проектах, где система имеет простую структуру и задачи. Например, это может быть система управления продажами в небольшом магазине. Объектно-ориентированная технология проектирования информационной системы может быть применена в крупных проектах, где система имеет сложную структуру и компоненты. Например, это может быть система управления производством на крупном предприятии, например, гипер-маркеты.

Примеры языков программирования для функционально-модульной технологии: Pascal, C, Fortran, Prolog. Примеры языков программирования для объектно-ориентированной технологии: Java, C#, C++, Ruby.

1.3 Структурный подход к проектированию информационной системы

Структурный подход включает в себя разделение системы на автоматизируемые функции, создавая иерархические древовидные структуры. Принципы включают «разделяй и властвуй», иерархическое упорядочение, абстрагирование, непротиворечивость и структурирование данных. Этот подход применяется для проектирования малых и средних информационных систем, но может быть сложным для крупных проектов. Структурный подход представляет собой метод проектирования ИС, основанный на декомпозиции системы на автоматизируемые функции: система разбивается на функциональные

подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее. Процесс разбиения продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимоувязаны [5].

Все наиболее распространенные методы структурного подхода базируются на ряде общих принципов. Базовыми принципами являются:

 принцип «разделяй и властвуй»;

 принцип иерархического упорядочения — принцип организации составных частей системы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне.

Выделение двух базовых принципов не означает, что остальные принципы являются второстепенными, поскольку игнорирование любого из них может привести к непредсказуемым последствиям (в том числе и к провалу всего проекта). Основными из этих принципов являются:

 принцип абстрагирования —¬ выделение существенных аспектов системы и отвлечение от несущественных;

 принцип непротиворечивости — обоснованность и согласованность элементов системы;

 принцип структурирования данных — данные должны быть структурированы и иерархически организованы.

Структурный подход широко используется при проектировании небольших и средних ИС, таких как системы учета и аналитики. Однако, данный подход не подходит для больших проектов с множеством взаимодействующих компонентов, так как проектирование и отладка подобных систем может оказаться трудной задачей.

2 Анализ требований и определение спецификаций к эскизному проекту в соответствии с предметной областью

Анализ требований – это важнейшая часть процесса разработки программного обеспечения (ПО), которая задаёт вектор всей остальной работе.

Основная задача анализа требований – это получение списка не дублируемых требований к разрабатываемому ПО.

Согласно предметной области, можно определить следующие требования:

база данных (БД) должна хранить данные:

 информация о поставщиках;

 информация о товарах;

 информация о заказах товаров;

 история заказов в магазины;

 информация о товарах на складе;

 аналитическая информация.

2.1 Диаграмма потоков данных

Диаграмма DFD наглядно отображает течение информации в пределах процесса или системы. Для изображения входных и выходных данных, точек хранения информации и путей ее передвижения между источниками и пунктами доставки в таких диаграммах применяются стандартные фигуры, такие как прямоугольники и круги, а также стрелки и краткие текстовые метки [7].

Метод DFD разбивает высокоуровневую диаграмму потока данных на набор более подробных диаграмм, обеспечивая общее представление о всей системе, а также более подробную декомпозицию. Дает общее представление о системе в целом, а также более подробную декомпозицию и, при необходимости, более подробную разбивку и описание отдельных действий для облегчения разъяснения, и понимания.

Диаграмма потоков данных может быть представлена в нескольких уровнях абстракции, каждый из которых описывает систему с разной детализацией. Более высокий уровень абстракции предоставляет общий обзор системы, в то время как более низкий уровень детализирует каждый процесс и поток данных.

Основными преимуществами использования диаграммы потоков данных являются улучшение понимания потоков данных в системе, возможность обнаружения ошибок и недостатков в системе на ранних этапах проектирования, а также возможность проведения анализа производительности и оптимизации системы.

На основе представленных сущностей и связей между ними, можно разработать диаграмму потоков данных, показывающую взаимодействие между сущностями в системе. Она может иметь несколько уровней детализации, что позволяет описывать систему на разных уровнях абстракции.

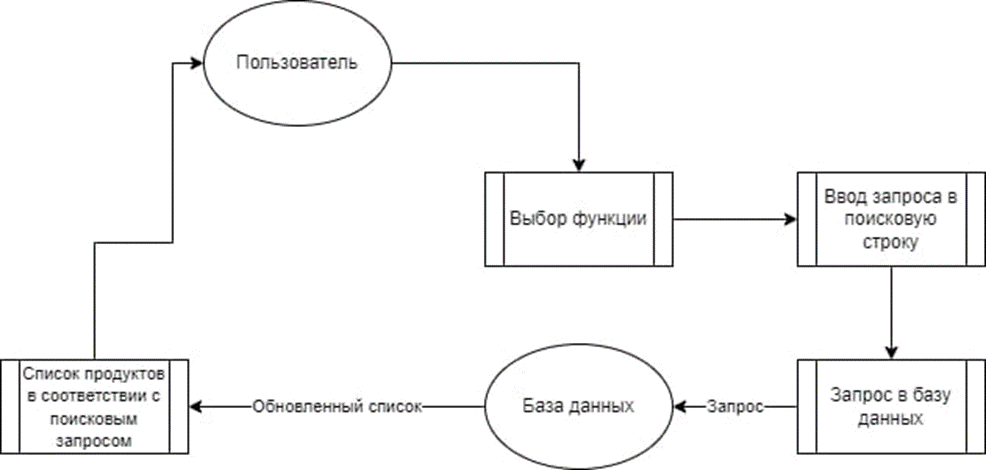


Рисунок 2.1 — Диаграмма потоков данных

2.2 Диаграмма «сущность-связь»

Схема «сущность-связь» (также ERD или ER-диаграмма) — это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы (или ER-модели) полагаются на стандартный набор символов, включая прямоугольники, ромбы, овалы и соединительные линии, для отображения сущностей, их атрибутов и связей. Эти диаграммы устроены по тому же принципу, что и грамматические структуры: сущности выполняют роль существительных, а связи — глаголов [8].

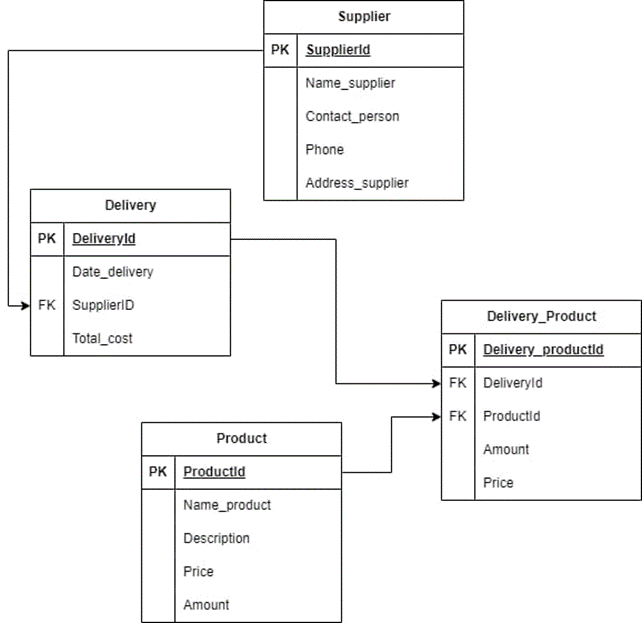


Рисунок 2.2 — Диаграмма «сущность-связь»

2.3 Диаграмма переходов состояний

Диаграммы переходов состояний (STD) — схема состояний и переходов. Предназначены для моделирования и документирования аспектов систем, зависящих от времени или реакции на событие. Они позволяют осуществлять декомпозицию управляющих процессов и описывают отношения между входными и выходными управляющими потоками для управляющего процесса [9].

ER – диаграммы широко используются при проектировании баз данных для описания структуры данных и отношений между ними. Они позволяют определить все сущности, которые должны быть включены в базу данных, а также их атрибуты и отношения между ними.

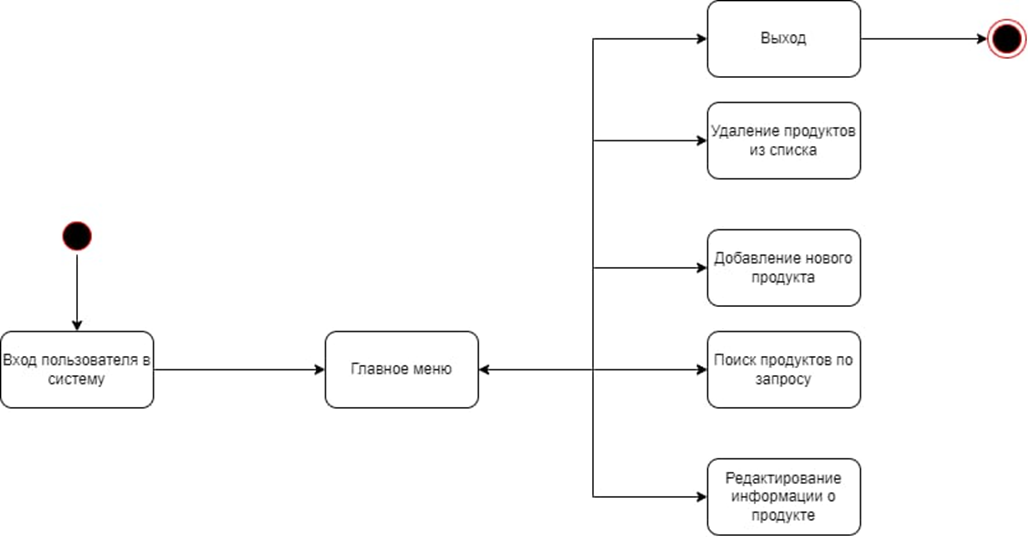


Рисунок 2.3 — Диаграмма переходов состояний

2.4 Функциональная диаграмма

Функциональная диаграмма (Functional Flow Block Diagram, FFBD) – это диаграммы, которые описывают особенности функционирования, показывают, как устроена работа того или иного объекта. Они демонстрируют пользователю функциональность так, чтобы это было понятно [11].

Функциональная диаграмма включает в себя блоки функций, связанные линиями, которые показывают порядок выполнения функций. Каждый блок представляет отдельную функцию, которая может быть декомпозирована на более мелкие подфункции. Функциональная диаграмма является частью метода IDEF (Integrated Definition for Function Modeling), который используется для моделирования бизнес-процессов, проектов и систем. Функциональными называют диаграммы, в первую очередь отражающие взаимосвязи функций разрабатываемого программного обеспечения.

Проектирование эскизного проекта при структурном анализе

Эскизным проектом называют пакет конструкторской документации, создаваемый на стадии разработки автоматизированной системы. Цель создания этих документов – установить принципиальные, конструктивные решения, представить их для ознакомления с принципами работы и устройством разрабатываемой системы. Также этим проектом может рассматриваться несколько вариантов устройства АСУ [11].

Эскизный проект на автоматизированную систему разрабатывают перед техническим проектом или вместе с ним. Эта документация может и не оформляться в случае, если ею не может быть предоставлено никаких новых данных – ее необходимость устанавливается тех заданием.

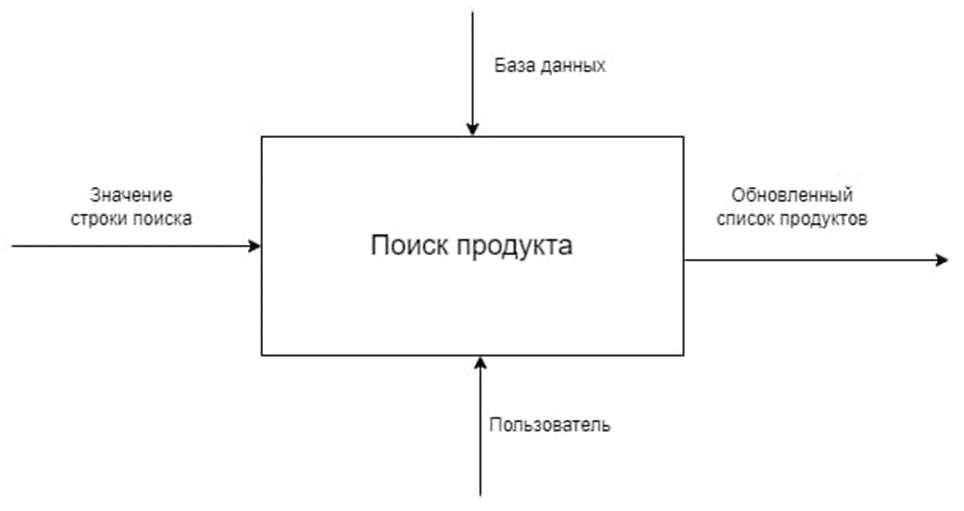


Рисунок 2.4 — Функциональная диаграмма

3 Проектирование эскизного проекта при структурном анализе

Эскизным проектом называют пакет конструкторской документации, создаваемый на стадии разработки автоматизированной системы. Цель создания этих документов – установить принципиальные, конструктивные решения, представить их для ознакомления с принципами работы и устройством разрабатываемой системы. Также этим проектом может рассматриваться несколько вариантов устройства АСУ [11].

Эскизный проект на автоматизированную систему разрабатывают перед техническим проектом или вместе с ним. Эта документация может и не оформляться в случае, если ею не может быть предоставлено никаких новых данных – ее необходимость устанавливается тех заданием.

3.1 Структурная схема разрабатываемой информационной системы

Структурная схема — это совокупность элементарных звеньев объекта и связей между ними, один из видов графической модели. Под элементарным звеном подразумевается часть объекта, системы управления и т. д., которая реализует элементарную функцию [12].

4.Разработка схемы базы данных

Для разработки схемы базы данных автоматизации поддержки руководителей магазинов (клиентов), нужно учитывать множество факторов. Вот некоторые ключевые моменты, которые могут помочь:

1) Определите основные объекты данных, такие как магазины, счета для оплаты заказов и заявки магазинов.

2) Разбейте каждый объект данных на атрибуты, которые необходимо отслеживать и хранить.

3) Установите связи между объектами данных. Например, клиент может иметь несколько заявок, а каждая заявка может иметь несколько оплат.

4) Разработайте таблицы или коллекции для каждого объекта данных и их атрибутов.

5) Определите типы данных для каждого атрибута, чтобы правильно хранить информацию.

6) Разработайте подходящую структуру базы данных, такую как реляционную (SQL) или документоориентированную (NoSQL), в зависимости от требований проекта.

7) Обеспечьте безопасность данных, реализуя соответствующие механизмы аутентификации и авторизации.

8) Проверьте и оптимизируйте схему базы данных для эффективного доступа к данным.

4.1 Проектирование реляционной базы данных на основе логической и инфологической моделей данных

Проектирование базы данных (БД) – одна из наиболее сложных и ответственных задач, связанных с созданием информационной системы (ИС). В результате её решения должны быть определены содержание БД, эффективный

для всех её будущих пользователей способ организации данных и инструментальные средства управления данными [13].

Основная цель процесса проектирования БД состоит в получении такого проекта, который удовлетворяет следующим требованиям:

1) Корректность схемы БД, т.е. база должна быть гомоморфным образом моделируемой предметной области (ПО), где каждому объекту предметной области соответствуют данные в памяти ЭВМ, а каждому процессу – адекватные процедуры обработки данных.

2) Обеспечение ограничений (на объёмы внешней и оперативной памяти и другие ресурсы вычислительной системы).

3) Эффективность функционирования (соблюдение ограничений на время реакции системы на запрос и обновление данных).

4) Защита данных (от аппаратных и программных сбоев и несанкционированного доступа).

5) Простота и удобство эксплуатации.

6) Гибкость, т.е. возможность развития и адаптации к изменениям предметной области и/или требований пользователей.

7) Процесс проектирования включает в себя следующие этапы.

8) Инфологическое проектирование.

9) Определение требований к операционной обстановке, в которой будет функционировать информационная система

10) Выбор системы управления базой данных (СУБД) и других инструментальных программных средств.

11) Логическое проектирование БД.

12) Физическое проектирование БД.

Инфологический подход не предоставляет формальных способов моделирования реальности, но он закладывает основы методологии проектирования баз данных.

Основными задачами инфологического проектирования являются определение предметной области системы и формирование взгляда на ПО с

позиций сообщества будущих пользователей БД, т.е. инфологической модели ПО.

Инфологическая модель ПО представляет собой описание структуры и динамики ПО, характера информационных потребностей пользователей в терминах, понятных пользователю и не зависимых от реализации БД. Это описание выражается в терминах не отдельных объектов ПО и связей между ними, а их типов, связанных с ними ограничений целостности и тех процессов, которые приводят к переходу предметной области из одного состояния в другое.

Рассмотрим основные подходы к созданию инфологической модели предметной области.

Функциональный подход к проектированию БД. Этот метод реализует принцип «от задач» и применяется тогда, когда известны функции некоторой группы лиц и/или комплекса задач, для обслуживания информационных потребностей которых создаётся рассматриваемая БД.

Предметный подход к проектированию БД применяется в тех случаях, когда у разработчиков есть чёткое представление о самой ПО и о том, какую именно информацию они хотели бы хранить в БД, а структура запросов не определена или определена не полностью. Тогда основное внимание уделяется исследованию ПО и наиболее адекватному её отображению в БД с учётом самого широкого спектра информационных запросов к ней.

Метод «сущность–связь» (entity–relation, ER–method) является комбинацией двух предыдущих и обладает достоинствами обоих. Этап инфологического проектирования начинается с моделирования ПО. Проектировщик разбивает её на ряд локальных областей, каждая из которых (в идеале) включает в себя информацию, достаточную для обеспечения запросов отдельной группы будущих пользователей или решения отдельной задачи (подзадачи). Каждое локальное представление моделируется отдельно, затем они объединяются.

Выбор локального представления зависит от масштабов ПО. Обычно она разбивается на локальные области таким образом, чтобы каждая из них соответствовала отдельному внешнему приложению и содержала 6-7 сущностей. На этапе логического проектирования разрабатывается логическая структура БД, соответствующая логической модели ПО. Решение этой задачи существенно зависит от модели данных, поддерживаемой выбранной СУБД.

Результатом выполнения этого этапа являются схемы БД концептуального и внешнего уровней архитектуры, составленные на языках определения данных (DDL, Data Definition Language), поддерживаемых данной СУБД.

4.2 Реализация базы данных в СУБД Microsoft SQL Server

Для выполнения проекта была выбрана СУБД Microsoft SQL Server. Схема базы данных представлена на рисунке 4.2.1.

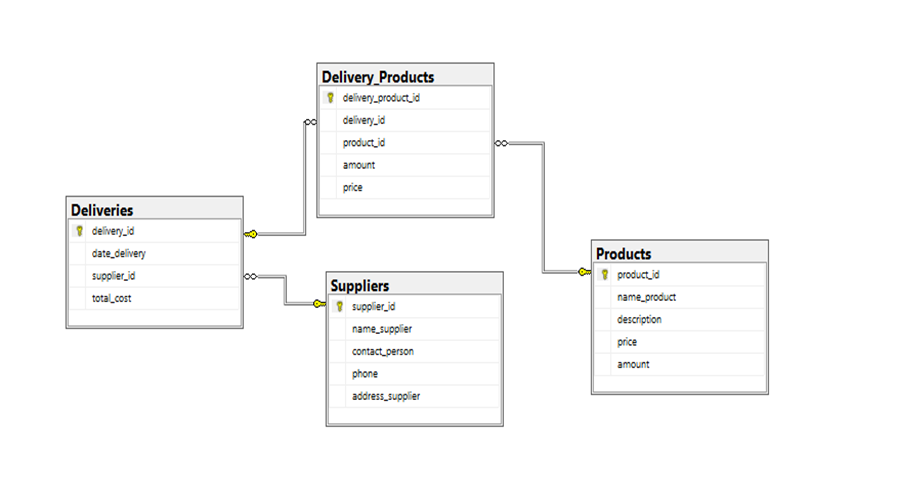


Рисунок 4.2.1 — Схема базы данных

Структура таблиц «данные клиентов», «история обслуживания», «результаты работы с клиентами», «история оплаты», «аналитическая информация» представлена на рисунках 4.2.2–4.2.5

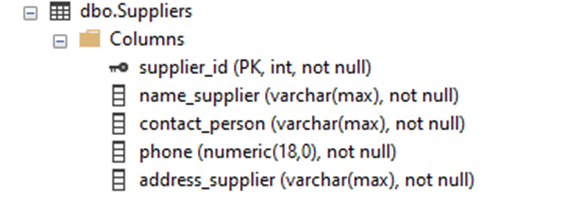


Рисунок 4.2.2 — Таблица «Информация о товарах»

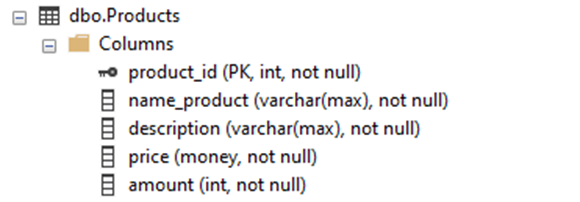


Рисунок 4.2.3 — Таблица «Результаты отслеживания поставки продуктов в продуктовые магазины»

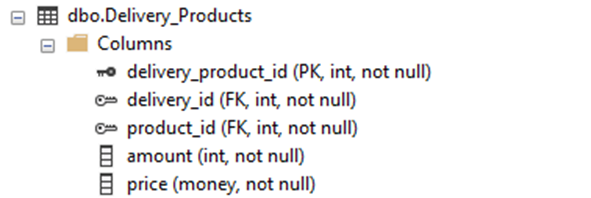


Рисунок 4.2.4 — Таблица «История поставки»

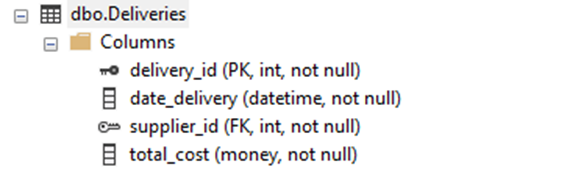


Рисунок 4.2.5 — Таблица «История отслеживания»

Заключение

Проведенный в ходе курсового проекта анализ требований разработки программного обеспечения автоматизированная по учету и поставке продуктов

Список использованных источников питания в магазины, были выявлены основные требования к функциональной и нефункциональной составляющим системы, также был проведен анализ существующих программных продуктов с целью сопоставления их функциональности с выделенными заказчиком требованиями.

В дальнейшем в базу данных будут добавляться новые сущности и записи, необходимые для расширения возможностей работы с данными и улучшением автоматизации поддержки клиентов магазина.

В процессе работы были приобретены практические навыки в работе в СУБД SQL Server.

Цель курсового проекта достигнута. Задачи выполнены.

Список использованной литературы

1 Гниденко И.Г., Павлов Ф.Ф., Федоров Д.Ю. Технология разработки программного обеспечения – М.: Издательство Юрайт, 2020.–235с.

2 Рудаков А.В. Технология разработки программных продуктов – М.: Издательский центр «Академия», 2019.–208с.

3 Федорова Г.Н. Участие в интеграции программных модулей – М.: Издательство «Академия», 2019.–304с.

Интернет-ресурсы

1 Microsoft Word. — Текст: электронный // https://ru.wikipedia.org: [сайт]. —URL: https://ru.wikipedia.org/wiki/Microsoft\_Word

2 Microsoft Excel — Текст: электронный // https://ru.wikipedia.org: [сайт]. —URL: https://ru.wikipedia.org/wiki/Microsoft\_Excel

3 Принципы построения информационных систем. — Текст: электронный // https://studfile.net: [сайт]. — URL: https://studfile.net/preview/7418698/page:13/

4 Функционально-модульная технология проектирования

информационной системы. — Текст: электронный // http://vit-prog.narod.ru: [сайт]. — URL: http://vit-prog.narod.ru/page/TRPP/section\_1/subject\_1.3.htm

5 Структурный подход к проектированию программного обеспечения. — Текст: электронный // https://studfile.net: [сайт]. — URL: https://studfile.net/preview/1966781/page:2/

6 Анализ требований. — Текст: электронный // https://stepanovd.com: [сайт]. — URL: https://stepanovd.com/training/20-vkr/65-vkrb-2017-1-oreshkina

7 Диаграмма DFD. — Текст: электронный // https://www.lucidchart.com: [сайт]. — URL: <https://www.lucidchart.com/pages/ru/диаграмма-dfd>

8 ER-диаграмма. — Текст: электронный // https://www.lucidchart.com: [сайт]. — URL: https://www.lucidchart.com/pages/ru/erd-диаграмма

9 State & Transition Diagram. — Текст: электронный // https://habr.com: [сайт]. — URL: https://habr.com/ru/articles/548192/

10 Функциональные диаграммы. — Текст: электронный // https://studfile.net: [сайт]. — URL: https://studfile.net/preview/5125263/page:11/

11 Эскизный проект. — Текст: электронный // https://www.swrit.ru: [сайт]. — URL: https://www.swrit.ru/eskiznyj-proekt.html

12 Структурная схема. — Текст: электронный // https://ru.wikipedia.org: [сайт]. — URL: https://ru.wikipedia.org/ wiki/Структурная\_схема

13 Проектирование реляционных баз данных. — Текст: электронный // http://rema44.ru: [сайт]. — URL: <http://rema44.ru/resurs/study/dbprj/dbprj.html>

ПРИЛОЖЕНИЕ А

(обязательно)

Техническое задание

Разработка технического задания

Введение

− описание предметной области;

− цель и задачи курсовой работы;

− ожидаемые результаты.

Требования к программному обеспечению

− функциональные требования;

− требования к пользовательскому интерфейсу;

− требования к базе данных.

Анализ существующих решений

− описание программных продуктов, используемых в автоматизированной поддержке по учету поставки продуктов питания в магазины;

− сравнительный анализ существующих решений;

− выбор наиболее подходящего решения.

Заключение

− результаты работы;

− выводы и рекомендации.

ПРИЛОЖЕНИЕ Б

(обязательно)

Скрипт создания схемы данных

Код на Microsoft SQL Server

CREATE TABLE Products

(

product\_id INT PRIMARY KEY IDENTITY NOT NULL,

name\_product VARCHAR(MAX) NOT NULL,

description VARCHAR(MAX) NOT NULL,

price MONEY NOT NULL,

amount INT NOT NULL

);

CREATE TABLE Suppliers

(

supplier\_id INT PRIMARY KEY IDENTITY NOT NULL,

name\_supplier VARCHAR(MAX) NOT NULL,

contact\_person VARCHAR(MAX) NOT NULL,

phone NUMERIC NOT NULL,

address\_supplier VARCHAR(MAX) NOT NULL

);

CREATE TABLE Deliveries

(

delivery\_id INT PRIMARY KEY IDENTITY NOT NULL,

date\_delivery DATETIME NOT NULL,

supplier\_id INT NOT NULL,

FOREIGN KEY (supplier\_id) REFERENCES Suppliers(supplier\_id),

total\_cost MONEY NOT NULL

);

CREATE TABLE Delivery\_Products

(

delivery\_product\_id INT PRIMARY KEY IDENTITY NOT NULL,

delivery\_id INT NOT NULL,

product\_id INT NOT NULL,

FOREIGN KEY (delivery\_id) REFERENCES Deliveries (delivery\_id),

FOREIGN KEY (product\_id) REFERENCES Products(product\_id),

amount INT NOT NULL,

price MONEY NOT NULL

);

class Program

{

static void Main(string[] args)

{

Storage storage = new Storage();

bool running = true;

while (running)

{

Console.WriteLine("1. Add Product");

Console.WriteLine("2. Add Supply");

Console.WriteLine("3. View Products");

Console.WriteLine("4. View Supplies");

Console.WriteLine("5. Exit");

Console.Write("Choose an option: ");

var choice = Console.ReadLine();

switch (choice)

{

case "1":

Console.Write("Enter product name: ");

var name = Console.ReadLine();

Console.Write("Enter product price: ");

var price = decimal.Parse(Console.ReadLine());

Console.Write("Enter product quantity: ");

var quantity = int.Parse(Console.ReadLine());

storage.AddProduct(name, price, quantity);

Console.WriteLine("Product added.");

break;

case "2":

Console.Write("Enter product ID for supply: ");

var productId = int.Parse(Console.ReadLine());

Console.Write("Enter quantity supplied: ");

var quantitySupplied = int.Parse(Console.ReadLine());

storage.AddSupply(productId, quantitySupplied);

Console.WriteLine("Supply added.");

break;

case "3":

Console.WriteLine("Products:");

foreach (var product in storage.GetProducts())

{

Console.WriteLine(product);

}

break;

case "4":

Console.WriteLine("Supplies:");

foreach (var supply in storage.GetSupplies())

{

Console.WriteLine(supply);

}

break;

case "5":

running = false;

break;

default:

Console.WriteLine("Invalid option. Please try again.");

break;

}

Console.WriteLine();

}

}

}

}

namespace Products

{

public class Products

{

public int Id { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

public int Quantity { get; set; }

public override string ToString()

{

return $"{Id}: {Name} - {Price:C} (Quantity: {Quantity})";

}

}

}

namespace Products

{

public class Supply

{

public int Id { get; set; }

public int ProductId { get; set; }

public int QuantitySupplied { get; set; }

public DateTime SupplyDate { get; set; }

public override string ToString()

{

return $"Supply ID: {Id}, Product ID: {ProductId}, Quantity: {QuantitySupplied}, Date: {SupplyDate}";

}

}

}

namespace Products

{

public class Storage

{

private List<Product> products = new List<Product>();

private List<Supply> supplies = new List<Supply>();

private int productIdCounter = 1;

private int supplyIdCounter = 1;

public void AddProduct(string name, decimal price, int quantity)

{

var product = new Product

{

Id = productIdCounter++,

Name = name,

Price = price,

Quantity = quantity

};

products.Add(product);

}

public void AddSupply(int productId, int quantitySupplied)

{

var supply = new Supply

{

Id = supplyIdCounter++,

ProductId = productId,

QuantitySupplied = quantitySupplied,

SupplyDate = DateTime.Now

};

supplies.Add(supply);

var product = products.FirstOrDefault(p => p.Id == productId);

if (product != null)

{

product.Quantity += quantitySupplied;

}

}

public List<Product> GetProducts()

{

return products;

}

public List<Supply> GetSupplies()

{

return supplies;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Products

{

public class Supply

{

public int Id { get; set; }

public int ProductId { get; set; }

public int QuantitySupplied { get; set; }

public DateTime SupplyDate { get; set; }

public override string ToString()

{

return $"Supply ID: {Id}, Product ID: {ProductId}, Quantity: {QuantitySupplied}, Date: {SupplyDate}";

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Products

{

public class Products

{

public int Id { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

public int Quantity { get; set; }

public override string ToString()

{

return $"{Id}: {Name} - {Price:C} (Quantity: {Quantity})";

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Products

{

public class Storage

{

private List<Product> products = new List<Product>();

private List<Supply> supplies = new List<Supply>();

private int productIdCounter = 1;

private int supplyIdCounter = 1;

public void AddProduct(string name, decimal price, int quantity)

{

var product = new Product

{

Id = productIdCounter++,

Name = name,

Price = price,

Quantity = quantity

};

products.Add(product);

}

public void AddSupply(int productId, int quantitySupplied)

{

var supply = new Supply

{

Id = supplyIdCounter++,

ProductId = productId,

QuantitySupplied = quantitySupplied,

SupplyDate = DateTime.Now

};

supplies.Add(supply);

var product = products.FirstOrDefault(p => p.Id == productId);

if (product != null)

{

product.Quantity += quantitySupplied;

}

}

public List<Product> GetProducts()

{

return products;

}

public List<Supply> GetSupplies()

{

return supplies;

}

}

}