**ДИПЛОМЕН ПРОЕКТ**

**ТЕМА:**

**РАЗРАБОТВАНЕ НА НАСТОЛНО ПРИЛОЖЕНИЕ ЗА ФИРМА ЗА ЕЛЕКТРОНИКА**

**ДИПЛОМАНТ: Константин Любенов Станоев**

**СПЕЦИАЛНОСТ: *код 4810201 „Системно програмиране“***

**ПРОФЕСИЯ:** ***код 481020 „Системен програмист“***

**РЪКОВОДИТЕЛ-КОНСУЛТАНТ: инж. Иван Киров**

**СОФИЯ**

**2024**

ЗАДАНИЕ

**Раздел I – Теоретична част**

1. **Увод:**

В съвременната дигитална епоха ефективното управление на информацията е от съществено значение за успешното функциониране на бизнеса. Компаниите, занимаващи се с търговия и дистрибуция на компютърна техника, ежедневно обработват големи обеми от данни, свързани с продукти, клиенти, поръчки и доставчици. Тази информация трябва да бъде надеждно съхранявана, лесно достъпна и подлежаща на ефективна обработка, за да се гарантира бързо и безпроблемно обслужване. В този контекст необходимостта от специализирани софтуерни решения, които оптимизират тези процеси, става все по-голяма.

Настоящият дипломен проект е посветен на разработването на настолно приложение за фирма, специализирана в продажбата на компютърна техника, включително **настолни компютри, лаптопи и сървъри**. Основната цел на приложението е да автоматизира ключови бизнес процеси, като осигури интуитивен интерфейс за управление на **продуктовия каталог, клиентските профили, поръчките и доставките**. Чрез използването на **релационна база данни** ще се гарантира структурирано и ефективно съхранение на информацията, а интегрираният потребителски интерфейс ще позволи бърз достъп до всички основни функционалности.

Дигитализацията и автоматизацията на бизнес процесите носят значителни предимства за търговските компании, като подобряват вътрешната организация и повишават конкурентоспособността. Въвеждането на софтуерна система, базирана на релационна база данни, ще даде възможност за **оптимизация на работните процеси, минимизиране на грешките и повишаване на ефективността**. Чрез детайлно анализиране на съществуващите решения и проектиране на функционалностите ще бъде създадена платформа, съобразена със спецификите на фирмата, която ще подобри управлението на ресурсите и ще повиши качеството на предлаганите услуги.

В следващите глави ще бъде направен анализ на съществуващите софтуерни решения, ще се разгледат принципите на релационните бази данни и ще се обоснове изборът на използваните технологии. Освен това ще бъдат представени основните функционалности на разработваното приложение и неговите предимства спрямо алтернативните методи за управление на информацията.

1. **Анализ на съществуващи решения:**

Преди да се пристъпи към проектиране и разработка на собствена система за управление на онлайн магазин за електроника, е от съществено значение да бъдат анализирани съществуващите решения в този домейн. Това включва както проучване на конкурентни платформи и тяхната функционалност, така и използваните от тях технологии и архитектурни подходи. Анализът служи като основа за определяне на силните и слабите страни на наличните решения, за да се избегнат техните недостатъци и да се надградят добрите практики.

**2.1 Изследване на популярни платформи** Онлайн търговията с електроника е добре развита област, в която доминират големи играчи като Amazon, eMAG и Technopolis. Тези платформи предлагат широк набор от функции:

* Интуитивен потребителски интерфейс
* Филтриране и търсене на продукти
* Потребителски профили и история на поръчките
* Сигурно плащане и доставка
* Управление на наличности и категории продукти

Въпреки че тези платформи са мащабни и комплексни, те често са твърде тежки за нуждите на малки и средни бизнеси. Това налага разработката на по-леки и адаптивни решения, съобразени с конкретни изисквания.

**2.2 Технологичен анализ на наличните решения** Много от съществуващите платформи използват популярни среди като PHP и JavaScript за уеб-базирани решения, в комбинация с бази от данни като MySQL, PostgreSQL или MongoDB. В настоящата дипломна работа обаче се прави избор за използване на:

* **C#** за бекенд логиката, поради стабилността и богатата екосистема на .NET.
* **MySQL** за базата от данни, тъй като тя е съвместима, надеждна и лесна за управление с **MySQL Workbench**.
* **Visual Studio** като интегрирана среда за разработка, която предлага богати възможности за дебъгване, тестване и визуален дизайн.

Това решение осигурява по-добра интеграция и гъвкавост при разработка на десктоп базирана система, която не изисква хостинг или специфични сървърни конфигурации.

**2.3 Идентифицирани проблеми и възможности за подобрение** При анализа на съществуващи решения се наблюдават някои често срещани слабости:

* Сложен потребителски интерфейс, който затруднява навигацията
* Ограничена възможност за персонализиране от страна на малкия бизнес
* Слаба интеграция между модулите за поръчки, инвентар и клиенти

Системата, разработена в рамките на дипломната работа, цели да адресира тези проблеми чрез:

* Изчистен и опростен дизайн
* Централизирано управление на данни чрез нормализирана релационна база
* Интуитивна връзка между отделните функционалности (продукти, клиенти, поръчки)

**2.4 Заключение от анализа** Проучването на съществуващи решения е от решаващо значение за дефиниране на обхвата и функционалността на разработваната система. Изборът на конкретни технологии и архитектурни подходи – C#, MySQL, Visual Studio – предоставя стабилна основа за разработка на адаптивно и функционално десктоп приложение, което отговаря на нуждите на електронна търговия в среден и малък мащаб.

1. **Основи на релационните бази данни**

Релационните бази данни са гръбнакът на съвременните информационни системи, осигуряващи надеждно и последователно управление на структурирани данни. Те позволяват ефективно съхранение, извличане и манипулация на информация, като осигуряват висока степен на логическа яснота и поддръжка на сложни зависимости между различни данни. За дипломния проект, фокусиран върху разработката на система за онлайн търговия с електроника, релационният модел предлага стабилна основа за създаване на надеждна и добре организирана база от данни, която е съвместима с инструментите MySQL и езика за програмиране C#.

**3.1 Историческо развитие и концепция** Релационният модел е въведен от Едгар Ф. Код през 1970 г. като теоретичен подход към организацията на данни. Основният му принцип е, че всички данни трябва да бъдат представени във вид на релации (таблици), състоящи се от редове (записи) и колони (атрибути). За разлика от по-старите йерархични и мрежови модели, релационният подход предлага по-голяма гъвкавост и логическа прозрачност, като позволява дефиниране на връзки между обекти от реалния свят по ясен и стандартизиран начин.

**3.2 Основни компоненти на релационния модел**

* **Таблици** – основни структури, които представят конкретни обекти, като продукти, клиенти, поръчки и т.н.
* **Редове (записи)** – всяка таблица съдържа множество редове, като всеки ред представя уникален запис.
* **Колони (атрибути)** – всяка колона представя конкретно свойство на обекта, напр. име, цена, дата.
* **Ключове:**
  + **Първичен ключ (Primary Key)** – уникален идентификатор за всеки ред, който гарантира целостта на данните.
  + **Външен ключ (Foreign Key)** – използва се за дефиниране на връзки между таблици, чрез препращане към първичен ключ от друга таблица.

**3.3 Видове връзки между таблици**

* **Един към един (1:1)** – рядко използвана връзка, напр. един клиент има една потребителска настройка.
* **Един към много (1:N)** – най-често срещаната връзка, напр. един клиент може да направи много поръчки.
* **Много към много (M:N)** – реализира се чрез допълнителна (свързваща) таблица, напр. поръчки и продукти.

**3.4 Нормализация на бази от данни** Нормализацията е процес на разделяне на големи таблици на по-малки и логически свързани таблици, с цел избягване на дублирани данни и аномалии при актуализации. Този процес преминава през няколко нормални форми:

* **1NF (Първа нормална форма)** – елиминиране на многозначни колони.
* **2NF (Втора нормална форма)** – премахване на частични зависимости.
* **3NF (Трета нормална форма)** – елиминиране на транзитивни зависимости. Нормализираната база е по-ефективна, по-надеждна и по-лесна за поддръжка.

**3.5 SQL – език за работа с релационни бази** SQL (Structured Query Language) е основен език за комуникация с релационни бази данни. Неговите основни команди включват:

* **SELECT** – извличане на данни
* **INSERT** – добавяне на записи
* **UPDATE** – промяна на съществуващи записи
* **DELETE** – изтриване на данни
* **CREATE, ALTER, DROP** – управление на структурата на базата Тези команди се използват активно чрез C# кода и ADO.NET компонентите в настоящия проект.

**3.6 Използване на MySQL и C# в проекта** Настоящата дипломна работа използва MySQL като система за управление на бази от данни, поради нейната надеждност, безплатен лиценз и добра поддръжка на SQL стандарта. C# и ADO.NET са използвани за осъществяване на връзката между приложението и базата, позволявайки динамично извличане, редактиране и визуализация на данни.

* **MySQL Workbench** се използва за визуално моделиране на базата, както и за ръчно тестване на заявки.
* **Visual Studio** и **C#** осигуряват потребителския интерфейс и логиката на приложението.

Използването на тези технологии осигурява стабилна и гъвкава архитектура, която позволява лесно разширяване и адаптиране към бъдещи бизнес нужди.

1. **Технологии за разработка**

Изборът на технологии за реализирането на софтуерен проект е от ключово значение за неговата стабилност, ефективност и поддръжка. При настоящото дипломно приложение, чиято цел е изграждането на система за управление на онлайн магазин за електроника, са използвани утвърдени и широко разпространени инструменти и програмни езици – **C#**, **MySQL**, **Visual Studio** и **MySQL Workbench**. Тези технологии предлагат богата екосистема от библиотеки, документация и възможности за интеграция, което ги прави идеални за създаването на пълноценна информационна система.

**4.1 C# като език за програмиране -** C# (C Sharp) е обектно-ориентиран език, разработен от Microsoft като част от .NET платформата. Той съчетава мощта и производителността на C++ със сигурността и управлението на паметта на Java. Сред основните предимства на C# за разработка на настолни приложения са:

* Поддръжка на богати графични интерфейси чрез Windows Forms или WPF
* Лесна интеграция с бази от данни чрез ADO.NET
* Добра типова безопасност и автоматично управление на паметта
* Развита среда за дебъгване и писане на код във Visual Studio

**4.2 Visual Studio** Visual Studio - е интегрирана среда за разработка (IDE), която предоставя широк набор от инструменти за писане, тестване и дебъгване на приложения, разработени на C#. Нейните предимства включват:

* Поддръжка на шаблони за проекти и форми
* Интегриран дизайнер за графичен интерфейс
* Инструменти за контрол на версиите (Git)
* Възможност за разширяване с допълнителни разширения и плъгини

**4.3 ADO.NET комуникация с база от данни -** ADO.NET е компонент на .NET Framework, който предоставя набор от библиотеки за работа с бази от данни. С негова помощ могат да се изпълняват SQL заявки, да се четат и модифицират данни, както и да се обработват резултати чрез DataTable и DataSet. Основни характеристики:

* Работа със свързан и несвързан модел
* Поддръжка на параметризирани заявки за сигурност срещу SQL инжекции
* Лесна интеграция с MySQL чрез MySQL Connector/NET

**4.4 MySQL като релационна база данни -** MySQL е един от най-популярните системи за управление на релационни бази от данни с отворен код. Тя се използва широко за уеб и десктоп приложения поради своята:

* Висока производителност при големи обеми от данни
* Поддръжка на транзакции, индекси, тригери и процедури
* Съвместимост с .NET приложения чрез съответни драйвери
* Добре документиран синтаксис и поддръжка на стандартен SQL

**4.5 MySQL Workbench -** MySQL Workbench е визуален инструмент за работа с MySQL бази данни. Използва се за:

* Създаване и моделиране на таблици, релации и ограничения
* Изпълнение на SQL заявки
* Преглед и модификация на данни
* Създаване на ER диаграми за визуализиране на структурата

**4.6 Взаимодействие между компонентите -** В системата, реализирана по време на проекта, взаимодействието между компонентите е реализирано по следния начин:

* Потребителят работи с графичен интерфейс, създаден с Windows Forms в средата на Visual Studio
* Чрез C# и ADO.NET се изпращат заявки към базата данни
* Базата от данни, създадена и управлявана в MySQL Workbench, отговаря на заявките и връща резултатите
* Резултатите се визуализират обратно в потребителския интерфейс

Тази архитектура осигурява ясно разделение между представянето, логиката и данните, като същевременно гарантира мащабируемост и възможност за бъдещо надграждане. Използваните технологии са стабилни, добре документирани и с широка поддръжка в професионалната общност, което допринася за успеха на проекта.

1. **Методология на разработка**

Процесът на разработка на информационната система за управление на онлайн магазин за електроника следва ясно структуриран подход, основан на класическия водопаден модел (Waterfall Model). Изборът на този модел се обуславя от ясните и предварително дефинирани изисквания на проекта, както и от необходимостта за последователно и детайлно изпълнение на всяка фаза от разработката. Всички дейности са извършени в рамките на средата за разработка **Visual Studio** с език за програмиране **C#** и използване на **MySQL Workbench** за проектиране и управление на базата данни.

Водопадният модел осигурява линейна и контролирана среда за разработка, при която всяка стъпка логически следва предходната. Това улеснява проследимостта, управлението на риска и поддръжката на проекта, като се минимизира възможността за неясноти и пропуски в изискванията. Макар и по-малко гъвкав от гъвкавите методологии (Agile), водопадният подход предоставя стабилност, особено подходяща при академични разработки с фиксирани параметри.

5.1 Анализ на изискванията

Този етап включва събиране и структуриране на информация за нуждите на системата, като се вземат предвид основните функционалности и очакванията на потребителите. Определени са ключовите участници в системата – администратори, служители и клиенти. Формулирани са изисквания за сигурност, ефективност, бързодействие и лесна поддръжка. Извършен е анализ на конкуренцията и типичните процеси в реални електронни магазини, както и сравнителна оценка на съществуващи платформи и техните слаби страни. Това позволява формиране на реалистични цели и обосновани решения за бъдещия софтуер.

5.2 Проектиране на системата

На този етап е създадена архитектурата на софтуерното приложение, базирана на три основни слоя – презентационен (потребителски интерфейс), логически (бизнес логика) и слой за достъп до данни (взаимодействие с MySQL базата). Създаден е ER-модел на базата данни с помощта на MySQL Workbench, определени са релации между таблиците и са зададени ключови зависимости. Специално внимание е отделено на нормализацията на данните, за да се избегне излишно дублиране и да се гарантира консистентност. Интерфейсът на потребителя е проектиран така, че да бъде интуитивен и достъпен, с внимание към UX принципи и удобство при навигация.

5.3 Имплементация

Разработката на системата се извършва в средата на Visual Studio, като се използва **C#** за логиката на приложението и **Windows Forms** за изграждане на графичния потребителски интерфейс. Изградена е връзка с базата данни чрез **MySQL Connector**, реализирани са CRUD операции чрез параметризирани SQL заявки, което осигурява надеждна защита срещу SQL инжекции. Всяка функционалност е имплементирана поетапно, с последователно добавяне и тестване на модули. Използвани са обектно-ориентирани принципи на програмиране, което улеснява разширяемостта и повторната употреба на код. Въведени са логически проверки и валидации на данните преди запис в базата, както и логика за обработка на изключения.

5.4 Тестване

Функционалното тестване е ключова част от методологията. Извършено е ръчно тестване на всеки отделен модул – вход в системата, работа с потребители, продукти, поръчки и отчети. Проверена е коректността на изчисленията, надеждността на валидациите, и съвместимостта на интерфейса. Осъществено е тестване на връзката с базата данни, ефективността на заявките и реакцията при погрешни входни данни. Бяха отстранени всички възникнали грешки, а логическите зависимости между таблиците в базата данни бяха потвърдени чрез симулирани потребителски действия и екстремни сценарии. Периодично е правен преглед на кода с цел откриване на скрити проблеми и потенциални уязвимости.

5.5 Документиране и представяне

След успешната реализация на всички функционалности е създадена пълна техническа документация, включваща описание на архитектурата, структурата на базата данни, интерфейса и използваните технологии. Документацията е структурирана така, че да бъде достъпна както за крайни потребители, така и за бъдещи разработчици. Направена е демонстрация на проекта и подготвено ръководство за работа със системата. Приложена е и подробна секция с често задавани въпроси (FAQ) и инструкции за отстраняване на основни проблеми.

Използваната методология позволява структурирана и надеждна реализация на проекта, с висока степен на контрол върху всеки етап от процеса и възможност за бъдещи разширения и оптимизации. Тя осигурява прозрачност, последователност и предвидимост, което е от съществено значение за успешното реализиране на софтуерно решение с практическа насоченост.

1. **Архитектура на системата**

Функционалните характеристики представляват основата на всяко софтуерно приложение. Те дефинират какво ще може да прави системата, как ще комуникира с потребителя и какви операции ще се извършват върху данните. Разработеното в рамките на тази дипломна работа приложение за управление на онлайн магазин за електроника е ориентирано към улесняване на процесите по продажба, инвентаризация и обслужване на клиенти. Системата е разработена с помощта на **C#** в **Visual Studio** и използва **MySQL** база от данни, управлявана чрез **MySQL Workbench**. Основната ѝ цел е да предостави цялостно решение за управление на продукти, поръчки, клиенти и административни дейности, като в същото време поддържа високо ниво на сигурност и ефективност.

**6.1 Основни функционалности**

* **Управление на продукти**: Приложението позволява добавяне, редактиране и изтриване на продукти. Всеки продукт съдържа информация за име, описание, цена, наличност, категория и тип (настолен компютър, лаптоп или сървър). Категоризацията помага на потребителите да навигират по-лесно из продуктовия каталог, а интегрираният механизъм за филтриране и търсене осигурява бърз достъп до желаните артикули.
* **Управление на клиенти**: В системата се поддържа детайлна база данни с клиентска информация, включително пълно име, телефонен номер, имейл адрес и физически адрес. Това позволява персонализирано обслужване и по-добро планиране на маркетингови стратегии.
* **Създаване и управление на поръчки**: Приложението предоставя възможност за създаване на поръчки чрез лесен и интуитивен интерфейс. Системата генерира уникален номер на поръчката, отбелязва дата и час на извършване, и съхранява пълна информация за транзакцията. Статусите на поръчките (на изчакване, обработка, завършена, отказана) се управляват динамично.
* **Филтриране и търсене**: Реализиран е механизъм за бързо и ефективно намиране на продукти и клиенти по различни критерии: име, ID, категория, цена, наличност. Това значително улеснява потребителя при работа с големи обеми от данни и подобрява цялостната потребителска ефективност.

**6.2 Административен панел** Системата разполага със защитен административен панел, предназначен за използване от персонал с високо ниво на достъп. Функционалностите включват:

* Достъп до пълната база данни и всички таблици
* Възможност за изпълнение на CRUD операции (Create, Read, Update, Delete) върху всички обекти
* Управление на категории и наличности

**6.3 Потребителски интерфейс** Потребителският интерфейс е реализиран с помощта на **Windows Forms** в средата на Visual Studio. Спазени са основни принципи на UX/UI дизайн:

* **Интуитивност**: Интерфейсът е логически структуриран, така че дори потребители без задълбочени технически познания да могат лесно да навигират в системата.
* **Удобство**: Формите са минималистични и функционални, с добре подредени контроли, осигуряващи бърз достъп до всички основни действия.
* **Скорост и производителност**: Използваните елементи са леки и оптимизирани за бързо зареждане и реакция, което прави приложението подходящо дори за по-стари машини.
* **Адаптивност**: Интерфейсът е разработен така, че да може лесно да бъде модифициран в бъдещи версии, в случай че се премине към уеб-базирана реализация.

**6.4 Бъдещи подобрения и възможности за разширяване** Предвид динамичното развитие на електронната търговия и нуждите на бизнеса, системата е проектирана така, че да може да бъде разширявана. Някои от възможните бъдещи подобрения включват:

* **Интеграция с онлайн разплащателни системи** – за осигуряване на автоматизирани разплащания чрез PayPal, кредитни/дебитни карти и др.
* **Журнал на действията** – история на административни дейности за целите на проследимост и сигурност
* **Преглед на системна статистика в реално време** (общ брой поръчки, среден приход, най-продавани артикули)
* **Изпращане на известия и напомняния** – чрез имейл или SMS при нова поръчка, промоции или нужда от доставка
* **Система за лоялни клиенти** – начисляване на точки при всяка покупка, които могат да се обменят за отстъпки или подаръци
* **Миграция към уеб базирана платформа** – използване на ASP.NET и Blazor за създаване на уеб версия с достъп от всяко устройство
* **Интеграция с мобилно приложение** – за мобилен достъп до най-важните функции на системата от страна на администратори и клиенти
* **Аналитични модули с графики и визуализации** – за по-ефективен бизнес анализ и прогнозиране
* **Справки и отчети** – генерирането на отчети, която подпомага мениджърския процес

Разработената функционалност поставя стабилна основа за създаване на пълноценна и професионална система за управление на онлайн магазин за електроника, която е в състояние да се развива и адаптира спрямо бъдещите нужди и технологични тенденции.

1. **Функционални характеристики на приложението**

Архитектурата на разработената система играе ключова роля в осигуряването на стабилна, разширяема и сигурна платформа за електронна търговия. Избраната архитектурна структура следва принципите на слоест дизайн (multi-layered architecture), при който логиката на приложението е разделена на отделни слоеве, всеки със свои отговорности. Това позволява лесна поддръжка, подобрения и повторна употреба на кода.

Системата се състои от следните основни слоеве:

**7.1 Презентационен слой (Presentation Layer)** Този слой отговаря за взаимодействието с крайния потребител. Реализиран е чрез **Windows Forms** в **Visual Studio** с помощта на **C#**. Интерфейсът е съобразен с основните принципи на удобство за потребителя (UX) и включва:

* Форма за вход с валидация на потребителски имена и пароли
* Форми за CRUD операции (създаване, редактиране, изтриване и преглед)
* Динамично зареждане на данни от базата чрез DataGridView компоненти
* Потвърждения за действия и съобщения за грешки

**7.2 Логически слой (Business Logic Layer)** Този слой реализира бизнес логиката на приложението, като обработва данните, получени от потребителския интерфейс, и ги подготвя за съхранение в базата. Отговорностите му включват:

* Проверка на валидност на въведени данни
* Изчисление на обща стойност на поръчките
* Генериране на уникални идентификатори
* Управление на статуса на поръчки
* Комуникация с логиката за отчетност и филтриране

**7.3 Данни и достъп до база данни (Data Access Layer)** Този слой осъществява връзката с **MySQL** базата данни чрез използване на **MySql.Data** библиотеката. Всички SQL заявки и операции с базата преминават през този слой. Основните функции включват:

* Изпълнение на заявки за извличане и обновяване на информация (SELECT, INSERT, UPDATE, DELETE)
* Свързване към базата чрез стрингове за връзка (connection strings)
* Работа с параметризирани заявки с цел предотвратяване на SQL инжекции
* Прехвърляне на данни към логическия слой чрез обекти и структури

**7.4 База данни (Database Layer)** Базата данни е създадена и поддържана с помощта на **MySQL Workbench** и се състои от няколко взаимносвързани таблици:

* **Customers** – съдържа информация за клиентите (име, имейл, адрес, телефон)
* **Products** – информация за продуктите (име, цена, количество, категория, производител)
* **Orders** – информация за поръчките (дата, клиент, количество, състояние, продукт)
* **Suppliers** – информация за доставчиците (име, имейл, телефон)
* Всяка таблица е свързана чрез външни ключове (foreign keys), което осигурява цялостта на данните и логическите зависимости между обектите

**7.5 Възможности за разширяване** Архитектурата е гъвкава и предразполага към бъдещи надграждания, като:

* Миграция към уеб-базирано приложение чрез **ASP.NET Core MVC**
* Добавяне на REST API за свързване с външни системи
* Включване на допълнителни модули – плащания, известия, мобилни приложения
* Използване на нива на достъп – администраторите имат пълен контрол, а потребителите – ограничени възможности
* Защита на вход чрез хеширане на пароли (при евентуална бъдеща реализация)

Тази архитектура гарантира стабилност, мащабируемост и ясна организация на кода. Тя позволява лесно откриване и отстраняване на грешки, както и бързо адаптиране към нови изисквания или промени в бизнес логиката на приложението.

1. **Заключение**

Разработката на системата за управление на електронен магазин за техника представлява успешно реализиране на съвременен софтуерен проект, който интегрира ключови принципи на обектно-ориентираното програмиране, структурирано моделиране на бази данни и съвременни технологии за разработка. Чрез използването на **Visual Studio** и езика **C#** в комбинация с **MySQL Workbench** и **MySQL** базата данни, бе създадено надеждно и функционално приложение, отговарящо на реални бизнес нужди.

Системата обхваща всички основни аспекти на управление на онлайн магазин – от регистриране и управление на клиенти, през организиране и поддръжка на продуктова гама, до извършване на поръчки и проследяване на техния статус. Изграденият потребителски интерфейс е ясен и достъпен, като осигурява високо ниво на използваемост, а архитектурата на системата гарантира възможности за лесна поддръжка, развитие и адаптация към бъдещи изисквания.

По време на разработката бяха приложени редица добри практики като разделяне на логическите отговорности чрез слоест модел, използване на параметризирани заявки за сигурност срещу SQL инжекции, и модулно структуриране на функционалностите. Това не само повиши качеството на крайния продукт, но и допринесе за професионално развитие в работата с реални софтуерни инструменти и технологии.

Системата разполага с потенциал за надграждане чрез добавяне на модули за електронни плащания, интеграция с външни услуги чрез API, мобилна версия на приложението и дори преминаване към облачна архитектура. По този начин настоящата дипломна работа поставя здрава основа, върху която могат да се изграждат по-сложни решения, отговарящи на нарастващите нужди на съвременния дигитален бизнес.

Разработката демонстрира как с помощта на достъпни, но мощни технологии като **C#** и **MySQL**, може да бъде създадено професионално софтуерно решение, което ефективно автоматизира и оптимизира процесите в един реален бизнес контекст.

**Раздел II – Практическа част**

1. Описание на програмата

Разработеното приложение представлява **десктоп софтуер**, реализиран с помощта на **C# и Windows Forms**, който комуникира с **релационна база данни** (MySQL или SQL Server). Целта на приложението е да предостави удобен графичен интерфейс за извършване на CRUD операции (създаване, четене, редактиране и изтриване) върху четири основни таблици: products, customers, orders и suppliers.

Програмата включва два основни прозореца:

* Логин форма – вход в системата чрез потребителско име и парола.
* Основен прозорец за обработка на данни – с достъп до всички таблици и функции.

1. Вход в системата (Login)

При стартиране на програмата се зарежда логин форма, която изисква въвеждане на **потребителско име и парола**. Интерфейсът включва:

* Икони за потребител и парола.
* Функция за скриване/показване на въведената парола.
* Бутон „Clear Fields“ за изчистване на въведените данни.
* Бутон „Exit“ за изход от програмата.

При валидни данни, приложението отваря основния интерфейс за обработка на данни.

1. Работа с данните

След успешен вход, се зарежда **главният прозорец** на приложението – ProcessingForm.

3.1 Избор на таблица

Потребителят може да избере с коя таблица да работи от падащо меню „Таблица“. Възможните опции са:

* products – Продукти
* customers – Клиенти
* orders – Поръчки
* suppliers – Доставчици

След избора, съответната форма се визуализира вдясно, а данните се зареждат в **таблица (DataGridView)** вляво.

3.2 CRUD функционалност

За всяка таблица са налични бутони за следните действия:

* **Намери** – търсене по ID (първичен ключ).
* **Записване** – добавяне на нов запис към базата.
* **Редактиране** – промяна на съществуващ запис.
* **Изтриване** – премахване на запис.

Пример: Таблица products

Потребителят попълва:

* Name (Име на продукта)
* Category (категория – избира се от падащо меню)
* Price (цена)
* QuantityInStock (налично количество)
* SupplierID (доставчик)

След това може да натисне „Записване“, за да добави нов продукт.

1. Работа с базата данни

Връзката с базата данни се реализира чрез SqlConnection (или MySqlConnection). Всеки бутон изпълнява SQL команда – INSERT, SELECT, UPDATE или DELETE чрез SqlCommand.

**Примерен код за добавяне на клиент:**