ОБЩИЕ ТРЕБОВАНИЯ И РЕКОМЕНДАЦИИ

Общие требования

Не позволяйте пользователю вводить некорректные значения в текстовые поля сущностей. Например, в случае несоответствия типа данных поля введенному значению. Оповестите пользователя о совершенной им ошибке.

Уведомляйте пользователя о совершаемых им ошибках или о совершении запрещенных в рамках задания действиях.

Запрещено удаление сущностей, которое приведет к нарушению ограничений связей. Например, запрещено удалять клиента, который связан с предложением или потребностью.

При удалении любых сущностей необходимо либо спрашивать подтверждение пользователя, либо реализовать возможность отмены операции удаления.

При возникновении непредвиденной ошибки приложение не должно аварийно завершать работу.

Визуальные компоненты должны соответствовать руководству по стилю. Используйте пиктограмму компании в качестве пиктограмм всех форм.

Структура проекта

Каждая сущность должна быть представлена в программе как минимум одним отдельным классом.

Для работы с разными сущностями используйте разные формы, где это уместно.

Проектирование схемы БД

Каждая сущность должна породить как минимум одну таблицу в базе данных. Храните поля сущностей в подходящих типах данных.

Добавьте ограничения (CHECK CONSTRAINT) которые отражают специфику предметной области.

Добавьте ограничения связности (FOREIGN KEY) между сущностями.

Оформление кода

1. Идентификаторы должны соответствовать стилю CamelCase.
2. Максимальная длина строки - 80 символов.
3. Отступ составляет 4 пробела (без tab).
4. Используйте комментарии для пояснения неочевидных фрагментов кода. Запрещено комментирование кода.
5. Допустимо использование не более одной команды в строке.

Хороший код воспринимается как обычный текст. Не используйте комментарии для пояснения

очевидных действий.

Рекомендации

Реализуйте переиспользуемые визуальные компоненты. Не дублируйте логику – это отнимет у вас много времени. Например, вам понадобится текстовое поле с валидацией целочисленного значения много более чем в одном месте. Другой пример: с каждой сущностью нужно осуществлять 4 одинаковых действия: чтение, обновление, создание и удаление.

Где уместно используйте наследование и полиморфизм. В рамках данного задания явно прослеживаются несколько иерархий классов.