**ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**«ЗАВОЛЖСКИЙ АВТОМОТОРНЫЙ ТЕХНИКУМ»**

Специальность 09.02.07 «Информационные системы и программирование»

**Курсовая работа**

**Пояснительная записка**

По дисциплине **Технология разработки и защиты баз данных**

Тема: Разработка автоматизированной информационной системы по учёту проданных товаров пекарни «Круглова» на базе платформы .NET.

Число листов курсового проекта (прописью)

Число страниц пояснительной записки

Студент Кожевников Сергей Романович Группа ИС-21Б

Руководитель Иванов Илья Александрович

Дата защиты Оценка

2024 - 2025 учебный год

Содержание

Введение 3

1. Проектирование 5
   1. Обоснование выбора среды разработки 5
   2. Выбор способа организации входных и выходных данных 8
   3. Проектирование моделей данных 13
2. Алгоритм представления задачи 17
   1. Детальное проектирование программного обеспечения 17
   2. Описание основных программных модулей 21
   3. Применяемые меры защиты информации 22

Заключение 24

Литература 27

**Введение**

Пекарня – небольшое предприятие по выпечке и реализации хлебобулочных и кондитерских изделий, как правило, также реализующее их на месте.

Продажи выпечки почти не зависят от сезона, продукция пользуется стабильным спросом круглый год.

Выпечка – это лучшее лекарство от стресса. Как показывают данные аналитических обзоров, в кризис прибавляют в продажах сладости и кондитерские изделия до 20–30%

Пекарня «Круглова» имеет более 10 филиалов, каждый из которых осуществляет продажу различных категорий выпечки, кондитерских изделий и других товаров. Для эффективной торговли требуется инструмент для оперативного оформления заказов клиентов, ведения информации об имеющихся товарах и анализа информации о продажах.

Автоматизированная информационная система - программно-аппаратный комплекс для авторизации деятельности организации, который обеспечивает хранение, передачу и обработку информации.

Автоматизированные информационные системы по ведению базы данных широко применяются в современной торговле. Такие системы упрощают работу сотрудников, а также повышают эффективность поиска и обработки информации.

Целью курсового проекта является создание автоматизированной информационной системы по учёту проданных товаров пекарни «Круглова» на базе платформы .NET.

В процессе проектирования нужно решить следующие задачи:

* провести анализ существующих систем;
* сформулировать требования к программе;
* построить структуру базы данных;
* выбрать среду разработки;
* реализовать программный продукт;
* протестировать программный продукт;
* разработать техническую и эксплуатационную документацию

Программа может использоваться для учёта товаров и формирования заказов.

1. **Постановка задачи**

Разработать и программно реализовать автоматизированную информационную систему (АИС) по учету проданных товаров пекарни «Круглова». В программе должны соблюдаться принципы информационной закрытости, модульности и иерархической организации.

Предметная область: системы пекарни «Круглова» содержит необходимую информацию о продукции и расценках пекарни.

АИС по учету проданных товаров пекарни «Круглова» предназначена для ведения базы данных, оформления и получения заказов пекарни.

Автоматизированная информационная система по учету проданных товаров пекарни «Круглова» должна иметь форму авторизации (администратор, менеджер, сотрудник), в режиме администратора должна быть предусмотрена возможность создания нового пользователя.

Автоматизированная информационная система пекарни должна иметь возможность ведения и редактирования базы данных (добавление, удаление, редактирование, сохранение).

Программа позволяет оформить и получить заказ, принять оплату и распечатать чек об оплате (в MS Excel), возможность получить скидку (на определенный товар), ведет учет остатков продуктов.

Пользователь должен иметь возможность отсортировать информацию о товарах (по возрастанию и убыванию) по названию или по стоимости. Возможность отфильтровать данные. Реализовать живой поиск информации.

Предусмотреть возможность составления отчетов (в MS Exсel).

Автоматизированная информационная система пекарни должна иметь возможность импорта и экспорта данных базы.

Язык программирования С#. База данных составляется в MySQL.

1. **Проектирование**
   1. **Обоснование выбора среды разработки**

Для создания данного программного продукта можно использовать следующие среды разработки:

* Visual Studio 2022 Community - это интегрированная среда разработки (IDE), разработанная компанией Microsoft. Она используется для разработки компьютерных программ. Встроенные языки включают C, C++, C#. Visual Studio использует платформы разработки программного обеспечения Microsoft, включая Windows Forms и Windows Presentation Foundation (WPF). Visual Studio включает редактор кода, поддерживающий IntelliSense (компонент автоматического завершения кода), а также рефакторинг кода. Visual Studio включает отладчик, который работает как на уровне исходного кода, так и на уровне машинного кода. Он работает как с управляемым кодом, так и с собственным кодом и может использоваться для отладки приложений, написанных на любом языке, поддерживаемом Visual Studio. Кроме того, он может подключаться к запущенным процессам, отслеживать и отлаживать их. Отладчик Visual Studio позволяет устанавливать точки останова (которые позволяют временно приостановить выполнение в определённой точке). Visual Studio включает в себя множество визуальных конструкторов, помогающих в разработке приложений. К этим инструментам относятся конструктор Windows Forms и конструктор WPF;
* Rider– кроссплатформенная интегрированная среда разработки программного обеспечения для платформы .NET, разрабатываемая компанией JetBrains. Rider поддерживает множество языков .NET-разработки, включая C#. Rider поддерживает модульные тесты в приложениях.NET Framework,.NET Core и Mono. Rider предоставляет интеграцию с Git. В специальном окне инструментов можно отслеживать локальные и внешние изменения в репозитории, управлять списками изменений, откладывать изменения, сравнивать файлы, делать коммиты и вливать локальные изменения в удаленный репозиторий. Встроенный инструмент визуального слияния позволяет с легкостью разрешать конфликты. Кроме того, Rider предупредит, если файлы были изменены после последней синхронизации;
* MonoDevelop - это интегрированная среда разработки с открытым исходным кодом, созданная для языков .NET. Она использует Mono и .NET Framework для разработки программного обеспечения. С помощью MonoDevelop можно эффективно создавать настольные приложения для различных операционных систем.

Исходя из вышеперечисленных характеристик сред разработки выбираем Visual Studio 2022 Community, т.к. данная интегрированная среда разработки имеет удобный отладчик и визуальный конструктор для разработки дизайна приложения.

В состав выбранной интегрированной среды разработки входят следующие языки программирования:

* C# - высокоуровневый объектно-ориентированный язык программирования общего назначения. Разработан как язык разработки приложений для платформы.NET Framework и .NET Core. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов, делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, исключения, комментарии в формате XML.C# предоставляет удобные классы для построения графических интерфейсов по технологии Windows Forms;
* C++ - компилируемый, статически типизированный язык программирования общего назначения. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков.
* C - компилируемый, статически типизированный язык программирования общего назначения. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C не является объектно-ориентированным языком программирования. Основные особенности Си:
* простая языковая база, из которой в стандартную библиотеку вынесены многие существенные возможности, вроде математических функций или функций работы с файлами;
* ориентация на процедурное программирование;
* система типов, предохраняющая от бессмысленных операций;
* использование препроцессора для абстрагирования однотипных операций;
* доступ к памяти через использование указателей;
* небольшое число ключевых слов;
* наличие указателей на функции и статические переменные;
* области видимости имён;
* структуры и объединения - определяемые пользователем собирательные типы данных, которыми можно манипулировать как одним целым.

В то же время в C отсутствуют:

* вложенные функции;
* прямое возвращение нескольких значений из функций;
* средства автоматического управления памятью;
* встроенные средства объектно-ориентированного программирования;
* средства функционального программирования.

Исходя из вышеперечисленных характеристик языков программирования выбираем C#, т.к. данный язык программирования изначально разработан как язык разработки приложений для платформы .NET, а также предоставляет удобные классы для построения графических интерфейсов по технологии Windows Forms.

* 1. **Выбор способа организации входных и выходных данных**

Для организации входных и выходных данных необходима база данных

* MySql - свободная реляционная СУБД. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты. MySQL имеет API и коннекторы для языков C, C++, библиотеки для языков платформы.NET;
* SQLite- компактная встраиваемая СУБД. Представляет собой библиотеку, с которой программа компонуется, и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа;
* MongoDB – документоориентированная система управления базами данных, не требующая описания схемы таблиц. Считается одним из классических примеров NoSQL-систем, использует JSON-подобные документы и схему базы данных. Система поддерживает ad-hoc-запросы: они могут возвращать конкретные поля документов и пользовательские JavaScript-функции. Поддерживается поиск по регулярным выражениям. Также можно настроить запрос на возвращение случайного набора результатов. Имеется поддержка индексов.

Исходя из вышеперечисленных характеристик баз данных выбираем MySql, т.к. данная СУБД является реляционной и работает по принципу «Клиент-сервер», что позволяет организовать централизованное хранилище данных.

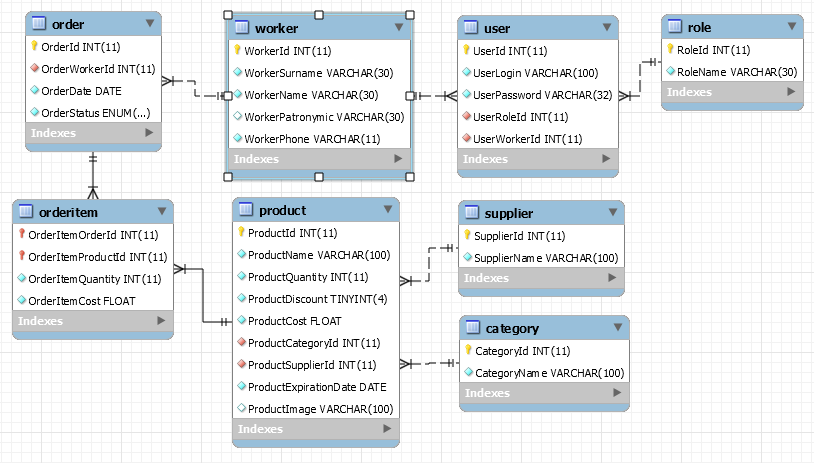


Рисунок 1.2.1 – ER-диаграмма

База данных имеет 8 таблиц, каждая имеет связи в соответствии с рисунком 1.2.1

Структура таблиц базы данных представлена таблицами 1.2.1-1.2.9

Таблица 1.2.1 - Структура таблицы продуктов

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Название столбца в БД | Тип данных | Ограничения |
| Идентификатор продукта | ProductId | INT | Обязательное, уникальное |
| Название продукта | ProductName | VARCHAR | Обязательное, до 100 символов |
| Количество продукта | ProductQuantity | INT | Обязательное |
| Скидка на продукт | ProductDiscount | TINYINT | Обязательное |
| Стоимость продукта | ProductCost | FLOAT | Обязательное, точность 2 |
| Идентификатор категории продукта | ProductCategoryId | INT | Обязательное, внешний ключ поля CategoryId |
| Идентификатор поставщика продукта | ProductSupplierId | INT | Обязательное, внешний ключ поля SupplierId |
| Срок годности продукта | ProductExpirationDate | DATE | Обязательное |
| Название изображения продукта | ProductImage | VARCHAR | До 100 символов |

Таблица 1.2.2 - Структура таблицы пользователей

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Название столбца в БД | Тип данных | Ограничения |
| Идентификатор пользователя | UserId | INT | Обязательное, уникальное |
| Логин пользователя | UserLogin | VARCHAR | Обязательное, до 100 символов |
| Hash пароля пользователя | UserPassword | VARCHAR | Обязательное, до 32 символов |
| Идентификатор роли пользователя | UserRoleId | INT | Обязательное, внешний ключ поля RoleId |
| Идентификатор работника, относящегося к пользователю | UserWorkerId | INT | Обязательное, внешний ключ поля WorkerId |

Таблица1.2.3 - Структура таблицы работников

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Название столбца в БД | Тип данных | Ограничения |
| Идентификатор работника | WorkerId | INT | Обязательное, уникальное |
| Фамилия работника | WorkerSurname | VARCHAR | Обязательное, до 30 символов |
| Имя работника | WorkerName | VARCHAR | Обязательное, до 30 символов |
| Отчество работника | WorkerPatronymic | VARCHAR | До 30 символов |
| Номер работника | WorkerPhone | VARCHAR | Обязательное, 11 символов |

Таблица1.2.4 - Структура таблицы позиций заказов

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Название столбца в БД | Тип данных | Ограничения |
| Идентификатор заказа | OrderItemOrderId | INT | Обязательное, уникальное при не уникальном OrderItemProductId, внешний ключ поля OrderId |
| Идентификатор продукта | OrderItemProductId | INT | Обязательное, уникальное при не уникальном OrderItemOrderId, внешний ключ поля ProductId |
| Количество продукта | OrderItemQuantity | INT | Обязательное |
| Стоимость продукта | OrderItemCost | FLOAT | Обязательное |

Таблица1.2.5 - Структура таблицы заказов

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Название столбца в БД | Тип данных | Ограничения |
| Идентификатор заказа | OrderId | INT | Обязательное, уникальное |
| Идентификатор пользователя сформировавшего заказ | OrderUserId | INT | Обязательное, внешний ключ поля UserId |
| Дата формирования заказа | OrderDate | DATE | Обязательное |
| Статус заказа | OrderStatus | ENUM(«Новый», «Проведён») | Обязательное |

Таблица1.2.6 - Структура таблицы категорий

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Название столбца в БД | Тип данных | Ограничения |
| Идентификатор категории | CategoryId | INT | Обязательное, уникальное |
| Наименование категории | CategoryName | VARCHAR | Обязательное, до 100 символов |

Таблица1.2.7 - Структура таблицы ролей

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Название столбца в БД | Тип данных | Ограничения |
| Идентификатор роли | RoleId | INT | Обязательное, уникальное |
| Наименование роли | RoleName | VARCHAR | Обязательное, до 30 символов |

Таблица 1.2.8 - Структура таблицы поставщиков

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Название столбца в БД | Тип данных | Ограничения |
| Идентификатор поставщика | SupplierId | INT | Обязательное, уникальное |
| Наименование поставщика | SupplierName | VARCHAR | Обязательное, до 100 символов |

Таблица1.2.9 - Структура таблицы клиентов

|  |  |  |  |
| --- | --- | --- | --- |
| Название столбца | Название столбца в БД | Тип данных | Ограничения |
| Идентификатор клиента | WorkerId | INT | Обязательное, уникальное |
| Фамилия клиента | WorkerSurname | VARCHAR | Обязательное, до 30 символов |
| Имя клиента | WorkerName | VARCHAR | Обязательное, до 30 символов |
| Отчество клиента | WorkerPatronymic | VARCHAR | До 30 символов |
| Номер клиента | WorkerPhone | VARCHAR | Обязательное, 11 символов |

* 1. **Проектирование моделей данных**

При проектировании моделей данных были построены диаграмма вариантов использования и диаграмма деятельности.

Диаграмма вариантов использования - диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей

Диаграмма вариантов использования представлена в соответствии с рисунком 1.3.1.

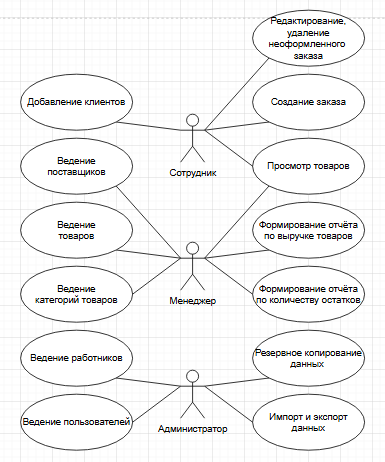


Рисунок 1.3.1 – Диаграмма вариантов использования

Сценарий диаграммы вариантов использования - специально написанный текст, который описывает поведение моделируемой системы в форме последовательности выполняемых действий актеров и самой системы.

Сценарий диаграммы вариантов использования представлен в соответствии с таблицей 1.3.1.

Таблица 1.3.1 - Варианты использования

|  |  |
| --- | --- |
| Варианты использования | Создание заказа |
| Актёры | Сотрудник, менеджер, администратор |
| Краткое описание | Сотрудник просматривает товары, сотрудник формирует заказы, сотрудник ведёт информацию о клиентах менеджер ведёт информацию о товарах, поставщиках и категориях, менеджер формирует отчёты по количеству остатков, менеджер формирует отчёты по выручке товаров, администратор ведёт информацию о работниках и пользователях приложения, администратор выполняет резервное копирование данных, администратор выполняет импорт и экспорт данных |
| Цель | Продажа товаров |
| Тип | Базовый |
| Ссылки на другие варианты использования | Просмотр товаров, ведение информации о товарах, ведение информации о поставщиках, ведение информации о категориях, ведение информации о клиентах, ведение информации о работниках, ведение информации о пользователях, формирование отчёта по количеству остатков, формирование отчёта по выручке товаров, резервное копирование данных, импорт и экспорт данных |

Последовательность действий - это список действий, которые выполняются одно за другим. Вся последовательность считается одним атомарным действием, поэтому ее нельзя прервать извне, но можно присоединять к переходам или шагам при взаимодействии.

Последовательность действий представлена в соответствии с таблицей 1.3.2.

Таблица 1.3.2 – Последовательность действий

|  |  |
| --- | --- |
| Действия актёров | Отклик системы |
| 1. Сотрудник авторизуется в системе  Искл. 1. Введён неверный логин или пароль | 2. Открывается главное меню в соответствии с ролью сотрудника  3. Очистка полей «Логин» и «Пароль», вывод сообщения об ошибке ввода |
| 4. Сотрудник открывает список товаров | 5. Отображается информация о товарах |
| 6. Сотрудник выбирает товар из списка и добавляет к заказу | 7. Создаётся заказ, отображается кнопка для перехода к заказу |
| 8. Сотрудник открывает заказ | 9. Отображаются товары, добавленные в заказ |
| 10. Сотрудник формирует заказ | 11. Данные о заказе сохраняются в базе данных, отображается сообщение о создании заказа |
| 12. Менеджер открывает список заказов | 13. Отображается информация о заказах |
| 14. Менеджер открывает заказ | 15. Отображаются товары в заказе и информация о заказе |
| 16. Менеджер формирует отчёт по выручке товаров | 17. Открывается отчёт по выручке товаров в Excel |
| 18. Менеджер редактирует заказ  Искл. 2. Введена некорректная информация | 19. Изменения о заказе сохраняются в базе данных, отображается сообщение об успешном редактировании  20. Отображение ошибки редактирования |
| 21. Менеджер открывает список актов приёма | 22. Отображается информация об актах приёма |
| 23. Менеджер открывает список товаров | 24. Отображается информация о товарах |
| 25. Менеджер формирует отчёт по количеству остатков | 26. Открывается отчёт по количеству остатков в Excel |
| 27. Администратор выбирает таблицу | 28. Отображается содержимое таблицы |
| 29. Администратор добавляет запись | 30. Запись добавляется в базу данных, отображается сообщение об успешном добавлении |

Продолжение таблицы 1.3.2

|  |  |
| --- | --- |
| Действия актёров | Отклик системы |
| 31. Администратор редактирует запись | 32. Запись изменяется в базе данных, отображается сообщение об успешном редактировании |
|  |  |
| 33. Администратор удаляет запись | 34. Отображается сообщение о подтверждении удаления |
| 35. Администратор подтверждает удаление записи | 36. Запись удаляется из базы данных, отображается сообщение об успешном удалении |

Диаграмма деятельности – это блок-схема, которая показывает, как поток управления переходит от одной деятельности к другой, при этом внимание фиксируется на результате деятельности.

Диаграммы деятельности сотрудника, менеджера и администратора предоставлены в соответствии с рисунками 1.3.2-1.3.4.

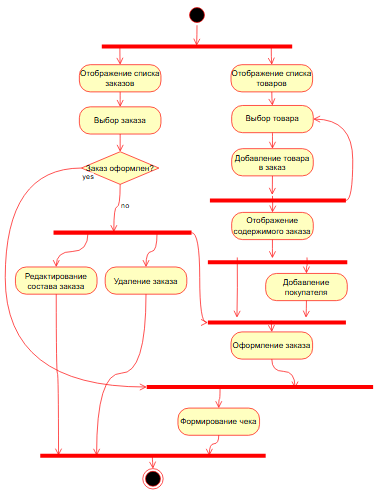


Рисунок 1.3.2 – Диаграмма деятельности сотрудника

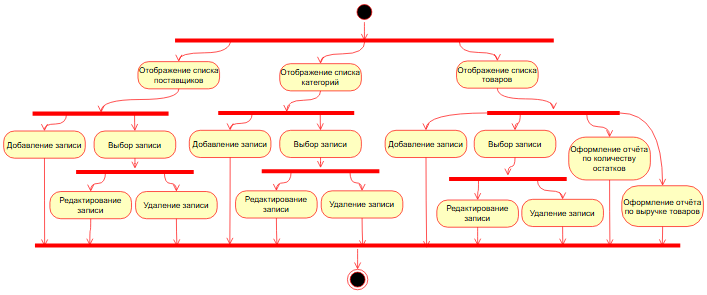


Рисунок 1.3.3 – Диаграмма деятельности менеджера

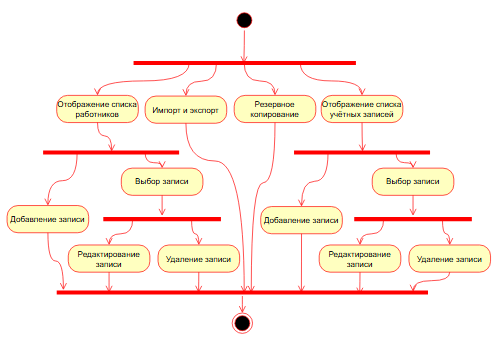


Рисунок 1.3.4 – Диаграмма деятельности администратора

1. **Алгоритм представления задачи**
   1. **Детальное проектирование программного обеспечения**

Прототип -это макет или пробная версия программного продукта.

Варианты прототипов:

* горизонтальный (поведенческий) - моделирует интерфейс пользователя приложения, не затрагивая логики и структуры данных. Горизонтальные прототипы следует использовать, когда необходимо проявить технические требования, имеющие строго оптимальную реализацию. Для горизонтальных прототипов необязательно использовать программную среду реализации, которая будет разрабатывать конечный вариант системы. Если в разработанном интерфейсе используются базы данных, то они интегрируются в программном коде, при этом текст на экране должен отражать реальную специфику предметной области. Иначе пользователю будет трудно сосредоточиться. При создании желательно реализовать ту часть кода, которая отвечает за перемещение между экранами, чтобы пользователь мог понять, как будет реагировать система на его действия;
* эволюционный - создаётся как первое приближение системы, призванное стать в последствии самой системой. Эволюционный прототип должен последовательно перерасти в код целевого приложения. Поэтому данный вид прототипов, требует тщательной разработки приложения и приёмов тестирования результатов.

Был выбран эволюционный прототип, потому что прототип будет постепенно развиваться, превращаясь в конечный продукт.

Прототип программы представлен в соответствии с рисунками 2.1.1-2.1.8

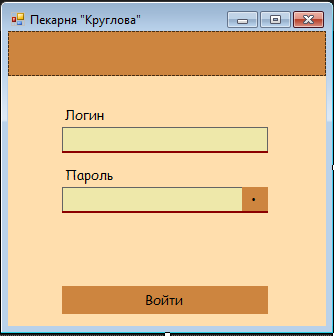


Рисунок 2.1.1 – Прототип формы авторизации

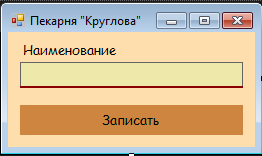


Рисунок 2.1.2 – Прототип формы элемента справочника

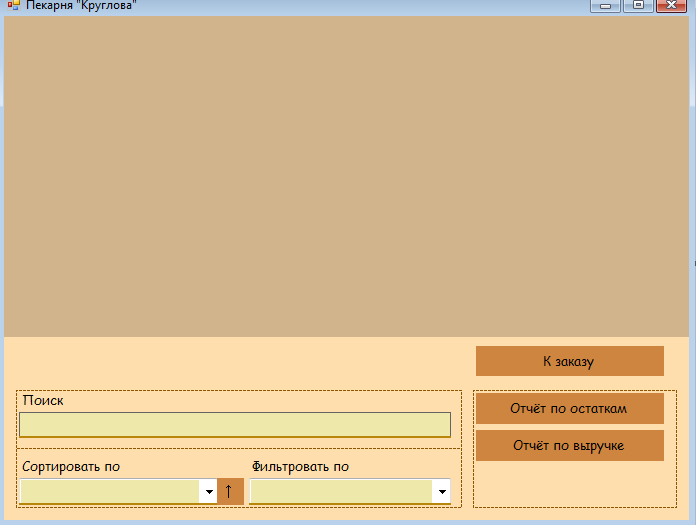


Рисунок 2.1.3 – Прототип формы списка

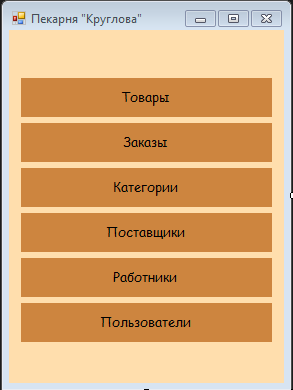


Рисунок 2.1.4 – Прототип формы меню

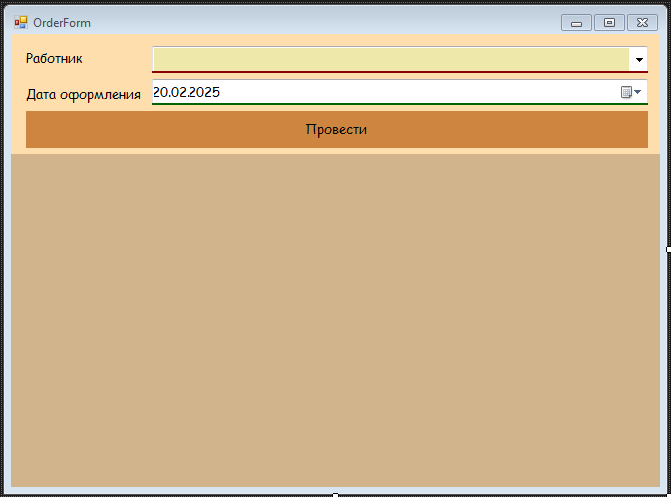


Рисунок 2.1.5 – Прототип формы заказа

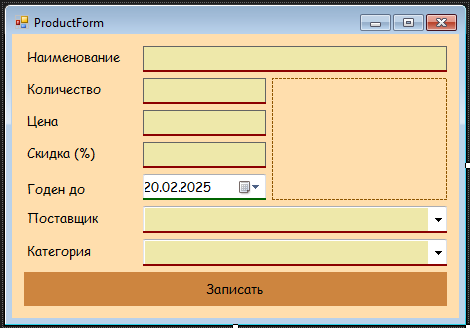


Рисунок 2.1.6 – Прототип формы товара

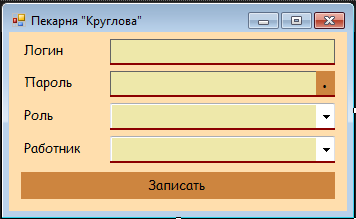


Рисунок 2.1.7 – Прототип формы пользователя

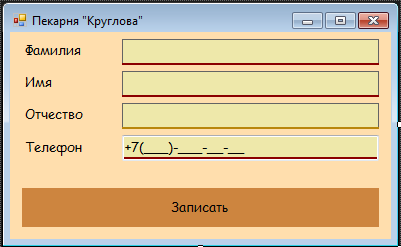


Рисунок 2.1.8 – Прототип формы работника

* 1. **Описание основных программных модулей**

AuthForm - модуль формы авторизации. Устанавливает, форматирует, отображает элементы управления, устанавливает обработчики событий на форму.

Connection – модуль для работы с базой данных: получения, добавления, редактирования и удаления данных.

DateSelectForm - модуль формы выбора даты. Устанавливает, форматирует, отображает элементы управления, устанавливает обработчики событий на форму.

DictionaryForm - модуль формы записи справочника. Устанавливает, форматирует, отображает элементы управления, устанавливает обработчики событий на форму.

ListForm - модуль формы списка. Устанавливает, форматирует, отображает элементы управления, устанавливает обработчики событий на форму.

MenuForm - модуль формы меню. Устанавливает, форматирует, отображает элементы управления, устанавливает обработчики событий на форму.

OrderForm - модуль формы заказа. Устанавливает, форматирует, отображает элементы управления, устанавливает обработчики событий на форму.

ProductForm - модуль формы записи товара. Устанавливает, форматирует, отображает элементы управления, устанавливает обработчики событий на форму.

Program - модуль, инициализирующий работу приложения.

ReportBuilder - модуль компоновки отчётов.

User - объект, содержащий данные о текущем пользователе в системе.

UserForm - модуль формы записи пользователя. Устанавливает, форматирует, отображает элементы управления, устанавливает обработчики событий на форму.

WorkerForm - модуль формы записи работника. Устанавливает, форматирует, отображает элементы управления, устанавливает обработчики событий на форму.

* 1. **Применяемые меры защиты информации**

Защита данных - обеспечение защиты информации от несанкционированного доступа к данным, их изменения или разрушения.

В процессе разработки информационной системы были реализованы следующие меры защиты информации:

* обеспечение безопасности данных:
  + проверка на заполнение обязательных полей;
  + блокировка ввода букв английского алфавита во всех полях, кроме логина и пароля;
  + при добавлении, изменении и удалении записи о пользователе, товаре, категории, поставщике, работнике выводится сообщение о подтверждении действия;
  + при создании продажи автоматически проставляются системная дата, информация об организации;
* обеспечение секретности данных:
  + скрытие части персональных данных пользователей;
  + скрытие пароля;
  + хранение пароля в виде хэша;
* разграничение прав доступа на следующие роли: администратор, менеджер, сотрудник.

**Заключение**

В ходе выполнения курсовой работы была создана автоматизированная информационная система по учёту проданных товаров пекарни «Круглова» на базе платформы .NET.

В процессе создания использовался язык C# и СУБД MySQL

Программный продукт обеспечивает выполнение следующих функций:

* авторизация пользователя (администратор, менеджер, сотрудник);
* программное приложение имеет формы: ведения пользователей, ведения списка товаров, списка продаж, изменения информации о поставщике, категории товара;
* реализация «живого» поиска при работе с информацией о товарах по наименованию, с информацией о пользователях по логину, с информацией справочников по наименованию, с информацией о работниках по фамилии;
* реализация фильтрации на формах с информацией о товарах по поставщику, категории;
* реализация сортировки информации на форме списка товаров по названию, по стоимости;
* реализация совместного использования поиска, сортировки и фильтрации на форме списка продуктов;
* в режиме «Администратор» пользователю предоставляются возможности:
  + просмотр, добавление, редактирование, удаление информации о товарах, продажах, категориях, поставщиках, работниках и пользователях в БД;
  + создание нового пользователя;
* в режиме «Менеджер» пользователю предоставляются возможности:
  + просмотр, добавление, редактирование информации о товарах;
  + создание отчета по количеству остатков, по выручке товаров в Excel;
  + возможность добавления и изменения изображения для товара;
* в режиме «Сотрудник» пользователю предоставляются возможности:
  + просмотр списка товаров;
  + создание заказа с расчетом итоговой стоимости и учетом скидки (скидка на определённый товар);
  + формирование чека;
* автоматизация ввода исходных данных:
  + выпадающий список используется при выборе категорий, поставщиков, работников, ролей;
  + окно календаря используется для выбора даты;
  + при вводе фамилии, имени и отчества запись начинается с заглавной буквы.

В дальнейшем планируется усовершенствовать автоматизированную информационную систему:

* добавление системы CAPTCHA;
* реализация пагинации на основной странице списка товаров;
* реализация функций импорт и экспорт данных в формате CSV;
* расчет общей суммы заказа с учетом скидок;
* реализация пропорционального масштабирования элементов для каждой формы при изменении размера/растягивании всей формы под конкретное расширение экрана;
* условное форматирование данных для визуального мониторинга данных;
* при отсутствии активности работы в системе автоматический перевод на форму авторизации;
* возможность резервного копирования как в автоматическом, так и в ручном режиме.

**Литература**

1 Бен Форта, Освой самостоятельно SQL за 10 минут. М., Вильямс, 2021

2 Мацяшек Л.А., Лионг Б.Л. Практическая программная инженерия на основе учебного примера - М.:БИНОМ, Лаборатория знаний – ЭБК BOOK.RU 2022.

3 Могрунов Е.П. Основы языка SQL, учебное пособие. – СПб.: БХВ-Петербург, 2022.

4 Новиков Б.А., Основы технологий баз данных, Postgres Professional. - М., 2021.

5 Прайс Марк Дж. C# 7 и .NET Core Кросс-платформенная разработка для профессионалов. 3-е изд. - СПб.: Питер, 2022.

6 Эндрю Троелсен Язык программирования C# 6.0 и платформа .NET 4.6, 7-е изд.: пер. с англ. - М.: ООО “И.Д. Вильямс”, 2021.