**ТРЕБОВАНИЯ И РЕКОМЕНДАЦИИ**

**Введение**

Для выполнения задач вы можете использовать любые инструменты, предоставляемые согласно инфраструктурному листу.

В случае нехватки времени для выполнения всех оставшихся задач вы можете пропускать выполнение некоторых задач в пользу других. Однако ожидается, что вы предоставите максимально завершенную работу в конце каждой сессии, чтобы облегчить оценку вашей работы.

**Правила**

Во время чемпионата необходимо соблюдать следующие правила:

* Запрещен доступ в Интернет (кроме одноразового доступа в течение сессии не более 15 минут за всю сессию);
* Запрещено использование любых гаджетов (мобильный телефон, планшет, смарт-часы, шлемы виртуальной реальности ит.д.);
* Запрещено использование ваших собственных устройств хранения данных (USB-накопители, жесткие диски и т.д.);
* Запрещено общение с другими участниками и экспертами чемпионата;
* Запрещено приносить на экзамен книги, заметки и т.д.;
* Разрешено использовать личные устройства ввода информации (клавиатура, мышь, трекбол и т.д.), но эти устройства должны быть проводными, непрограммируемыми и должны работать без дополнительной установки драйверов (эти требования предварительно проверяются техническим экспертом).

Несоблюдение этих правил может привести к удалению с площадки проведения.

**Название приложения**

Используйте соответствующие названия для ваших приложений и файлов.

**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.

**Руководство по стилю**

Визуальные компоненты должны соответствовать макету. Обеспечьте соблюдение требований всех компонентов в следующих областях:

* цветовая схема,
* размещение логотипа,
* установка иконки приложения.

|  |  |
| --- | --- |
| **Основные цвета** | |
| Фирменный зелёный | #78B24B |
| Ярко-зеленый | #89FC43 |
| Желтый | #F8FC43 |
| Красный | #FC4343 |

**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, datepicker для дат и тп);
* 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).

У всех диаграмм и графиков приложения должна быть добавлена легенда.

Для оформления кнопок-действий пользователя можно использовать смайлики (emoji) из Windows (Win + .).  Например, использовать смайлик карандашика, сохранения и тд.

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 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 путем загрузки файлов в предоставленную систему контроля версий git. Для сохранения данных по результатам проектирования (диаграммы, текстовые документы или сценарии тестирования) создайте репозиторий Designing и загружайте туда файлы с каждой сессии. Для  сохранения исходного кода проекта(-ов) вы можете создавать дополнительные необходимые репозитории.

Практическими результатами работы 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\_rus.md). Заполните дополнительную информацию о проекте и способе запуска приложения в файле readme.md, а также о выполненной работе в рамках каждой сессии.

Обратите внимание, что дополнительного времени после окончания сессии на сохранение не предусмотрено, поэтому будьте бдительны и загружайте результаты работ своевременно в рамках сессии.

**Взаимодействие через API**

Разрабатываемый продукт должен быть реализован с помощью двух разных проектов: клиентская и серверная части. Серверная часть должна быть напрямую связана с базой данных, а также содержать в себе нужные конечные точки (endpoints) для запросов с клиентской части посредством HTTP. Клиентское приложение, в свою очередь, не должно быть связано с базой данных напрямую: получать и отправлять нужные данные следует через запросы в серверной части. Взаимодействие через HTTP должно быть осуществлено следуя архитектуре REST.

Клиентская часть при работе с API должна учитывать возможные проблемы с интернет-соединением, а также прочие ошибки на стороне сервера. В случае ошибок при запросах к серверу, приложение должно информировать о них пользователя, а также дополнительно указывать техническую информацию об ошибке.

Все методы серверной части, по мере их реализации в рамках всех сессий, должны быть задокументированы с помощью создания коллекции в приложении Postman и содержать в себе описание передаваемых параметров, а также примеры ответов от сервера (examples). В случае если вы не реализовали нужные методы на стороне сервера, то их необходимо спроектировать и добавить в postman-коллекцию.

**Обновление UI без перезагрузки**

В случае если пользователь меняет какие-либо данные в клиентском приложении, а измененные данные отображаются в какой-либо дополнительной части приложения, то необходимо, чтобы интерфейс приложения обновлялся везде. Например, если меняется иконка/название проекта, то после сохранения без перезагрузки приложения должен обновиться и список проектов (в меню).