### Общие требования

Название приложения

Используйте соответствующие названия для ваших приложений и файлов. Так, например, наименование настольного приложения должно обязательно включать название компании-заказчика.

Файловая структура

Файловая структура проекта должна отражать логику, заложенную в приложение. Например, все формы содержатся в одной директории, пользовательские визуальные компоненты – в другой, классы сущностей – в третьей.

Структура проекта

Каждая сущность должна быть представлена в программе как минимум одним отдельным классом. Классы должны быть небольшими, понятными и выполнять одну единственную функцию (Single responsibility principle).

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

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.

Руководство по стилю

Визуальные компоненты должны соответствовать руководству по стилю, предоставленному в качестве ресурсов к заданию в соответствующем файле. Обеспечьте соблюдение требований всех компонентов в следующих областях:

• цветовая схема,

• размещение логотипа,

• использование шрифтов,

• установка иконки приложения.

Макет и технические характеристики

Все компоненты системы должны иметь единый согласованный внешний вид, соответствующий руководству по стилю, а также следующим требованиям:

• разметка и дизайн (предпочтение отдается масштабируемой компоновке; должно присутствовать ограничение на минимальный размер окна; должна присутствовать возможность изменения размеров окна, где это необходимо; увеличение размеров окна должно увеличивать размер контентной части, например, таблицы с данными из БД);

• группировка элементов (в логические категории);

• использование соответствующих элементов управления (например, выпадающих списков для отображения подстановочных значений из базы данных);

• расположение и выравнивание элементов (метки, поля для ввода и т.д.);

• последовательный переход фокуса по элементам интерфейса (по нажатию клавиши TAB);

• общая компоновка логична, понятна и проста в использовании;

• последовательный пользовательский интерфейс, позволяющий перемещаться между существующими окнами в приложении (в том числе обратно, например, с помощью кнопки «Назад»);

• соответствующий заголовок на каждом окне приложения (не должно быть значений по умолчанию типа MainWindow, Form1 и тп).

Все предоставленные макеты являются лишь примером и рекомендациями к расположению элементов – нет необходимости следовать им «попиксельно».

Обратная связь с пользователем

Уведомляйте пользователя о совершаемых им ошибках или о запрещенных в рамках задания действиях, запрашивайте подтверждение перед удалением, предупреждайте о неотвратимых операциях, информируйте об отсутствии результатов поиска и т.п. Окна сообщений соответствующих типов (например, ошибка, предупреждение, информация) должны отображаться с соответствующим заголовком и пиктограммой. Текст сообщения должен быть полезным и информативным, содержать полную информацию о совершенных ошибках пользователя и порядок действий для их исправления. Также можно использовать визуальные подсказки для пользователя при вводе данных.

Обработка ошибок

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

Обратите внимание на использование абсолютных и относительных путей к изображениям. Приложение должно корректно работать в том числе и при перемещении папки с исполняемым файлом.

При возникновении непредвиденной ошибки приложение не должно аварийно завершать работу.

Оформление кода

Идентификаторы переменных, методов и классов должны отражать суть и/или цель их использования, в том числе и наименования элементов управления (например, не должно быть значений по умолчанию типа Form1, button3).

Идентификаторы должны соответствовать соглашению об именовании (Code Convention) и стилю CamelCase (для C#, Java и 1С) и snake\_case (для Python).

Допустимо использование не более одной команды в строке.

Комментарии

Используйте комментарии для пояснения неочевидных фрагментов кода.

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

Используйте тип комментариев, который в дальнейшем позволит сгенерировать XML-документацию, с соответствующими тегами (например, param, return(s), summary и др.)

Оценка

Каждая задача оценивается путем тестирования реализации требуемого функционала. Так как требования к реализуемой системе очень высоки, возможно, будут использоваться средства для автоматизированного тестирования приложения. В связи с этим, в ходе разработки, может возникнуть необходимость следовать определенным правилам именования и структурирования проекта.

Предоставление результатов

Все практические результаты должны быть переданы заказчику путем загрузки файлов в предоставленную систему контроля версий git. Для каждой сессии могут быть выданы специфические инструкции по сохранению каждого модуля. Практическими результатами являются

• исходный код приложения (в виде коммита текущей версии проекта, но не архивом),

• прочие графические/текстовые файлы.

При создании коммитов используйте содержательные и понятные комментарии.

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.

Репозиторий обязательно должен содержать описание в формате Markdown (см. шаблон в файле README-Template.md или README-Template\_rus.md). Заполните также дополнительную информацию о проекте и способе запуска приложения в файле readme.md.