Департамент образования и науки Костромской области

Областное государственное бюджетное профессиональное образовательное учреждение «Костромской энергетический техникум имени Ф.В. Чижова»

(ОГБПОУ «КЭТ имени Ф.В. Чижова»)

КУРСОВАЯ РАБОТА

**Разработка информационной системы для**

**Работа ресторана «Jaciro»**

ПМ.06 Сопровождение информационных систем

МДК.06.03 Устройство и функционирование информационной системы

специальность: 09.02.07 «Информационные системы и программирование»

Студент Сажин Иван Евгеньевич \_\_\_\_\_\_\_\_\_\_\_\_ 21.12.2022 г.

ФИО Подпись

Оценка выполнения и защиты курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_

Руководитель Бессараб С.К. \_\_\_\_\_\_\_\_\_\_\_\_ 21.12.2022 г.

Подпись

Кострома 2022

**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ КОСТРОМСКОЙ ОБЛАСТИ**

**ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**«КОСТРОМСКОЙ ЭНЕРГЕТИЧЕСКИЙ ТЕХНИКУМ ИМ. Ф.В. ЧИЖОВА»**

**специальность 09.02.07 «Информационные системы и программирование»**

|  |  |  |
| --- | --- | --- |
|  | СОГЛАСОВАНО | **УТВЕРЖДАЮ:** |
|  | (Костромская область) | Зам. директора по учебной работе  \_\_\_\_\_\_\_\_\_\_\_(Е.И.Тюрина)  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_20 г. |
|  |  |  |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г |  |

**КУРСОВОЙ ПРОЕКТ**

**ЗАДАНИЕ**

**Студента \_\_ группы \_\_\_ курса**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**2022 год**

**Тема курсового проекта:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Содержание пояснительной записки

ТИТУЛЬНЫЙ ЛИСТ

ОГЛАВЛЕНИЕ

ГЛОССАРИЙ

ВВЕДЕНИЕ

ГЛАВА 1. ОБОСНОВАНИЕ ОБЪЕКТА АВТОМАТИЗАЦИИ

* 1. Краткая информация об информационной системе
  2. Общие требования к информационной системе
  3. Разработка технического задания на разработку информационной системы
  4. Обоснование необходимости внедрения и использования информационной системы

ГЛАВА 2. БАЗА ДАННЫХ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1. Физическая структура базы данных

ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1. Разработка кода информационной системы

ГЛАВА 4. ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

4.1. Юнит-тестирование

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

СПИСОК ИЛЛЮСТРАЦИЙ

ПРИЛОЖЕНИЯ

Приложение №1 Инструкции для использования системы (руководство пользователя);

Приложение №2 Техническое задание;

Приложение №3 ER-ДИАГРАММА;

Приложение №4 ЛИСТИНГ КОДА;

Приложение №5 Ссылка на облачно хранилище и репозиторий с курсовой работой.

1. Графическая часть
2. ER-ДИАГРАММА;
3. ЛИСТИНГ КОДА;
4. Электронная часть

Содержимое папки с курсовым проектом в облачном хранилище (на репозитории) и на электронном носителе:

1. Папка - Практическая часть (информационная система, исходный код заархивированный; скрипт базы данных);

2. Файл - Текст курсовой работы (в формате .docx) и распечатать. Объем 30 страниц без приложения;

3. Файл - Приложения (техническое задание, диаграммы, скриншоты основных окон ИС, листинг кода);

4. Папка - Презентация и доклад (время на выступление максимум 5 -7 мин, 15 слайдов максимум);

5. Файл - Видео (с демонстрацией работы основного функционала не больше 1 мин (видео в формате mp4 объем до 10 Мб);

6. Файл - readme.txt (внутри: ФИО, тема курсовой работы, ссылка на облачное хранилище, ссылка на репозиторий);

Задание получил\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания: \_\_\_\_\_\_\_

Срок выполнения: \_\_\_\_\_\_\_\_\_\_

Руководитель курсового проекта: преподаватель \_\_\_\_\_\_\_\_\_ (Бессараб С.К.)

Рассмотрено и утверждено на заседании ПЦК \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Председатель ПЦК \_\_\_\_\_\_\_\_\_(Ищук О.В.)

Введение

Вторая половина ХХ века запомнилась нам, как начало четвертой информационной революции, которая сильно повлияла на будущее всего человечества, идущая по сей день. Начавшись с изобретения микропроцессорной технологии, создания первого персонального компьютера и интернета, она с каждым годом вносила все более колоссальные изменения в жизни людей.

Сегодня людей не сильно удивляют «очередные» открытия в информационной культуре, ведь поток информации очень велик и уследить за всем очень сложно и почти невозможно. Но одним из самых значимых открытий являются «Информационные технологии».

Впервые об этом упоминалось в статье 1958 года «Harvard Business Review». Её авторы, Гарольд Дж. Ливитт и Томас Л. Уислер отметили, что «У этой технологии нет единого и общего названия. Мы будем называть её информационной технологией (IT)».

Чем же так хороши информационные технологии? В жизни эти технологии направлены на анализ уже имеющейся информации для улучшения ее качества, в производстве на упрощение, улучшение старых, или создание новых методов работы, которые, впоследствии, должны положительно повлиять на получаемую прибыль.

В настоящее время ни одно производство не пренебрегает в использовании информационных технологиях, и ресторан тому не исключение. Они упрощают бухгалтерский учет, планирование закупок, банкетов, смен работников и так далее.

**Объект исследования**: Ресторан «Jaciro»

**Предмет исследования**: автоматизация приёма заказов гостей ресторана

**Цель**: Создание приложения для приема и редактирования заказов ресторана.

**Задачи:**

1. Провести анализ предметной области, в которой требуется применение информационной системы;
2. Спроектировать структуру будущей информационной системы;
3. Спроектировать структуру базы данных;
4. Выполнить программную реализацию информационной системы;
5. Протестировать информационную систему.

Таким образом, разработанная информационная система – это совокупность программно-технических средств, баз данных и знаний, предназначенных для автоматизации приема заказа гостей ресторана.

ГЛАВА 1. ОБОСНОВАНИЕ ОБЪЕКТА АВТОМАТИЗАЦИИ

**Информационная система (ИС)** — система, предназначенная для хранения, поиска и обработки информации, и соответствующие организационные ресурсы (человеческие, технические, финансовые и т. д.), которые обеспечивают и распространяют информацию (ISO/IEC 2382:2015).

Информационная система «Jaciro» предназначения для оптимизации и автоматизации приёма заказов гостей ресторана.

* 1. Краткая информация об информационной системе

В данном разделе представлена кратная информация об разработанной информационной системе в виде таблицы

Таблица 1

Краткая информация об информационной системе

|  |  |
| --- | --- |
| **Название** | **Значение** |
| Наименование | «Jaciro» |
| Версия | 1.0 Beta |
| Архитектура | Локальная |
| Характер обработки данных | Информационно-справочная |
| Сфера | Общее питание |
| Язык разработки | WPF NetFrameWork |

Представленная таблица дает краткое представление об информационной системе

* 1. Общие требования к информационной системе

Информационная система «Jaciro» должна отвечать следующим требованиям:

1. **Гибкость**. Способность к адаптации и дальнейшему развитию подразумевают возможность приспособления информационной системы к новым условиям, новым потребностям предприятия.
2. **Надежность**. Требование надёжности обеспечивается созданием резервных копий хранимой информации, выполнения операций протоколирования, поддержанием качества каналов связи и физических носителей информации, использованием современных программных и аппаратных средств.
3. **Эффективность**. Система является эффективной, если с учётом выделенных ей ресурсов она позволяет решать возложенные на неё задачи в минимальные сроки. Эффективность системы обеспечивается оптимизацией данных и методов их обработки, применением оригинальных разработок, идей, методов проектирования.
4. **Безопасность**. Под безопасностью, прежде всего, подразумевается свойство системы, в силу которого посторонние лица не имеют доступа к информационным ресурсам организации, кроме тех, которые для них предназначены. Требование безопасности обеспечивается современными средствами разработки информационных систем, современной аппаратурой, методами защиты информации, применением паролей и протоколированием, постоянным мониторингом состояния безопасности операционных систем и средств их защиты.

**Функциональные требования к системе:**

1. Система должна обеспечивать соответствие разрабатываемых форм и документов, отчетов, бизнес-процессов текущим требованиям законодательства, регулирующих органов и пакету методологических документов, разрабатываемых в рамках объема проекта;
2. Система должна обеспечить возможность корректного ведения бухгалтерского и налогового учета, учета МСФО и составления отчетности, в том числе налоговой, в соответствии с нормами и правилами, установленными Законодательством РФ;
3. Система должна обеспечивать защиту от неправильных действий персонала, приводящих к аварийному состоянию Системы, от случайных изменений и разрушения информации и программ.
   1. Разработка технического задания на разработку информационной системы

**Техническое задание (ТЗ, техзадание)** — документ или несколько документов, определяющих цель, структуру, свойства и методы какого-либо проекта, и исключающие двусмысленное толкование различными исполнителями. Техническое задание используется в машиностроении, производстве и бизнесе для того, чтобы поставщики, покупатели и пользователи материалов, продуктов или услуг понимали и согласовывали все требования.

Техническое задание оформляют в соответствии с общими требованиями к текстовым конструкторским документам по ГОСТ 2.105-95 (ЕСКД. Общие требования к текстовым документам) на листах формата А4, как правило, без рамки и основной надписи.

*Этапы разработки технического задания*

|  |  |
| --- | --- |
| **Этап работы** | **Содержание работы** |
| Обоснование необходимости разработки программы | 1. Постановка задачи 2. Сбор исходных материалов 3. Выбор и обоснование критериев эффективности и качества разрабатываемой программы. 4. Обоснование необходимости проведения научно-исследовательских работ. |
| Научно-исследовательские работы | Определение структуры входных и выходных данных.  Предварительный выбор методов решения задач.  Обоснование целесообразности применения ранее разработанных программ.  Определение требований к техническим средствам.  Обоснование принципиальной возможности решения поставленной задачи |
| Разработка и утверждение технического задания | Определение требований к программе.  Разработка технико-экономического обоснования разработки программы.  Определение стадий, этапов и сроков разработки программы и документации на неё.  Выбор языков программирования.  Определение необходимости проведения научно-исследовательских работ на последующих стадиях.  Согласование и утверждение технического задания. |

* 1. Обоснование необходимости внедрения и использования информационной системы

**Внедрение информационной системы** — это процессы слияния программного продукта с финансовой и управленческой работой организации, которые требуют от исполнителей, занятых в проекте, наличия специализированных знаний, определённого опыта и требуемого подхода к осуществлению проекта.

Основной целью внедрения ИС является создание единого информационного пространства, позволяющую решать не только учетные функции, но и управленческие задачи.

Предприятие-заказчик сам определяет для себя вариант внедрения. Существуют два варианта: **комплексное внедрение** и **внедрение собственными силами.**

Комплексные системы достаточно дороги, внедрять их довольно сложно и долго. Многие системы невозможно адаптировать без разработчика, а предприятия зачастую сильно удалены от них. Примером могут служить все импортные системы. В дополнении к сказанному: для импортных систем высока стоимость работы консультантов, стоимость обучения, для эффективной работы с разработчиком крайне желательно знание иностранного языка.

ГЛАВА 2. БАЗА ДАННЫХ ИНФОРМАЦИОННОЙ СИСТЕМЫ

База данных очень важна для бесперебойного взаимодействия систем, содержащих глобальную информацию. Например, о государственных ресурсах или определенных территориях. В них детально расписаны все данные о строениях, гидрографии и растительности данных мест.

* 1. Физическая структура базы данных

**База данных** – это совокупность массивов и файлов данных, организованная по определённым правилам, предусматривающим стандартные принципы описания, хранения и обработки данных независимо от их вида.

**База данных** – это совокупность организованной информации, относящейся к определённой предметной области, предназначенная для длительного хранения во внешней памяти компьютера и постоянного применения.

**Система управления базами данных, сокр. СУБД (англ. Database Management System, сокр. DBMS)** — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Виды баз данных:

1. **Фактографическая** – содержит краткую информацию об объектах некоторой системы в строго фиксированном формате;
2. **Документальная** – содержит документы самого разного типа: текстовые, графические, звуковые, мультимедийные;
3. **Распределённая** – база данных, разные части которой хранятся на различных компьютерах, объединённых в сеть;
4. **Централизованная** – база данных, хранящихся на одном компьютере;
5. **Реляционная** – база данных с табличной организацией данных;
6. **Неструктурированная (NoSQL)** - база данных, в которой делается попытка решить проблемы масштабируемости и доступности за счёт атомарности (англ. atomicity) и согласованности данных, но не имеющих четкой (реляционной) структуры.

**Проектирование базы данных** — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные этапы проектирования баз данных

1. Концептуальное (инфологическое) проектирование
2. Логическое (даталогическое) проектирование
3. Физическое проектирование

Процедуры концептуального проектирования

Цель этапа концептуального проектирования - создание концептуальной модели данных исходя из представлений пользователей о предметной области. Для ее достижения выполняется ряд последовательных процедур.

* + - 1. Определение сущностей и их документирование. Для идентификации сущностей определяются объекты, которые существуют независимо от других. Такие объекты являются сущностями. Каждой сущности присваивается осмысленное имя, понятное пользователям. Имена и описания сущностей заносятся в словарь данных. Если возможно, то устанавливается ожидаемое количество экземпляров каждой сущности.
      2. Определение связей между сущностями и их документирование. Определяются только те связи между сущностями, которые необходимы для удовлетворения требований к проекту базы данных. Устанавливается тип каждой из них. Выявляется класс принадлежности сущностей. Связям присваиваются осмысленные имена, выраженные глаголами. Развернутое описание каждой связи с указанием ее типа и класса принадлежности сущностей, участвующих в связи, заносится в словарь данных.
      3. Создание ER-модели предметной области. Для представления сущностей и связей между ними используются ER-диаграммы. На их основе создается единый наглядный образ моделируемой предметной области - ER-модель предметной области.
      4. Определение атрибутов и их документирование. Выявляются все атрибуты, описывающие сущности созданной ER-модели. Каждому атрибуту присваивается осмысленное имя, понятное пользователям. О каждом атрибуте в словарь данных помещаются следующие сведения:
* имя атрибута и его описание:
* тип и размерность значений:
* значение, принимаемое для атрибута по умолчанию (если такое имеется);
* может ли атрибут иметь Null-значения;
* является ли атрибут составным, и если это так, то из каких простых атрибутов он состоит. Например, атрибут "Ф.И.О. клиента" может состоять из простых атрибутов "Фамилия", "Имя", "Отчество", а может быть простым, содержащим единые значения, как-то "Сидорский Евгений Михайлович". Если пользователь не нуждается в доступе к отдельным элементам "Ф.И.О.", то атрибут представляется как простой:
* является ли атрибут расчетным, и если это так, то как вычисляются его значения.

1. Определение значений атрибутов и их документирование. Для каждого атрибута сущности, участвующей в ER-модели, определяется набор допустимых значений и ему присваивается имя. Например, атрибут "Тип счета" может иметь только значения "Депозитный", "текущий", "до востребования", "карт-счет". Обновляются записи словаря данных, относящиеся к атрибутам, - в них заносятся имена наборов значений атрибутов.
2. Определение первичных ключей для сущностей и их документирование. На этом шаге руководствуются определением первичного ключа - как атрибута или набора атрибутов сущности, позволяющего уникальным образом идентифицировать ее экземпляры. Сведения о первичных ключах помещаются словарь данных.
3. Обсуждение концептуальной модели данных с конечными пользователями. Концептуальная модель данных представляется ER-моделью с сопроводительной документацией, содержащей описание разработанной модели данных. Если будут обнаружены несоответствия предметной области, то в модель вносятся изменения до тех пор, пока пользователи не подтвердят, что предложенная им модель адекватно отображает их личные представления.

Процедуры логического проектирования

Цель этапа логического проектирования - преобразование концептуальной модели на основе выбранной модели данных в логическую модель, не зависимую от особенностей используемой в дальнейшем СУБД для физической реализации базы данных. Для ее достижения выполняются следующие процедуры.

1. Выбор модели данных. Чаще всего выбирается реляционная модель данных в связи с наглядностью табличного представления данных и удобства работы с ними.
2. Определение набора таблиц исходя из ER-модели и их документирование. Для каждой сущности ER-модели создается таблица. Имя сущности - имя таблицы.

Осуществляется формирование структуры таблиц на основании изложенных в параграфе 1.4 правил. Устанавливаются связи между таблицами посредством механизма первичных и внешних ключей. Структуры таблиц и установленные связи между ними документируются.

1. Нормализация таблиц. Для правильного выполнения нормализации проектировщик должен глубоко изучить семантику и особенности использования данных. На этом шаге он проверяет корректность структуры таблиц, созданных на предыдущем шаге, посредством применения к ним процедуры нормализации. Эта процедура была описана в параграфе 1.5. Она заключается в приведении каждой из таблиц, по крайней мере, к 3НФ. В результате нормализации получается очень гибкий проект базы данных, позволяющий легко вносить в нее нужные расширения.
2. Проверка логической модели данных на предмет возможности выполнения всех транзакций, предусмотренных пользователями. Транзакция — это набор действий, выполняемых отдельным пользователем или прикладной программой с целью изменения содержимого базы данных. Так, примером транзакции в проекте БАНК может быть передача права распоряжаться счетами некоторого клиента другому клиенту. В этом случае в базу данных потребуется внести сразу несколько изменений. Если во время выполнения транзакции произойдет сбой в работе компьютера, то база данных окажется в противоречивом состоянии, так как некоторые изменения уже будут внесены, а остальные еще нет. Поэтому все частичные изменения должны быть отменены для возвращения базы данных в прежнее непротиворечивое состояние. Перечень транзакций определяется действиями пользователей в предметной области. Используя ER-модель, словарь данных и установленные связи между первичными и внешними ключами, производится попытка выполнить все необходимые операции доступа к данным вручную. Если какую-либо операцию выполнить вручную не удается, то составленная логическая модель данных является неадекватной и содержит ошибки, которые надо устранить. Возможно, они связаны с пропуском в модели сущности, связи или атрибута.
3. Определение требований поддержки целостности данных и их документирование. Эти требования представляют собой ограничения, которые вводятся с целью предотвратить помещение в базу данных противоречивых данных. На этом шаге вопросы целостности данных освещаются безотносительно к конкретным аспектам ее реализации. Должны быть рассмотрены следующие типы ограничений:

* обязательные данные. Выясняется, есть ли атрибуты, которые не могут иметь Null-значений;
* ограничения для значений атрибутов. Определяются допустимые значения для атрибутов:
* целостность сущностей. Она достигается, если первичный ключ сущности не содержит Null-значений:
* ссылочная целостность. Она понимается так, что значение внешнего ключа должно обязательно присутствовать в первичном ключе одной из строк таблицы для родительской сущности;
* ограничения, накладываемые бизнес-правилами. Например, в случае с проектом БАНК может быть принято правило, запрещающее клиенту распоряжаться, скажем, более чем тремя счетами.

Сведения обо всех установленных ограничениях целостности данных помещаются в словарь данных.

1. Создание окончательного варианта логической модели данных и обсуждение его с пользователями. На этом шаге подготавливается окончательный вариант ER-модели, представляющей логическую модель данных. Сама модель и обновленная документация, включая словарь данных и реляционную схему связи таблиц, представляется для просмотра и анализа пользователям, которые должны убедиться, что она точно отображает предметную область.

Процедуры физического проектирования

Цель этапа физического проектирования хранения данных и эффективных методов доступа к данным базы. При логическом проектировании отвечают на вопрос - что надо сделать, а при физическом- выбирается способ, как это сделать. Процедуры физического проектирования следующие.

1. Проектирование таблиц базы данных средствами выбранной СУБД. Осуществляется выбор реляционной СУБД. которая будет использоваться для создания базы данных, размещаемой на машинных носителях. Глубоко изучаются ее функциональные возможности по проектированию таблиц. Затем выполняется проектирование таблиц и схемы их связи в среде СУБД. Подготовленный проект базы данных описывается в сопровождаемой документации.
2. Реализация бизнес-правил в среде выбранной СУБД. Обновление информации в таблицах может быть ограничено бизнес-правилами. Способ их реализации зависит от выбранной СУБД. Одни системы для реализации требований предметной области предлагают больше возможностей, другие- меньше. В некоторых системах вообще отсутствует поддержка реализации бизнес-правил. В таком случае разрабатываются приложения для реализации их ограничений. Все решения, принятые в связи с реализацией бизнес-правил предметной области, подробно описываются в сопроводительной документации.
3. Проектирование физической организации базы данных. На этом шаге выбирается наилучшая файловая организация для таблиц. Выявляются транзакции, которые будут выполняться в проектируемой базе данных, и выделяются наиболее важные из них. Анализируется пропускная способность транзакций - количество транзакций, которые могут быть обработаны за заданный интервал времени, и время ответа - промежуток времени, необходимый для выполнения одной транзакции. На основании указанных показателей принимаются решения об оптимизации производительности базы данных путем определения индексов в таблицах, ускоряющих выборку данных из базы, или снижения требований к уровню нормализации таблиц. Проводится оценка дискового объема памяти. необходимого для размещения создаваемой базы данных. Стремятся к его минимизации.

Принятые решения по изложенным вопросам документируются.

1. Разработка стратегии защиты базы данных. База данных представляет собой ценный корпоративный ресурс, и организации ее зашиты уделяется большое внимание. Для этого проектировщики должны иметь полное и ясное представление обо всех средствах зашиты, предоставляемых выбранной СУБД.
2. Организация мониторинга функционирования базы данных и ее настройка. После создания физического проекта базы данных организуется непрерывное слежение за ее функционированием. Полученные сведения об уровне производительности базы данных используются для ее настройки. Для этого привлекаются и средства выбранной СУБД. Решения о внесении любых изменений в функционирующую базу данных должны быть обдуманными и всесторонне взвешенными. описание конкретной реализации базы данных, размещаемой во внешней памяти компьютера. Это описание структуры.

ERD модель базы данных ИС «Jaciro» представлена на Рисунке 1. База данных ИС «Jaciro» содержит 9 таблиц:

**Staff** – Таблица данных сотрудников ресторана

**Orders** – Таблица данных о всех заказах

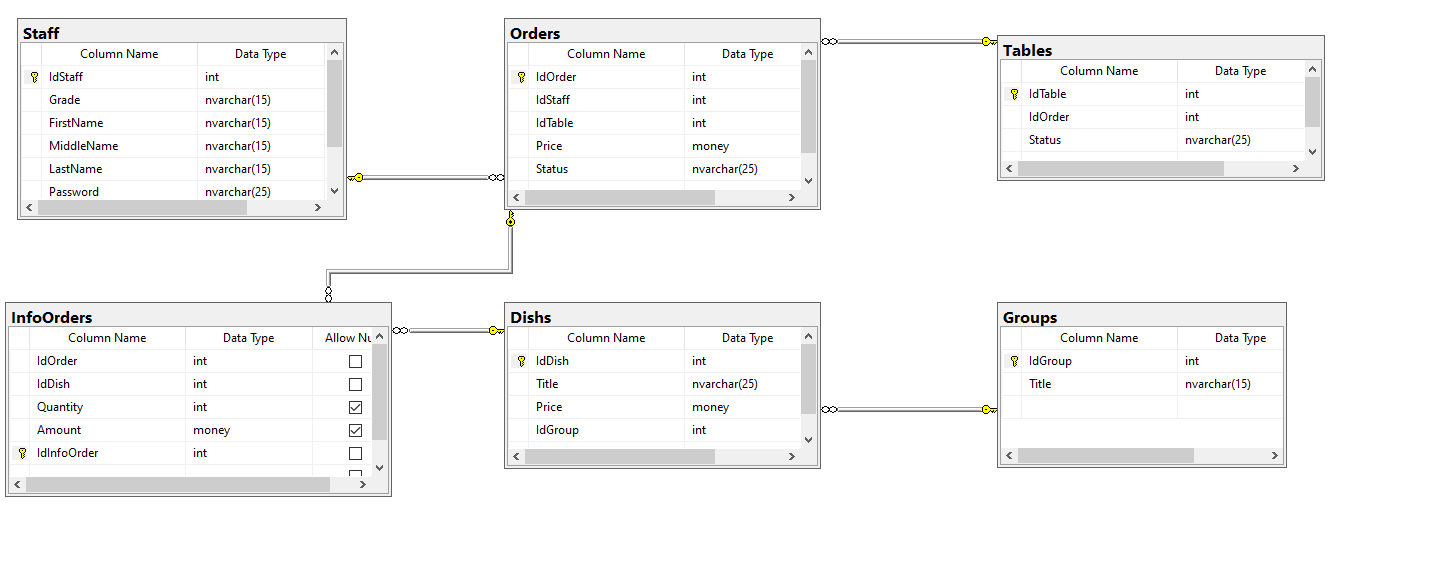
**InfoOrders** – Таблица данных о заказе

**Dishs** – Таблица данных о блюдах

**Drinks** – Таблица данных о напитках

**Groups** – Таблица данных о группах блюд и напитков

**Tables** – Таблица данных о столах



*Рисунок 1*

ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

Информационная система представляет собой локальное приложение.

3.1. Разработка кода информационной системы

**Разработка программного обеспечения** — деятельность по созданию нового программного обеспечения. Разработка программного обеспечения как инженерная дисциплина является составной частью программной инженерии, наряду с дисциплинами, отвечающими за функционирование и сопровождение программных продуктов

**Процесс** — совокупность взаимосвязанных или взаимодействующих видов деятельности, преобразующих входы в выходы.

Процесс разработки состоит из множества подпроцессов, или дисциплин, некоторые из которых перечислены ниже.

1. Анализ требований → Спецификация программного обеспечения
2. Проектирование программного обеспечения
3. Программирование
4. Тестирование программного обеспечения
5. Системная интеграция (System integration)
6. Внедрение программного обеспечения (или Установка программного обеспечения)
7. Сопровождение программного обеспечения

ГЛАВА 4. ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

4.1. План тестирования

**Тест-план** – это большой документ, который чаще всего описывает весь объем работ по тестированию проекта либо части проекта (например, релиза или предрелизного билда). План тестирования описывает, что будет тестироваться, в какие сроки, какими инструментами, какая команда, обязанности и ответственности каждого члена команды.

**Тестирование** – это проверка созданного программного продукта на соответствие заданным требованиям, и на отсутствие дефектов. Тестирование подразумевает запуск программы (или ее компонентов) при помощи ручных (или автоматизированных) программных средств тестирования; тестироваться могут и отдельные компоненты (или отдельные свойства) программы.

**Тестирование программного обеспечения** — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом. Тестирование является неотъемлемой частью жизненного цикла программного обеспечения. Само по себе тестирование – длительный процесс проверок на соответствие ожидаемого результата. Нельзя выделить какой-то один этап как важный, каждый из них имеет одинаковый вес. При создании продукта тестировщик не просто играет важную роль, а участвует на каждом этапе разработки от концепции до выхода продукта в свет.

Как уже было отмечено, тестирование – неотъемлемая часть жизненного цикла продукта, однако необходимо понимать, с чего стоит начинать тестирование, и когда оно заканчивается. Например, неопытные специалисты ошибочно думают, что после релиза они могут переключаться на другой проект.

Всего принято выделять 7 этапов тестирования:

* **Работа с требованиями.** Знакомство с требованиями заказчика, что должен из себя представлять итоговый продукт, обсуждение.
* **Разработка стратегии тестирования.** Оценка сроков тестирования, выявление среды тестирования, объединение всей информации, полученной при работе с требованиями.
* **Создание тестовой документации.** Написание сценариев, которые позволят проверить функционал.
* **Тестирование прототипа.** Тестирование основного функционала продукта, корректировка целей, добавление фичей.
* **Основное тестирование.** Выполнение общей проверки продукта.
* **Стабилизация.** На данном этапе происходит работа над устранением багов.
* **Эксплуатация.** Проводится регресс-тестирование, устранение ошибок, которые нашел конечный пользователь.

4.2. Написание тест-кейсов

**Тест-кейс** — это профессиональная документация тестировщика, последовательность действий, направленная на проверку какого-либо функционала, описывающая как прийти к фактическому результату.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Название | Модуль | Предусловия | Шаги | Ожидаемый результат |
| 1 | Входа в систему разных ролей | Авторизация |  | 1. Написать пароль принадлежащие пользователю 2. Войти в систему. | Вход в систему |
| 2 | Создание заказа | Создание заказа | Войти в систему | 1. Нажать на кнопку «Заказы» 2. Нажать на кнопку «Создать Заказ» 3. Выбрать Блюдо и указать его количество 4. Нажать на кнопку «Добавить Блюдо» | 1. Новый заказ отобразиться в списке всех заказов |
| 3 | Добавить сотрудника | Сотрудники | Войти в систему | 1. Нажать на кнопку «Сотрудники» 2. Нажать на кнопку «Добавить сотрудника» | 1. Созданный сотрудник отобразится в списке всех сотрудников |

***Тест-кейс №1***

Таким образом, в ходе тестирования информационной системы на наличие ошибок, были выявлены ошибки, которые помогли улучшить систему.

* 1. Юнит-тестирование

**Юнит тестирование** – это процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы. Модульное тестирование проводят сразу после написания кода. Поскольку каждый модуль пишется отдельно, тестировать его тоже можно изолированно, без связки с другими.

Если пропустить этап юнит-тестирования, в следующий раз не получится понять, что именно вызвало ошибку: какой-то из модулей или неправильно настроенная интеграция между ними. Придётся разбираться, тратить время и всё равно тестировать отдельные юниты.

Заключение

В итоге работы была разработана информационная система для ресторана «Jaciro»

**Были выполнены следующие задачи**

1. Провести анализ предметной области, в которой требуется применение информационной системы;
2. Спроектировать структуру будущей информационной системы;
3. Спроектировать структуру базы данных;
4. Выполнить программную реализацию информационной системы;
5. Протестировать информационную систему.

В результате работы были закреплены знания в разработке приложений на платформе Microsoft Visual Studio, в среде выполнения .net Framwork.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

|  |  |
| --- | --- |
| [1] | «Microsoft Общие сведения Visual Studio» [В Интернете]. Available: https://learn.microsoft.com/ru-ru/VisualStudio/get-started/visual-studio-ide?view=vs-2019 |
| [2] | «WSR Программные решения для бизнеса» [В Интернете]. Available: https://nationalteam.worldskills.ru/skills/programmnye-resheniya-dlya-biznesa/ |
| [3] | «Metanit C#» [В Интернете]. Available:  https://metanit.com/sharp/ |
| [4] | «Что такое СУБД,» [В Интернете]. Available:  https://www.nic.ru/help/chto-takoe-subd\_8580.html. |
| [5] | «Этапы проектирования баз данных,» [В Интернете]. Available: https://studopedia.ru/2\_10121\_etapi-proektirovaniya-baz-dannih.html. |
| [6] | А. Н. Королева, «Основные объекты СУБД: таблицы, формы, запросы, отчеты,» 01 06 2017. [В Интернете]. Available: https://spravochnick.ru/bazy\_dannyh/sistema\_upravleniya\_bazami\_dannyh\_-\_subd/osnovnye\_obekty\_subd\_tablicy\_formy\_zaprosy\_otchety/. |
| [7] | «Фундаментальная теория тестирования,» 25 3 2021. [В Интернете]. Available:  https://habr.com/ru/post/549054/. |
| [8] | «Виды тестовой документации,» [В Интернете]. Available: https://bugza.info/vidy-testovoj-dokumentacii/. |
| [9] | «Как писать тест-кейсы: полное руководство,» [В Интернете]. Available: https://testengineer.ru/kak-pisat-test-kejsy-polnoe-rukovodstvo/. |
|  |  |
|  |  |

СПИСОК ИЛЛЮСТРАЦИЙ и листинг кода

MainWindow.xaml

Title="MainWindow" Height="600" Width="1000" Icon="Image/unnamed1.jpg">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="100"></RowDefinition>

<RowDefinition Height="\*"></RowDefinition>

<RowDefinition Height="50"></RowDefinition>

</Grid.RowDefinitions>

<StackPanel Grid.Row="0" Orientation="Horizontal" Background="Black">

<Image Source="Image/unnamed1.jpg" Margin="15 0 0 0" Height="75" />

<Button Content="Сотрудники" x:Name="CreateStaff" Height="35" Width="140" Margin="15 0 0 0" Click="CreateStaff\_Click" Visibility="Hidden"/>

<Button Content="Заказы" x:Name="UpdateOrder" Height="35" Width="140" Margin="15 0 0 0" Click="Order\_Click" Visibility="Hidden"/>

<TextBlock Height="25" Margin="15 0 0 0" Width="225" x:Name="Role" Foreground="White" />

<TextBox Height="25" Margin="15 0 0 0" Width="90" x:Name="Password"/>

<Button Content="Войти" x:Name="SignIn" Height="25" Width="75" Margin="15 0 0 0" Click="SignIn\_Click"/>

</StackPanel>

<Frame Grid.Row="1" x:Name="MainFrame" NavigationUIVisibility="Hidden" />

<StackPanel Grid.Row="2" Orientation="Horizontal" Background="Black" >

<Button Content="Выйти" x:Name="Exit" Height="35" Width="140" Margin="15 0 0 0" Click="Exit\_Click" Visibility="Hidden"/>

</StackPanel>

</Grid>

MainWindow.xaml.cs

public partial class MainWindow : Window

{

public static Staff \_staff;

public MainWindow()

{

InitializeComponent();

MainFrame.Navigate(new ClearPage());

ClassPage.MainFrame = MainFrame;

}

private void Exit\_Click(object sender, RoutedEventArgs e)

{

Password.Text = "";

Role.Text = "";

SignIn.Visibility = Visibility.Visible;

Password.Visibility = Visibility.Visible;

UpdateOrder.Visibility = Visibility.Hidden;

Exit.Visibility = Visibility.Hidden;

CreateStaff.Visibility = Visibility.Hidden;

MainFrame.Navigate(new ClearPage());

ClassPage.MainFrame = MainFrame;

}

private void Order\_Click(object sender, RoutedEventArgs e)

{

MainFrame.Navigate(new OrdersPage());

ClassPage.MainFrame = MainFrame;

}

private void SignIn\_Click(object sender, RoutedEventArgs e)

{

foreach (var i in KursovayaEntities.GetContext().Staffs)

{

if (Password.Text == i.Password)

{

Role.Text = i.FirstName + " " + i.Grade;

\_staff = i;

if (i.Grade == "Менеджер")

{

CreateStaff.Visibility = Visibility.Visible;

}

SignIn.Visibility = Visibility.Hidden;

Password.Visibility = Visibility.Hidden;

UpdateOrder.Visibility = Visibility.Visible;

Exit.Visibility = Visibility.Visible;

}

}

}

private void CreateStaff\_Click(object sender, RoutedEventArgs e)

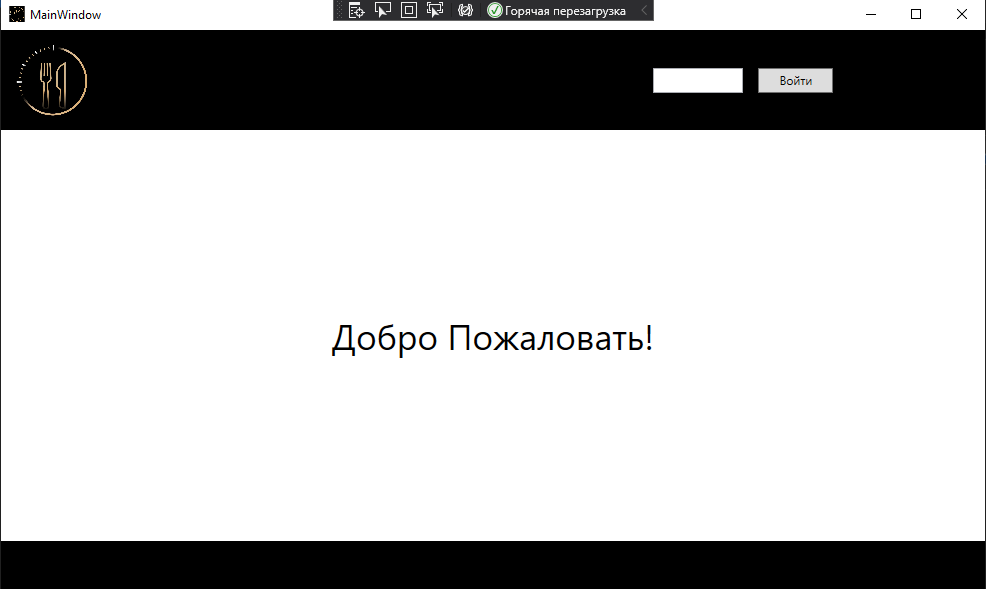
{

MainFrame.Navigate(new StaffPage());

ClassPage.MainFrame = MainFrame;

}

}



ClassPage.cs

class ClassPage

{

public static Frame MainFrame { get; set; }

}

StaffPage.xaml

Title="StaffPage" IsVisibleChanged="Page\_IsVisibleChanged">

<Grid>

<ListView x:Name="LV" >

<ListView.ItemTemplate>

<DataTemplate>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<TextBlock Text="Должность" Foreground="Black" Grid.Column="0" Grid.Row="0" Margin="0 0 15 0" FontWeight="Bold"/>

<TextBlock Text="{Binding Grade}" Foreground="Black" Grid.Column="1" Grid.Row="0"/>

<TextBlock Text="Имя" Foreground="Black" Grid.Column="0" Grid.Row="1" Margin="0 0 15 0" FontWeight="Bold"/>

<TextBlock Text="{Binding MiddleName}" Foreground="Black" Grid.Column="1" Grid.Row="1"/>

<TextBlock Text="Фамилия" Foreground="Black" Grid.Column="0" Grid.Row="2" Margin="0 0 15 0" FontWeight="Bold"/>

<TextBlock Text="{Binding FirstName}" Foreground="Black" Grid.Column="1" Grid.Row="2"/>

<TextBlock Text="Отчество" Foreground="Black" Grid.Column="0" Grid.Row="3" Margin="0 0 15 0" FontWeight="Bold"/>

<TextBlock Text="{Binding LastName}" Foreground="Black" Grid.Column="1" Grid.Row="3"/>

<Button Width="150" Content="Редактировать" Height="20" Grid.Column="0" Grid.Row="4" x:Name="UpdateStaff" Margin="0 15 15 15" Click="UpdateStaff\_Click"/>

<Button Width="150" Content="Удалить" Height="20" Grid.Column="1" Grid.Row="4" x:Name="DeleteStaff" Margin="0 15 0 15" Click="DeleteStaff\_Click"/>

</Grid>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

<Button Content="Добавить Сотрудника" Width="250" Height="40" x:Name="CreateStaff" Click="CreateStaff\_Click" VerticalAlignment="Center" HorizontalAlignment="Right" Margin="0 0 35 0"/>

</Grid>

StaffPage.xaml.cs

public partial class StaffPage : Page

{

public StaffPage()

{

InitializeComponent();

LV.ItemsSource = KursovayaEntities.GetContext().Staffs.ToList();

private void CreateStaff\_Click(object sender, RoutedEventArgs e)

{

ClassPage.MainFrame.Navigate(new CreateStaff(null));

}

private void UpdateStaff\_Click(object sender, RoutedEventArgs e)

{

var \_staff = (sender as Button).DataContext as Staff;

ClassPage.MainFrame.Navigate(new CreateStaff(\_staff));

}

private void DeleteStaff\_Click(object sender, RoutedEventArgs e)

{

var \_staff = (sender as Button).DataContext as Staff;

KursovayaEntities.GetContext().Staffs.Remove(\_staff);

KursovayaEntities.GetContext().SaveChanges();

(LV.ItemsSource as List<Staff>).Remove(\_staff);

LV.Items.Refresh();

MessageBox.Show("Сотрудник удален!");

}

private void Page\_IsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)

{

if (Visibility == Visibility.Visible)

{

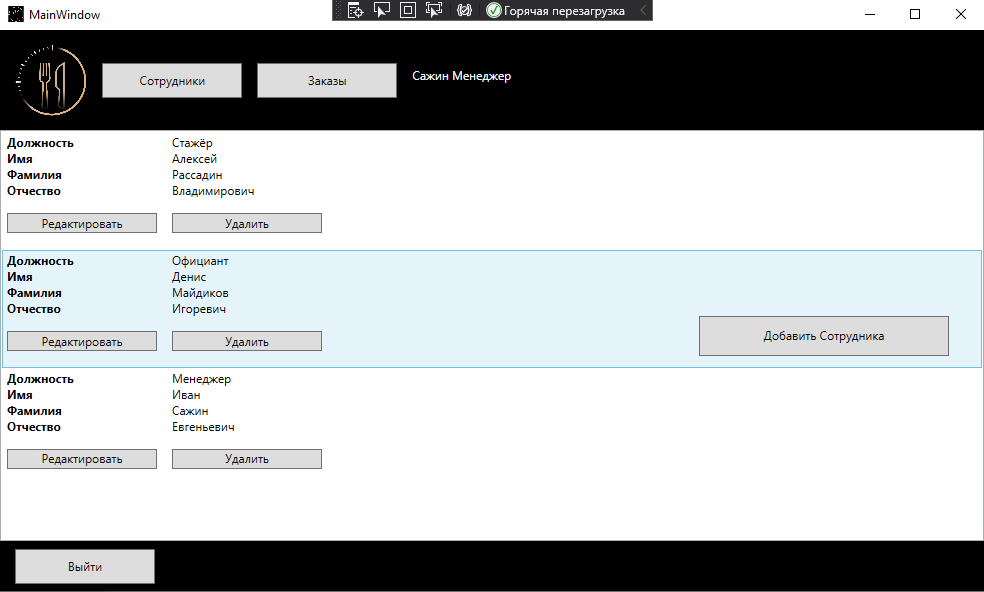
KursovayaEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());

LV.ItemsSource = KursovayaEntities.GetContext().Staffs.ToList();

}

}

}



CreateStaff.xaml

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<TextBlock Text="Введите Фамилию сотрудника" Grid.Row="0" Grid.Column="0" Height="30" FontSize="25"/>

<TextBox x:Name="TB0" Grid.Row="0" Height="30" Width="200" Grid.Column="1" HorizontalAlignment="Left" Text="{Binding FirstName}"/>

<TextBlock Text="Введите Имя сотрудника" Grid.Row="1" Grid.Column="0" Height="30" FontSize="25"/>

<TextBox Grid.Row="1" x:Name="TB1" Height="30" Width="200" Grid.Column="1" HorizontalAlignment="Left" Text="{Binding MiddleName}"/>

<TextBlock Text="Введите Отчество сотрудника" Grid.Row="2" Grid.Column="0" Height="30" FontSize="25"/>

<TextBox Grid.Row="2" x:Name="TB2" Height="30" Width="200" Grid.Column="1" HorizontalAlignment="Left" Text="{Binding LastName}"/>

<TextBlock Text="Введите Должность сотрудника" Grid.Row="3" Grid.Column="0" Height="30" FontSize="25"/>

<ComboBox Grid.Row="3" x:Name="ComboboX" Height="30" Width="200" Grid.Column="1" HorizontalAlignment="Left" SelectedItem="{Binding Grade}"/>

<TextBlock Text="Введите Пароль сотрудника" Grid.Row="4" Grid.Column="0" Height="30" FontSize="25"/>

<TextBox Grid.Row="4" x:Name="TB4" Height="30" Width="200" Grid.Column="1" HorizontalAlignment="Left" Text="{Binding Password}"/>

<Button Content="Сохранить" Grid.Row="5" Height="30" Width="200" Click="CreateStaff\_Click"/>

<Button Content="Назад" Grid.Row="5" Grid.Column="1" Height="30" Width="100" Click="GoBack" HorizontalAlignment="Right"/>

</Grid>

CreateStaff.xaml.cs

public partial class CreateStaff : Page

{

Staff \_staff = new Staff();

public CreateStaff(Staff \_x)

{

InitializeComponent();

ComboboX.ItemsSource = KursovayaEntities.GetContext().Staffs.Select(\_y => \_y.Grade).Distinct().ToList();

if (\_x == null)

{

\_staff = new Staff();

}

else

{

\_staff = \_x;

TB0.Text = \_staff.FirstName;

TB1.Text = \_staff.MiddleName;

TB2.Text = \_staff.LastName;

ComboboX.SelectedItem = \_staff.Grade;

TB4.Text = \_staff.Password;

}

DataContext = \_staff;

}

private void CreateStaff\_Click(object sender, RoutedEventArgs e)

{

\_staff.FirstName = TB0.Text;

\_staff.MiddleName = TB1.Text;

\_staff.LastName = TB2.Text;

\_staff.Grade = ComboboX.Text;

\_staff.Password = TB4.Text;

if (\_staff.IdStaff == 0)

KursovayaEntities.GetContext().Staffs.Add(\_staff);

try

{

KursovayaEntities.GetContext().SaveChanges();

MessageBox.Show("Данные успешно сохранены!");

ClassPage.MainFrame.Navigate(new StaffPage());

}

catch (Exception)

{

MessageBox.Show("Данные не сохранены!");

}

}

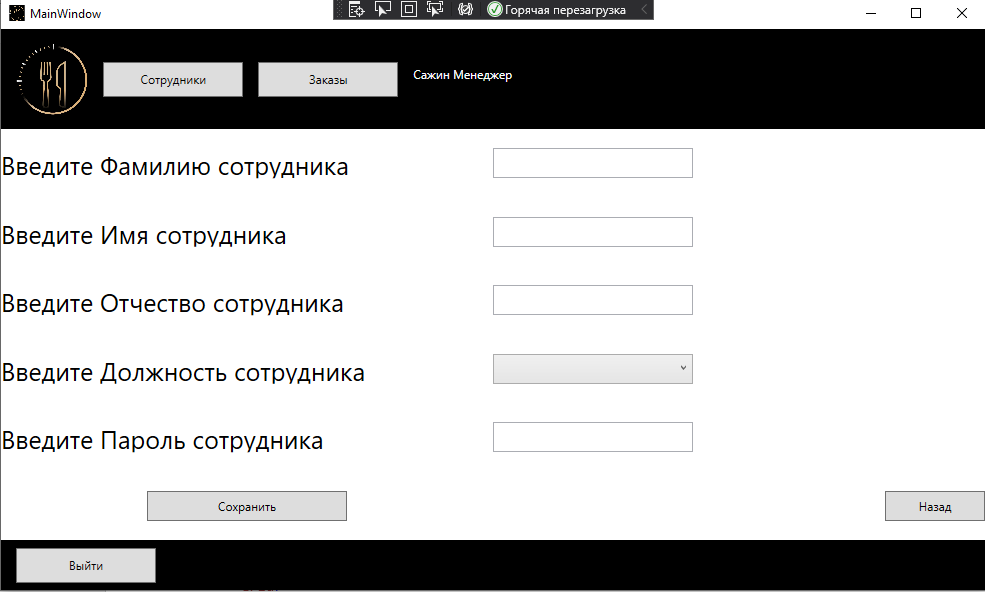
private void GoBack(object sender, RoutedEventArgs e)

{

ClassPage.MainFrame.Navigate(new StaffPage());

}

}



OrdersPage.xaml

Grid>

<DataGrid x:Name="DG" AutoGenerateColumns="False" IsReadOnly="True">

<DataGrid.Columns>

<DataGridTextColumn Header="Номер заказа" Width="Auto" Binding="{Binding IdOrder}"/>

<DataGridTextColumn Header="Сотрудник" Width="Auto" Binding="{Binding Staff.FirstName}"/>

<DataGridTextColumn Header="Сумма Заказа" Width="Auto" Binding="{Binding Path=Price, StringFormat={}{0:C}}"/>

<DataGridTextColumn Header="Статус" Width="Auto" Binding="{Binding Status}"/>

<DataGridTemplateColumn>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="InfoOrder" Content="Посмореть информацию" Width="150" Height="30" Margin="5" Click="InfoOrder\_Click"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

<DataGridTemplateColumn>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button x:Name="DeletedOrder" Content="Удалить Заказ" Width="150" Height="30" Margin="5" Click="DeletedOrder\_Click"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<Button x:Name="CreateInfoOrder" Content="Создать Заказ" Width="125" Height="30" VerticalAlignment="Bottom" Margin="0 0 0 50" Click="CreateInfoOrder\_Click"/>

</Grid>

OrdersPage.xaml.cs

public OrdersPage()

{

InitializeComponent();

var \_data = KursovayaEntities.GetContext().Orders.ToList();

foreach (var i in \_data)

{

i.Price = i.InfoOrders.Select(io => io.Amount).Sum();

}

DG.ItemsSource = \_data;

}

private void InfoOrder\_Click(object sender, RoutedEventArgs e)

{

var \_order = (sender as Button).DataContext as Order;

ClassPage.MainFrame.Navigate(new CrudOrderPage(\_order));

}

private void CreateInfoOrder\_Click(object sender, RoutedEventArgs e)

{

ClassPage.MainFrame.Navigate(new CrudOrderPage(null));

}

private void DeletedOrder\_Click(object sender, RoutedEventArgs e)

{

var \_order = (sender as Button).DataContext as Order;

KursovayaEntities.GetContext().InfoOrders.RemoveRange(\_order.InfoOrders);

KursovayaEntities.GetContext().Orders.Remove(\_order);

KursovayaEntities.GetContext().SaveChanges();

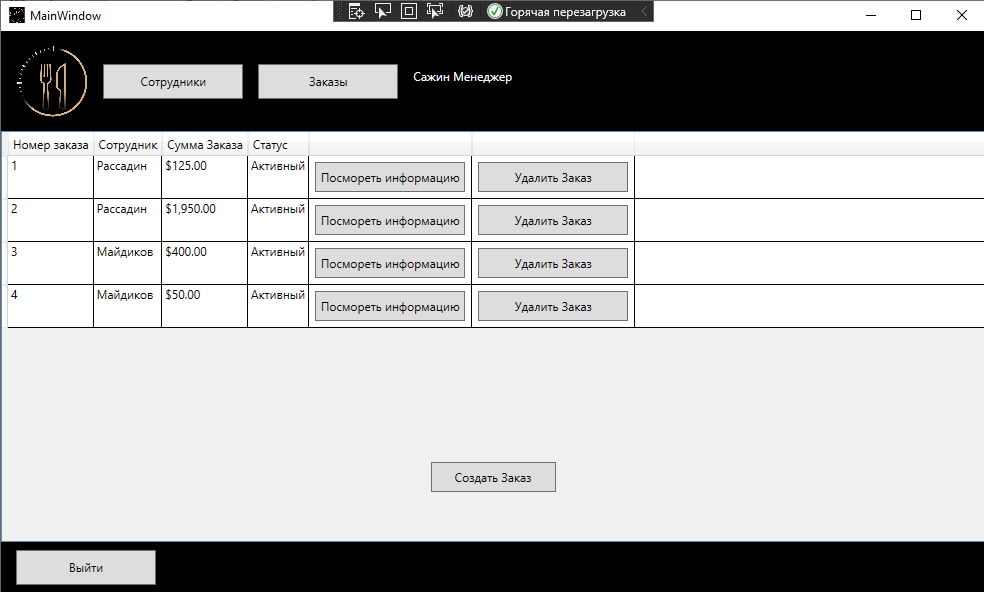
(DG.ItemsSource as ICollection<Order>).Remove(\_order);

DG.Items.Refresh();

MessageBox.Show("Заказ Удален");

}

}



CrudOrderPage.xaml

<Grid>

<StackPanel>

<StackPanel Orientation="Horizontal" VerticalAlignment="Top">

<TextBlock Text="Номер заказа" Margin="10 10 0 0" Height="20" FontWeight="Bold"/>

<TextBlock x:Name="NumberOrder" Margin="5 10 0 0" Width="auto" Height="20" FontWeight="Bold"/>

<TextBlock Text="Стоимость Заказа" Margin="35 10 0 0" Height="20" FontWeight="Bold"/>

<TextBlock x:Name="PriceOrder" Margin="5 10 0 0" Width="auto" Height="20" FontWeight="Bold"/>

</StackPanel>

<DataGrid x:Name="DG0" Margin="0 10 0 10" Height="auto" AutoGenerateColumns="False">

<DataGrid.Columns>

<DataGridTextColumn Header="Блюдо" Binding="{Binding Dish.Title}" Width="auto" />

<DataGridTextColumn Header="Количетсво" Binding="{Binding Quantity}" Width="auto" />

<DataGridTextColumn Header="Сумма" Binding="{Binding Path=Amount, StringFormat={}{0:C}}" Width="auto" />

<DataGridTemplateColumn>

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Content="Удалить блюдо" Margin="5" x:Name="DeleteDish" Click="DeleteDish\_Click" />

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<StackPanel Orientation="Horizontal" VerticalAlignment="Bottom">

<Button Content="Добавить Блюдо" x:Name="CreateDish" Width="150" Height="30" Margin="50 10 0 20" Click="CreateDish\_Click"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<StackPanel Width="Auto" Orientation="Vertical">

<TextBlock Text="Выберете Блюдо" Margin="20"/>

<TextBlock Text="Количество Блюд" Margin="20"/>

</StackPanel>

<StackPanel Width="200">

<ComboBox x:Name="CMDish" Margin="20" DisplayMemberPath="Title"/>

<TextBox x:Name="Kolvo" Margin="20" />

</StackPanel>

</StackPanel>

</StackPanel>

</Grid>

CrudOrderPage.xaml.cs

public partial class CrudOrderPage : Page

{

private Order \_order1;

public CrudOrderPage(Order \_order)

{

InitializeComponent();

CMDish.ItemsSource = KursovayaEntities.GetContext().Dishs.ToList();

CMDish.SelectedIndex = 0;

if (\_order == null)

{

\_order = new Order();

\_order.Price = 0;

\_order.Staff = MainWindow.\_staff;

\_order.Status = "Активный";

}

this.\_order1 = \_order;

DG0.ItemsSource = \_order.InfoOrders;

NumberOrder.Text = Convert.ToString(\_order.IdOrder);

PriceOrder.Text = "$ " + Convert.ToString((int) \_order.Price);

}

private void CreateDish\_Click(object sender, RoutedEventArgs e)

{

var \_infoOrder = new InfoOrder();

\_infoOrder.Dish = CMDish.SelectedItem as Dish;

\_infoOrder.Order = \_order1;

\_infoOrder.Quantity = int.Parse(Kolvo.Text);

\_infoOrder.Amount = \_infoOrder.Quantity \* \_infoOrder.Dish.Price;

\_order1.InfoOrders.Add(\_infoOrder);

DG0.ItemsSource = \_order1.InfoOrders;

DG0.Items.Refresh();

\_order1.Price = \_order1.InfoOrders.Select(io => io.Amount).Sum();

PriceOrder.Text = "$ " + Convert.ToString((int)\_order1.Price);

KursovayaEntities.GetContext().InfoOrders.Add(\_infoOrder);

try

{

KursovayaEntities.GetContext().SaveChanges();

MessageBox.Show("Данные успешно сохранены!");

}

catch (Exception)

{

MessageBox.Show("Данные не сохранены!");

throw;

}

}

private void DeleteDish\_Click(object sender, RoutedEventArgs e)

{

var \_info0 = (sender as Button).DataContext as InfoOrder;

KursovayaEntities.GetContext().InfoOrders.Remove(\_info0);

KursovayaEntities.GetContext().SaveChanges();

(DG0.ItemsSource as ICollection<InfoOrder>).Remove(\_info0);

DG0.Items.Refresh();

\_order1.Price = \_order1.InfoOrders.Select(io => io.Amount).Sum();

PriceOrder.Text = "$ " + Convert.ToString((int)\_order1.Price);

\_order1.InfoOrders.Remove(\_info0);

MessageBox.Show("блюдо удалено!");

}

