

тестового проекта Занятие 3

ИТ-программное решение для бизнеса

Независимый дизайнер тестового проекта: Рамин Мохаммаддуст

Независимый валидатор тестового проекта: Афшин Дехгани



Введение

На этом занятии вы возьмете на себя роль разработчика полного цикла, ответственного за создание пользовательского интерфейса (UI) и серверного АРI для новой программной системы пекарни.

Цель данного заняьтия — воплотить дизайн и требования, изложенные в сессии 2, в функциональное настольное приложение для персонала и надежный API для управления данными о продуктах и заказах. Оценка будет производиться по вашей способности создавать интуитивно понятные и визуально привлекательные пользовательские интерфейсы, а также по вашим навыкам разработки безопасных и эффективных API.

Это занятие предназначено для оценки ваших навыков по следующим направлениям:

- Разработка интерфейсов: Разработка и внедрение пользовательских интерфейсов, которые были бы удобны для пользователя, визуально привлекательны и соответствовали бренду Belle Croissant Lyonnais.
- Серверная разработка: Создание RESTful API, который соответствует лучшим отраслевым практикам, надежно обрабатывает взаимодействие с данными и обеспечивает бесперебойную работу интерфейсного приложения.
- Интеграция: Обеспечение бесперебойной связи и обмена данными между интерфейсным и серверным компонентами.
- Решение проблем: Выявление и устранение технических проблем, возникающих в процессе разработки.

Содержание

Данный учебный пакет содержит следующие материалы:

- 1. **Session Instructions (PDF):** Подробные инструкции с описанием задач, которые необходимо выполнить, и ожидаемых результатов для этого занятия.
- 2. **Common Folder:** Эта папка содержит дополнительные ресурсы, такие как логотип, значки, руководство по стилю и другие элементы дизайна Belle Croissant Lyonnais, которые могут быть использованы при разработке приложения.
- 3. **Database Schema (SQL):** SQL-скрипт, содержащий структуру таблиц Promotions и LoyaltyProgram, которые вы будете использовать на данном занятии.

Описание проекта и задач

На этом занятии вы создадите основу для настольного приложения Belle Croissant Lyonnais.

Методические рекомендации:

- 1. Простота в использовании: Сделайте интерфейс простым и понятным для персонала.
- 2. **Привлекательный вид:** При выборе дизайна следуйте руководству по стилю Belle Croissant Lyonnais.
- 3. Корректная работа: Убедитесь, что все части приложения работают корректно и без ошибок.



- 4. **Безопасность:** Защитите данные заказчиков и убедитесь, что приложение безопасно в использовании.
- 5. Своевременность: Выполните все задания в установленный срок.

Технические особенности:

- 1. **Настройка базы данных:** Создайте и заполните базу данных в соответствии с предоставленной схемой.
- 2. **Разработка API:** Внедрите RESTful API, который соответствует лучшим практикам и включает аутентификацию.
- 3. **Пользовательский интерфейс:** Разработайте интуитивно понятные экраны для управления продуктами и заказами.
- 4. **Проверка данных:** Убедитесь, что пользователь вводит правильные и полные данные для всех операций.
- 5. **Обработка ошибок:** Отображение четких сообщений пользователю при возникновении каких-либо проблем.

Дополнительные факторы:

- Приложение должно работать плавно и быстро в предоставленной среде разработки.
- Используйте четкие надписи и инструкции для всех элементов UI.
- Организуйте информацию таким образом, чтобы сотрудникам было легко ее воспринимать.
- Рассмотрите крайние случаи и потенциальные ошибки, возникающие во время ввода данных пользователем и их обработки.
- Внедрите комплексные модульные тесты для обеспечения функциональности API.

Инструкции для участника

3.1 Настройка базы данных и импорт данных

Цель:

Создать и заполнить базу данных Belle Croissant Lyonnais в соответствии с предоставленной схемой, гарантируя точность и целостность данных.

Задачи:

- 1. Создание базы данных:
 - Создайте базу данных с именем Bellecroissantly Nonnais (или похожим, подходящим названием).

2. Выполнение схемы:

- Выполните предоставленный SQL-скрипт (Database_Schema.sql), чтобы создать следующие таблицы и их взаимосвязи:
 - Products



- Customers
- Orders
- OrderItems

3. Импорт данных:

 Импортируйте данные из очищенных CSV-файлов (products_cleaned.csv, customers_cleaned.csv и sales_transactions_cleaned.csv) в соответствующие таблицы базы данных. Убедитесь, что типы данных соответствуют определению схемы.

Результаты:

• Учетные данные базы данных: Укажите строку подключения или учетные данные, необходимые для доступа к вашей базе данных (имя сервера, имя базы данных, имя пользователя, пароль). Вы можете сохранить эту информацию в простом текстовом файле с именем Session3_DatabaseCredentials.txt.

Дополнительные примечания:

- Обратите внимание на типы данных в CSV-файлах и убедитесь, что они соответствуют определенным в схеме типам данных.
- Используйте соответствующие инструменты или методы импорта для эффективной передачи данных в базу данных.
- Перепроверьте данные после импорта, чтобы убедиться в их точности и полноте.
- Оценка будет сосредоточена на правильности и полноте настройки базы данных и процесса импорта данных.

3.2 Разработка серверного АРІ

Цель:

Разработать безопасный и эффективный RESTful API, используя .NET Web API, который позволяет настольному приложению Belle Croissant Lyonnais взаимодействовать с базой данных.

Задачи:

- 1. Конечные точки API: Создайте следующие способы взаимодействия приложения с базой данных:
 - о Продукты:
 - GET /api/products: Получить все продукты
 - GET /api/products/{id}: Получить один продукт по идентификатору
 - POST /api/products: Добавить новый продукт
 - PUT /api/products/{id}}: Изменить продукт по идентификатору
 - DELETE /api/products/{id}: Удалить продукт по идентификатору

о Заказчики:

- GET /api/customers: Получить доступ ко всем заказчикам
- GET /api/customers/{id}: Получить одного заказчика по идентификатору
- POST /api/customers: Добавить нового заказчика



• PUT /api/customers/{id}: Изменить заказчика по идентификатору

о Заказы:

- GET /api/orders: Получать все заказы
- GET /api/orders/{id}: Получить один заказ по идентификатору
- POST /api/orders: Добавить новый заказ
- PUT /api/orders/{id}/complete: Завершить заказ по идентификатору
- PUT /api/orders/{id}/cancel: Отменить заказ по идентификатору

2. Проверка данных:

- Убедитесь, что приложение отправляет в базу данных нужные данные (цифры, буквы, даты и т.д.).
- о Очистите данные, чтобы убедиться в их безопасности в использовании.

3. Обработка ошибок:

о При возникновении проблем отправляйте правильные сообщения обратно в приложение.

4. Идентификация:

 Убедитесь, что только нужные пользователи имеют доступ к API. Используйте систему паролей (базовая аутентификация), в которой имя пользователя - "staff", а пароль -"BCLyon2024".

Результаты:

- Запуск API: Заставьте API работать в заданном месте.
- Session3_API_Endpoints.txt: Файл с описанием возможных действий в API, методов запроса какихлибо данных и возможных ответов (в формате JSON).

3.3 Разработка UI — Управление продуктом

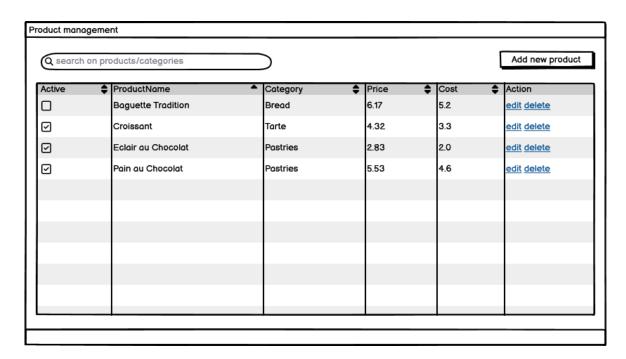
Цель:

Разработать удобный интерфейс для управления продуктами в настольном приложении Belle Croissant Lyonnais с использованием конечных точек API, созданных в Задаче 3.2.

Задачи:

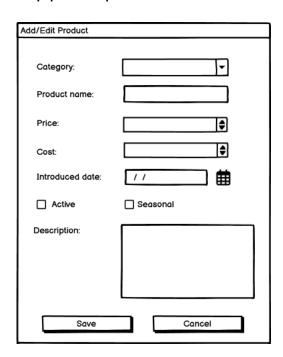
1. Список продуктов:





- о Отобразите в таблице все продукты, полученные из конечной точки API GET /api/products.
- о Столбцы таблицы: ProductName, Category, Price, Cost, Active (Yes/No).
- о Добавьте сортировку (по возрастанию/по убыванию) для каждого столбца.
- о Добавьте строку поиска для фильтрации товаров по названию или категории.

2. Добавление/редактирование формы товара:



- о Создайте форму для добавления новых продуктов или редактирования существующих.
- Поля формы:



- ProductName (требуется ввести текст, не более 100 символов)
- Category (выпадающий список, обязательно, значения из файла products_cleaned.csv)
- Price (требуется ввести числовое значение, оно должно быть положительным)
- Cost (требуется ввести числовое значение; оно должно быть положительным и меньше цены)
- Description (текстовое поле, необязательно)
- Seasonal (флажок)
- Active (флажок)
- IntroducedDate (средство выбора даты, обязательно)
- В форме должны быть кнопки "Save" и "Cancel".
- Проверьте правильность входных данных.
- Используйте API для следующих операций:
 - POST /api/products для сохранения новых продуктов.
 - PUT /api/products/{id} для обновления существующих продуктов.
 - Обрабатывайте ответы API и отображайте соответствующие сообщения (об успешном завершении/ошибке) для пользователя.

3. Удаление функциональности продукта:

- Добавьте кнопку "Delete" рядом с каждым товаром в списке.
- о Отобразите диалоговое окно подтверждения перед удалением.
- о Используйте API (DELETE /api/products/{id}) для удаления продукта.
- Обработайте ответ API и соответствующим образом обновите список продуктов.

Результаты:

Включите пользовательский интерфейс в существующий файл Session3_DesktopApp.exe.

Дополнительные примечания:

- Следуйте рекомендациям по проектированию и разработке пользовательского интерфейса.
- UI должен быть чувствительным и адаптироваться к различным размерам экрана.
- Учитывайте обработку ошибок и отзывы пользователей в пользовательском интерфейсе.

3.4 Разработка UI — Управление заказами

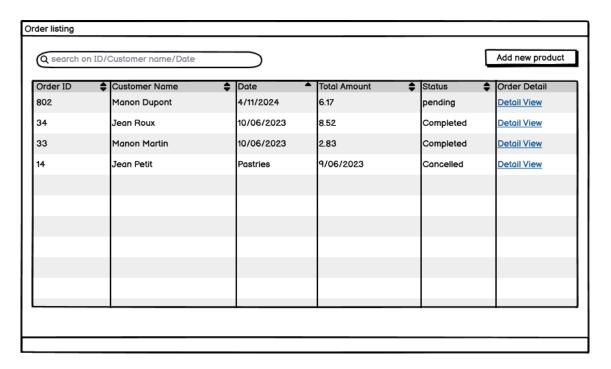
Цель:

Разработка удобного интерфейса для управления заказами заказчиков в настольном приложении Belle Croissant Lyonnais с использованием конечных точек API, созданных в Задаче 3.2.

Задачи:

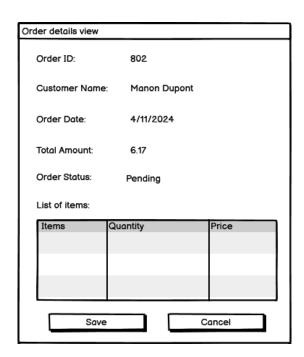
1. Список заказов:





- о Отобразите в таблице все заказы из базы данных.
- Столбцы таблицы: Order ID, Customer Name, Date, Total Amount, Status (Pending, Processing, Completed, Cancelled).
- о Разрешите сортировку (по возрастанию/убыванию) по каждому столбцу.
- Добавьте строку поиска для фильтрации заказов по идентификатору, имени заказчика или дате.

2. Просмотр сведений о заказе:





- Создайте отдельное окно или раздел для отображения подробной информации о выбранном заказе.
- Отобразите следующую информацию:
 - Идентификатор заказа
 - ФИО заказчика
 - Дата и время заказа
 - Общая сумма
 - Статус заказа
 - Список заказанных товаров с указанием количества и цены

3. Обновление статуса заказа:

- о Добавьте кнопки или выпадающее меню для обновления статуса заказа.
- Доступные статусы: "Processing", "Completed", "Cancelled".
- о Используйте АРІ для обновления статуса заказа в базе данных.

Результаты:

Включите пользовательский интерфейс в существующий файл Session3_DesktopApp.exe.

Дополнительные примечания:

- Используйте предпочитаемую вами платформу пользовательского интерфейса .NET.
- UI должен быть чувствительным и адаптироваться к различным размерам экрана.
- Большое значение имеют обработка ошибок и обратная связь с пользователями (например, отображение сообщений при успешном обновлении заказа или возникновении ошибки).

3.5 Приемка и тестирование в режиме "черного ящика"

Цель

Разработать приемочные тесты для проверки функциональности API и обеспечения соответствия указанным требованиям. Провести тестирование в режиме "черного ящика" для проверки поведения API без знания внутренней работы.

Задачи

- 1. Настройка среды тестирования:
 - Убедитесь, что приложение Flask запущено локально.
 - Используйте предоставленный скрипт test_backend.py для запуска тестов.

2. Приемочные тесты:

- Конечные точки продукта:
 - Убедитесь, что GET /api/products открывает список всех продуктов.
 - Убедитесь, что GET /api/products/{id} открывает правильные сведения о продукте.



- Убедитесь, что POST /api/products создает новый продукт с действительными данными.
- Убедитесь, что PUT /api/products/{id} обновляет существующий продукт.
- Убедитесь, что при удалении /api/products/{id} указанный продукт был удален.

• Конечные точки заказа:

- Убедитесь, что GET /api/orders возвращает список всех заказов.
- Убедитесь, что GET /api/orders/{id} возвращает правильные данные о заказе.
- Убедитесь, что POST /api/orders создает новый заказ с действительными данными.
- Убедитесь, что PUT /api/orders/{id}/complete помечает заказ как выполненный.
- Убедитесь, что PUT /api/orders/{id}/cancel отменяет заказ и восстанавливает запасы.

3. Тестирование в режиме "черного ящика":

- Протестируйте API, не зная внутренней структуры кода.
- Сосредоточьтесь на проверке ввода-вывода и обработке ошибок.
- Используйте различные входные данные для проверки крайних случаев и недопустимых сценариев.

Результаты

- Имя файла: Session3_AcceptanceTests.zip
- Тип файла: Сжатый архив (.zip)
- Содержание:
 - Все тестовые файлы (.cs-файлы для .NET) для приемки и решения для тестирования в режиме "черного ящика".
 - Любые необходимые конфигурационные файлы.

Дополнительные примечания

- Используйте методы имитационного моделирования или прерывания для имитации внешних зависимостей.
- Убедитесь, что тесты выполняются изолированно и не зависят от внешних ресурсов.
- Для получения рекомендаций по написанию эффективных тестов обратитесь к документу "Web API Design Best Practices".



3.6 Модульное тестирование и тестированием методом "белого ящика"

Цель:

Целью данного модуля является разработка и тестирование системы управления запасами хлебобулочных изделий с использованием модульных методов тестирования и методов "белого ящика". Участники будут внедрять классы для управления хлебобулочными изделиями и запасами, разрабатывать комплексные тестовые примеры, а также обеспечивать качество кода и функциональность с помощью тщательного тестирования.

Задачи

1. Внедрение класса Bakervitem:

- Создайте класс с именем Bakeryltem со свойствами для названия, цены, количества и даты истечения срока действия.
- Реализуйте метод IsExpired(), чтобы определить, истек ли срок действия элемента.

2. Внедрение класса BakeryInventory:

- Создайте класс с именем BakeryInventory для управления коллекцией объектов BakeryItem.
- Внедрение методов для следующих операций:
 - AddItem(объект BakeryItem): Добавляет новый товар в товарный запас.
 - RemoveItems(название строки): Удаляет элемент или элементы из товарного запаса по имени.
 - GetTotalValue(): Вычисляет общую стоимость (цена * количество) всех товаров в товарном запасе.

3. Написание модульных тестов:

- Используйте NUnit для написания модульных тестов для обоих классов. Проверьте выполнение следующих требований:
 - Свойства Bakeryltem заданы и извлечены правильно.
 - Метод IsExpired точно определяет, истек ли срок действия единицы.
 - Методы BakeryInventory корректно добавляют, удаляют и рассчитывают общую стоимость продуктов.
 - Обработайте пограничные случаи, такие как попытка удалить несуществующий элемент.

4. Проведите тестирование методом "белого ящика":

- Проанализируйте код, чтобы определить критические пути и потенциальные области риска.
- Убедитесь, что тестовые примеры охватывают все возможные пути выполнения.

Результаты

Имя файла: BakeryInventoryTesting.zip



- Тип файла: Сжатый архив (.zip)
- Содержание:
 - Файлы классов C# для BakeryItem и BakeryInventory.
 - Тестовый проект с использованием NUnit для тестирования обоих классов.

Дополнительные примечания

- Убедитесь, что ваш код хорошо документирован с добавлением объясняющими логику комментариев.
- Используйте лучшие практики в отношении соглашений об именовании и организации кода.
- Перед отправкой убедитесь, что все модульные тесты успешно пройдены.