Требования и рекомендации

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.

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, полиморфизм и инкапсуляцию.

ФАЙЛОВАЯ СТРУКТУРА

Файловая структура должна отражать логическое разделение разных компонентов проекта. Например, все формы содержатся в одной директории, пользовательские визуальные компоненты – в другой, классы сущностей – в третьей.

ЛОГИЧЕСКАЯ СТРУКТУРА

Логика представления (работа с пользовательским вводом/выводом, формы, обработка событий) не должна быть перемешана с бизнес-логикой (ограничения и требования, сформулированные в заданиях), а также не должна быть перемешана с логикой доступа к базе данных (SQL-запросы, запись, получение данных). В идеале это должны быть три независимых модуля.

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.

РУКОВОДСТВО ПО СТИЛЮ

Визуальные компоненты должны соответствовать руководству по стилю. Используйте предоставленный логотип в качестве логотипа на всех формах.

ОБРАБОТКА ОШИБОК

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 с соответствующими пиктограммами окна сообщения.

При возникновении непредвиденной ошибки приложение не должно аварийно завершать работу.

УДАЛЕНИЕ

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

Пользователь системы должен обязательно подтвердить любую операцию удаления.

Операции удаления сущностей следует реализовать как обновление одного из атрибутов сущности. То есть фактически удалять данные нельзя.

Удаленные сущности по умолчанию не отображаются.

ОФОРМЛЕНИЕ КОДА

Классы должны быть небольшими, понятными и выполнять одну единственную функцию (Single responsibility principle).

Идентификаторы переменных, методов и классов должны отражать суть и/или цель их использования.

1. Идентификаторы должны соответствовать стилю CamelCase (для C# и Java) и snake_case (для Python).

2. Максимальная длина строки - 80 символов.

3. Используйте комментарии для пояснения неочевидных фрагментов кода. Запрещено комментирование кода.

4. Допустимо использование не более одной команды в строке.

Хороший код воспринимается как обычный текст. Не используйте комментарии для пояснения очевидных действий. Комментарии должны присутствовать только в местах, которые требуют дополнительного пояснения.

ПРЕДОСТАВЛЕНИЕ РЕЗУЛЬТАТОВ

Все практические результаты должны быть переданы заказчику путем загрузки файлов на предоставленный вам репозиторий системы контроля версий git. Практическими результатами являются исходный код приложения (в виде коммита текущей версии проекта, но не архивом), исполняемые файлы и графические/текстовые файлы.

Результаты работы каждой сессии должны быть загружены в отдельную ветку с названием «Сессия X» (X – номер сессии).

Для оценки работы будет учитываться только содержимое репозитория. При оценке рассматриваются заметки только в электронном виде (readme.md). Рукописные примечания не будут использоваться для оценки.