Перед нашей командой стояла задача - необходимо выработать интересную идею для реализации как курсовой работы. При этом она должна быть не слишком сложна и не слишком проста, так как было желание научиться чему-то новому. Так же мы хотели учесть практическое применение нашего приложения, хотели сделать так, чтобы мы могли пользоваться им сами и рекомендовать другим.

Отбросив большое количество вариантов от игр до гороскопов, мы остановились на списке задач.

Мы решили, что это будет отличный проект для того, чтобы и научиться чему-то новому, и в то же время оставаться заинтересованными в разработке. Этот проект имеет четкое практическое применение, его легко показать на оценку другим людям и самим оценивать результат своей работы. На начальных этапах разработки он не так сложен, но в то же время имеет большой потенциал для развития как приложения, а значит и для нас самих как разработчиков.

Многие люди сталкиваются с проблемой нехватки времени и ощущением того, что всё валится из рук. В такой ситуации на помощь приходят приложения для организации жизни, например, списки задач (или таск-трекеры), которые помогают структурировать свои задачи и отслеживать их выполнение.

Список задач — программа для управления проектами. Такое приложение поможет организовать и структурировать свою работу. Её функционал позволяет пользователю следить за рабочими процессами и контролировать выполнение поставленных задач.

*Цель проекта*: создать удобный список задач, в котором можно создавать уникальные задачи и группировать их по спискам с возможностью отслеживать статус их выполнения.

Мы постарались организовать работу в команде как можно более эффективно, распределив роли, но в то же время помогая друг другу с выполнением смежных задач. Я изучил гибкую методологию разработки SCRUM и постарался внедрить её приемы в работу команды. На еженедельных встречах мы обсуждали, что было сделано за ушедшую неделю и что мы хотим сделать на следующей неделе, попутно собирая нашу статистику по выполненным задачам.

Для совместной удалённой работы с кодом мы использовали сервис по управлению сборками GitHub. На этом же сервисе удобно реализовано средство по ведению проектов.

При разработке мы придерживались правил качественного программирования:

* Декомпозиция
* Модульность
* Корректность
* Тестируемость
* Устойчивость
* Стилизованность
* Читаемость
* Универсальность

А также выполнения концепций Объектно-Ориентированного Программирования:

* Абстракция – представление объекта минимальным набором полей и методов с достаточной точностью
* Наследование – классы-наследники наследуют функциональность некоторого существующего типа
* Полиморфизм – несколько реализаций одной и той же сущности
* Инкапсуляция - изоляции контрактных обязательств абстракции от их реализации.

Наш проект оперирует несколькими понятиями сущностей и обладает списком возможностей:

*Сущности:*

* Карточка дела (карточка, карта) - набор характеристик карточки, который включает в себя краткую постановку задачи (название), описание задачи, дату последнего обновления, отметку о выполнении.

- Отметка о важности. Выделяет задачу среди всех других. Позволяет пользователю определить приоритетные для себя дела.

- Отметка о выполнении. Определяет статус задачи: В работе или выполнена

* Список карт (список, список задач) - карты, сгруппированные пользователем в отдельный список. Позволяет хранить несколько карт, сортировать их по различным параметрам, осуществлять поиск карт по части названия или описания.
* Список списков карт (менеджер списков) - список, содержащий все списки карт. Позволяет хранить списки карт, осуществлять поиск по части названия или описания по всем содержащимся в нём спискам.

*Возможности:*

* Добавление/Удаление карт, списков
* Выполнение задачи(карты). Отметить задачу как выполненную. Есть возможность просмотреть все выполненные задачи в выбранном списке.
* Отметка о важности задачи(карты). Отметить задачу как важную. Такая задача отображается в списке выше всех остальных
* Сортировка карт. Есть возможность сортировать карты в разном порядке и по разным параметрам: по дате, по степени выполнения и по важности.
* Просмотр выполненных задач. Возможность посмотреть, какие задачи были выполнены ранее в выбранном списке.
* Поиск задачи(карты) по части названия или дате.

**2.2. Примеры задач, решаемых проектом**

1. Отслеживание дел по дому (уборка, покупки, подарки близким и т.п.)
2. Отслеживание рабочих проектов (Формирование списка чётких задач, алгоритма, которому можно следовать и поэтапно выполнять поставленную задачу)
3. Постановка глобальных целей (научиться программировать, закончить университет, изучить французский язык ) и отслеживание пути к их выполнению (добавление тематических задач: записаться на языковые курсы и т.п.)

\*Это не значит, что вовсе не можем использовать события и событийную модель, однако везде, где возможно, вместо событий следует использовать команды. В WPF команды представлены интерфейсом. Однако WPF имеет в качестве реализации этого интерфейса класс System.Windows.Input.RoutedCommand, который ограничен по функциональности. Поэтому, как правило, придется реализовывать свои собственные команды с помощью реализации ICommand.

Возможность отделить графический интерфейс от логики приложения, благодаря чему над разными частями приложения могут относительно автономно работать разные специалисты: над интерфейсом - дизайнеры, над кодом логики - программисты.

Декларати́вное программи́рование — парадигма программирования, в которой задаётся спецификация решения задачи, то есть описывается ожидаемый результат, а не способ его получения. Противоположностью декларативного является императивное программирование, при котором на том или ином уровне детализации требуется описание последовательности шагов для решения задачи. В качестве примеров декларативных языков обычно приводят HTML и SQL.

Пишешь то что хочешь получить а не то как это получить