минобрнауки россии

федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |  |
| --- | --- |
| Институт (факультет) | Институт Информационных Технологий |
| Кафедра | Математического и Программного Обеспечения ЭВМ |

**КУРСОВАЯ РАБОТА**

|  |
| --- |
| по дисциплине C#-программирование |

|  |  |
| --- | --- |
| на тему | Разработка программного обеспечения на языке C# |

|  |
| --- |
| Выполнил студент группы |
| 1ПИб-02-3оп-22 |
| направление подготовки (специальности) |
| 09.03.04, Программная инженерия |
| *шифр, наименование* |
| Маркелов Сергей Александрович |
| *фамилия, имя, отчество* |

|  |
| --- |
| Руководитель |
| Шаханов Никита Иванович |
| *фамилия, имя, отчество* |
| старший преподаватель |
| *должность* |

|  |
| --- |
| Дата представления работы |
| «\_\_11\_\_»\_\_\_\_\_\_\_июня \_\_\_\_\_2025 г. |
|  |
| Заключение о допуске к защите |
|  |
|  |
|  |
|  |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| количество баллов |
| Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Череповец, 2025

*Год*

Оглавление

[Введение 3](#_Toc200495307)

[1. Изучение и описание предметной области 5](#_Toc200495308)

[2. Постановка задачи 8](#_Toc200495309)

[3. Логическое проектирование 11](#_Toc200495310)

[4. Физическое проектирование 15](#_Toc200495311)

[5. Проектирование интерфейса 16](#_Toc200495312)

[6. Тестирование программы 18](#_Toc200495313)

[Заключение 22](#_Toc200495314)

[Список литературы 23](#_Toc200495315)

[Приложение 1. Техническое задание 25](#_Toc200495316)

[Приложение 2. Текст программы 33](#_Toc200495317)

[Приложение 3. Руководство пользователя 58](#_Toc200495318)

# Введение

Программы играют огромную роль в современном мире. Существует большое количество различных видов программ: драйвера, утилиты, плееры, браузеры, различные редакторы (текстовые, графические, музыкальные видеоредакторы), мессенджеры, архиваторы, антивирусы и многие другие.

Благодаря программам мы можем смотреть, обрабатывать или создавать фотографии, видео, музыку, работать с документами, обрабатывать данные и многое другое.

Один из видов программ – планировщики задач, или по-другому task-менеджеры. Task-менеджер – это приложение или система для планирования, отслеживания и управления задачами. Он помогает организовать рабочий процесс, контролировать выполнение дел и улучшать личную или командную продуктивность.

Программы создаются благодаря языкам программирования – формальным знаковым системам, предназначенным для описания алгоритмов обработки данных. Каждый язык программирования имеет свой алфавит и синтаксис [4].

C# – это объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он предназначен для создания приложений под платформу .NET и поддерживает современный синтаксис, аналогичный C++ и Java [9].

C# используется для разработки как десктопных, так и web-приложений, мобильных программ, игр (например, в Unity), а также облачных и корпоративных решений.

Язык C# предоставляет богатую стандартную библиотеку, строгую типизацию и множество встроенных инструментов для повышения безопасности и удобства разработки.

Целью курсовой работы является разработка приложения на языке C#, представляющее собой task-менеджер в виде канбан-доски.

Разработка приложения проводится в команде из 2 человек:

* Маркелов Сергей Александрович (группа 1ПИб-02-3оп-22) – тимлид, Frontend-разработчик;
* Подтелков Владислав Владимирович (группа 1ПИб-02-3оп-22) – Backend-разработчик.

# Изучение и описание предметной области

Task-менеджер (или менеджер задач) – это программа, которая помогает планировать, отслеживать и управлять задачами. Такие инструменты могут быть очень простыми (список дел) или сложными (система управления проектами с дедлайнами, подзадачами и отслеживанием прогресса) [6].

Task-менеджеры полезны как в работе (организация проектов, дедлайны, командная работа), так и в личной жизни (покупки, планирование учёбы, целей, хобби и т. д.).

Основные функции task-менеджера:

* создание задач (например: «Сделать отчёт», «Позвонить клиенту»);
* установка дедлайнов (сроков выполнения);
* назначение приоритетов (важные/неважные, срочные/несрочные);
* группировка по проектам или категориям;
* добавление подзадач и чеклистов;
* назначение ответственных (если командная работа);
* напоминания и уведомления;
* отслеживание выполнения задач.

Существуют разные виды task-менеджеров:

* списковые – простой список задач с возможностью помечать их выполненными и удалять;
* канбан-доски – задачи представлены в виде карточек, которые перемещаются между разными колонками (например, «План», «В работе», «Готово»);
* табличные – задачи в виде таблицы с колонками (например, «Название», «Статус», «Дата», «Ответственный» и т. д.);
* ментальные карты – задачи отображаются как узлы на древовидной диаграмме;
* календарные – задачи и события располагаются в календаре (день, неделя, месяц).

Для реализации task-менеджера был выбран вид канбан-доски.

Канбан-доска – это визуальный способ управления задачами, в котором задачи представлены в виде карточек и перемещаются между колонками, отражающими стадии выполнения (рис. 1-2). Это один из самых популярных и интуитивных видов task-менеджеров, особенно в команде и при agile-разработке [5].



Рис. 1. Канбан-доска

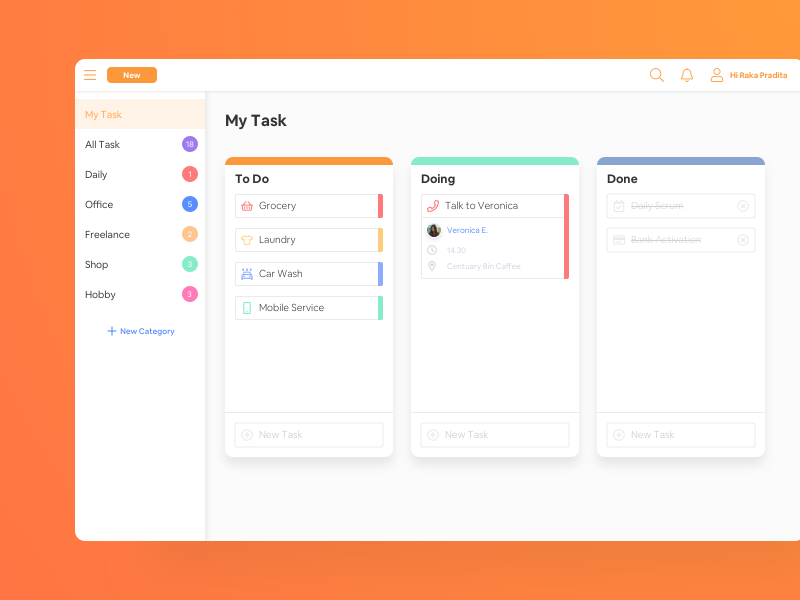


Рис. 2. Task-менеджер в виде канбан-доски

Каждая задача – это карточка, которая проходит через этапы работы, обычно организованные в колонки. Карточки перетаскиваются из одной колонки в другую по мере выполнения. Самый простой вариант канбан-доски состоит из 3 колонок:

* «To Do» («Сделать»);
* «In Progress» («В работе»);
* «Done» («Сделано»).

Карточка задачи может содержать следующие элементы:

* название задачи;
* описание;
* срок выполнения (дедлайн);
* назначенные участники;
* теги, метки;
* чеклист подзадач;
* комментарии.

Главные принципы Канбана:

* непрерывный поток – задачи постоянно двигаются по стадиям;
* визуализация работы – вся команда видит, что происходит;
* фокус на завершении, а не на запуске новых задач.

# Постановка задачи

Для реализации task-менеджера в виде канбан-доски был выбран следующий стек:

* технология WPF;
* паттерн MVVM;
* фреймворк Prism;
* система управления базами данных SQLite;
* система контроля версий Git.

Windows Presentation Foundation (WPF) – это технология от Microsoft для создания графических интерфейсов в Windows-приложениях. В отличие от старого Windows Forms, WPF использует DirectX для рендеринга, что позволяет строить современные, визуально привлекательные и производительные интерфейсы [7].

Одна из ключевых особенностей WPF – разделение дизайна и логики. Внешний вид интерфейса описывается в XAML (язык разметки на основе XML), а бизнес-логика пишется на C#. Это делает код чище и удобнее для работы в командах, где дизайнеры и разработчики могут работать параллельно.

WPF поддерживает сложные механизмы привязки данных (Data Binding), благодаря которым интерфейс автоматически обновляется при изменении данных. Также здесь есть продвинутые возможности стилизации и анимации, что позволяет создавать динамичные и кастомизированные интерфейсы.

MVVM (Model-View-ViewModel) – это архитектурный паттерн, который часто используется в WPF для разделения кода на логические компоненты, упрощения тестирования и поддержки модульности приложения [12].

Основные концепции MVVM:

* Model (Модель) – отвечает за данные и бизнес-логику; это могут быть классы, работающие с базой данных, API или вычислениями;
* View (Представление) – визуальная часть, которая отображает интерфейс и обрабатывает пользовательский ввод, но не содержит логики;
* ViewModel (Модель представления) – промежуточный слой между View и Model, преобразующий данные для отображения и обрабатывающий команды от пользователя.

Prism – это фреймворк, разработанный компанией Microsoft, предназначенный для создания масштабируемых, модульных и поддерживаемых настольных приложений на платформе WPF. Prism реализует набор архитектурных шаблонов, которые помогают разработчикам строить приложения с четко организованной структурой, разделением ответственности и возможностью расширения [11].

Основу фреймворка составляет система поддержки паттерна MVVM, включающая базовые реализации ключевых интерфейсов. Класс BindableBase автоматизирует механизм уведомлений об изменении свойств через INotifyPropertyChanged, а DelegateCommand предлагает стандартную реализацию интерфейса ICommand. Это позволяет разработчикам сосредоточиться на бизнес-логике, минимизируя написание шаблонного кода.

Важной особенностью Prism является его модульная система. Фреймворк предоставляет механизмы для динамической загрузки и управления независимыми функциональными модулями, что особенно ценно при разработке крупных приложений с распределённой командой разработчиков. Реализация модульности осуществляется через интерфейс IModule и специализированный каталог модулей.

Для организации навигации между представлениями Prism предлагает концепцию регионов (IRegionManager). Этот подход позволяет определять области содержимого в пользовательском интерфейсе и управлять ими независимо, что обеспечивает гибкость при построении сложных оконных интерфейсов.

Инфраструктура внедрения зависимостей, поддерживающая различные контейнеры (Unity, DryIoc), упрощает управление зависимостями между компонентами приложения. Дополнительно фреймворк включает механизм событий (IEventAggregator) для реализации слабосвязанного взаимодействия между различными частями системы.

SQLite – это компактная встраиваемая реляционная система управления базами данных (СУБД). В отличие от клиент-серверных СУБД (таких как MySQL или PostgreSQL), SQLite не требует отдельного серверного процесса и хранит всю базу данных в одном файле на диске, что делает её идеальным решением для локального хранения данных в приложениях [13].

Основной особенностью SQLite является её автономность – для работы не требуется настройка или администрирование. Библиотека обеспечивает полноценную поддержку SQL, включая сложные запросы, транзакции и триггеры, при этом оставаясь крайне легковесной. Все данные хранятся в одном кроссплатформенном файле, что упрощает перенос и резервное копирование.

Git – это распределённая система контроля версий. Она позволяет отслеживать изменения в файлах, координировать работу нескольких разработчиков над одним проектом и при необходимости возвращаться к предыдущим версиям кода [10].

Git работает по принципу распределённой архитектуры: каждый участник проекта хранит полную копию репозитория со всей историей изменений. Это отличает Git от централизованных систем и обеспечивает гибкость – разработчики могут работать автономно, а затем синхронизировать изменения через удалённые репозитории (например, на GitHub или GitLab).

# Логическое проектирование

Блок-схема – это графическое представление алгоритма, в котором отдельные шаги изображаются в виде геометрических фигур (блоков), соединённых стрелками, указывающими направление выполнения. С помощью блок-схем можно рассмотреть работу алгоритма всей программы или некоторых ее функций [3].

Основными процессами в программе являются:

* создание задач;
* изменение задач;
* удаление задач;
* перетаскивание задач.

Блок-схема процесса создания новой задачи представлена на рис. 3.



Рис. 3. Блок-схема процесса создания задачи

Блок-схема процессов изменения и удаления задачи представлена на рис. 4.



Рис. 4. Блок-схема процессов изменения и удаления задачи

Блок-схема процесса перетаскивания задачи представлена на рис. 5.



Рис. 5. Блок-схема процесса перетаскивания задачи

# Физическое проектирование

Для логической структуры программы, представленной выше, были разработаны следующие модули (табл. 1).

Таблица 1

Модули программы

|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение | Тип |
| MainWindow.xaml | Главное окно с канбан-доской | XAML |
| MainWindow.xaml.cs | Логика главного окна | Code-behind |
| MainWindowViewModel.cs | Управление задачами и их состоянием | ViewModel |
| NotificationView.xaml | Окно уведомлений | XAML |
| NotificationView.xaml.cs | Логика окна уведомлений | Code-behind |
| NotificationViewModel.cs | Управление состоянием уведомлений | ViewModel |
| KanbanTaskViewModel.cs | Управление отдельной задачей | ViewModel |
| GreaterThanConverter.cs | Конвертер для сравнения значений | Converter |
| NumberToVisibilityConverter.cs | Конвертер для управления видимостью | Converter |
| DragDropBehavior.cs | Поведение для drag-and-drop | Behavior |
| App.xaml | Глобальные ресурсы и стили | XAML |
| App.xaml.cs | Точка входа приложения | Code-behind |

# Проектирование интерфейса

При проектировании интерфейса программы важной частью является создание макета.

Макет интерфейса программы – это визуальная схема или прототип, который показывает расположение элементов управления (кнопок, меню, полей ввода и т. д.) и общий дизайн будущего программного обеспечения [2].

Макет интерфейса необходим для того, чтобы наглядно представить, как пользователь будет взаимодействовать с программой. Он помогает разобраться в логике работы интерфейса, определить последовательность действий и убедиться, что все элементы расположены удобно и логично.

С помощью макета можно заранее оценить удобство пользовательского опыта (UX), проверить, насколько интуитивно понятны навигация и функционал. Это позволяет выявить потенциальные проблемы до начала разработки и избежать дорогостоящих правок на поздних этапах.

Кроме того, макет служит важным инструментом согласования дизайна между дизайнерами, разработчиками и заказчиком. Он дает возможность наглядно продемонстрировать идею, внести корректировки и утвердить окончательный вариант до того, как программисты приступят к написанию кода. Таким образом, макет интерфейса экономит время, снижает риски недопонимания и помогает создать более продуманный и удобный продукт.

Макет интерфейса разрабатываемой программы представлен на рис. 6.

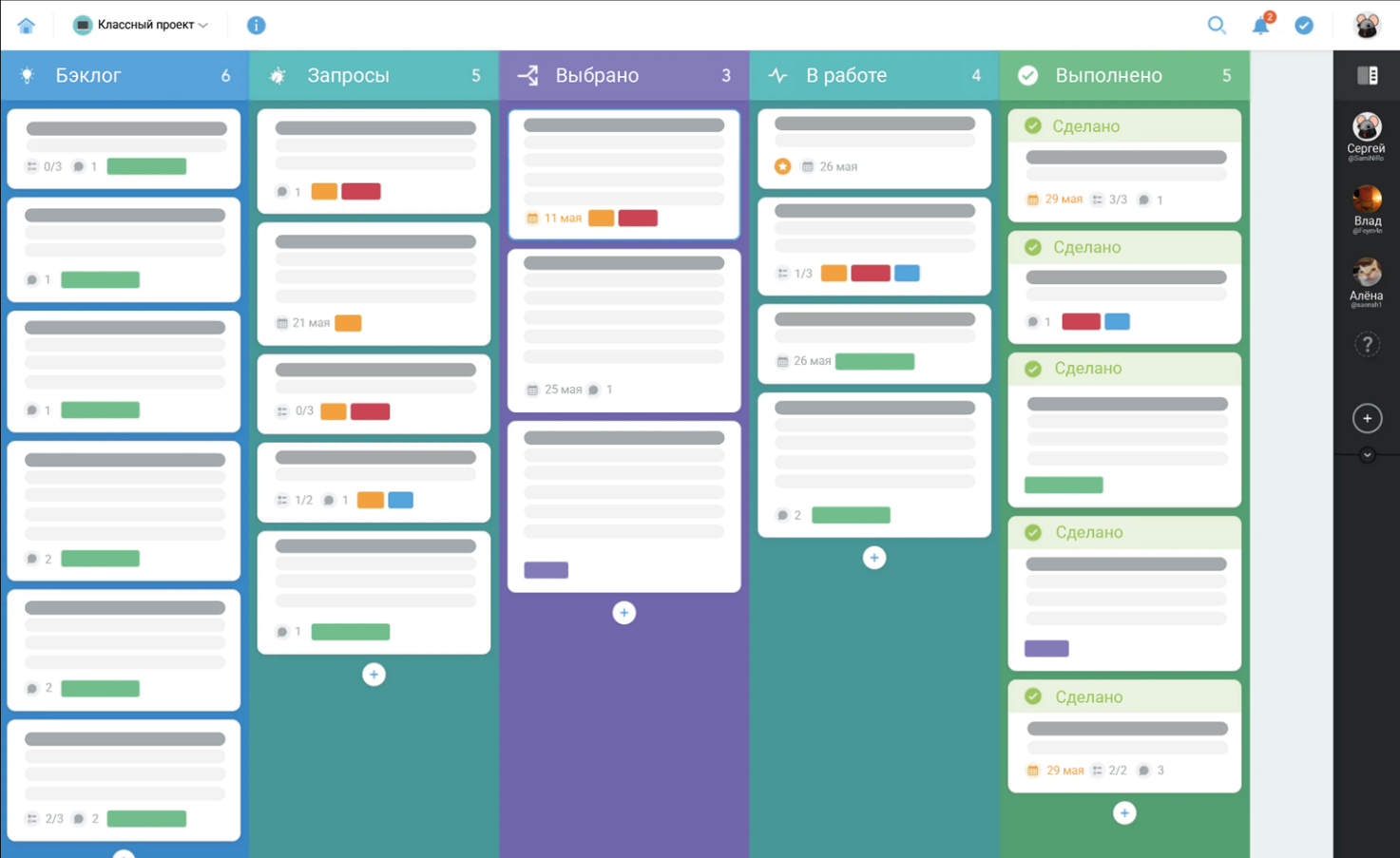


Рис. 6. Макет интерфейса программы

# Тестирование программы

Тестирование программного обеспечения – это процесс анализа, проверки и оценки программного продукта с целью выявления дефектов, проверки соответствия продукта установленным требованиям и обеспечения его качества. Оно охватывает действия по запуску программ в контролируемых условиях, наблюдению за их поведением, сравнению полученных результатов с ожидаемыми и фиксации всех несоответствий. Тестирование служит инструментом обеспечения надёжности и стабильности системы, а также помогает убедиться, что программное обеспечение соответствует функциональным и нефункциональным требованиям, удовлетворяет ожиданиям заказчика и готово к эксплуатации в реальных условиях [8].

Результаты тестирования модулей представлены в табл. 2.

Таблица 2

Тестирование модулей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата | Тестируемый модуль или подпрограмма | Кто проводил тестирование | Описание теста | Результат тестирования |
| 1 | 2 | 3 | 4 | 5 |
| 28.03.2025 | App.xaml.cs | Маркелов С. А. | Проверка инициализации окна | Успех |
| 01.04.2025 | MainWindow.xaml | Маркелов С. А. | Проверка корректности отображения всех колонок канбан-доски | Успех |
| 01.04.2025 | MainWindow.xaml | Маркелов С. А. | Проверка стилей колонок канбан-доски | Ошибка. Стили отображаются неверно |

Продолжение табл. 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 01.04.2025 | MainWindow.xaml | Маркелов С. А. | Проверка стилей колонок канбан-доски | Успех |
| 01.04.2025 | MainWindow.xaml | Маркелов С. А. | Проверка отображения кнопок добавления задач | Ошибка. Не отображается интерфейс кнопок |
| 01.04.2025 | MainWindow.xaml | Маркелов С. А. | Проверка отображения кнопок добавления задач | Успех |
| 01.04.2025 | MainWindow.xaml | Маркелов С. А. | Проверка отображения счетчиков задач | Успех |
| 02.04.2025 | MainWindowViewModel.cs | Подтелков В. В. | Проверка инициализации коллекции задач | Ошибка. Задачи пропадали при перезапуске приложения |
| 02.04.2025 | MainWindowViewModel.cs | Подтелков В. В. | Проверка инициализации коллекции задач | Успех |
| 02.04.2025 | MainWindowViewModel.cs | Маркелов С. А. | Проверка редактирования задачи | Успех |
| 02.04.2025 | MainWindowViewModel.cs | Маркелов С. А. | Проверка удаления задачи | Успех |
| 02.04.2025 | MainWindowViewModel.cs | Подтелков В. В. | Проверка перемещения задачи | Ошибка. Задача всегда перемещалась в 1 колонку |

Продолжение табл. 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 02.04.2025 | MainWindowViewModel.cs | Подтелков В. В. | Проверка перемещения задачи | Успех |
| 15.04.2025 | NotificationView.xaml | Маркелов С. А. | Проверка анимации появления уведомления об ошибке | Успех |
| 15.04.2025 | NotificationView.xaml | Маркелов С. А. | Проверка анимации исчезновения уведомления об ошибке | Успех |
| 15.04.2025 | NotificationView.xaml | Маркелов С. А. | Проверка стилей уведомления об ошибке | Успех |
| 15.04.2025 | NotificationView.xaml | Маркелов С. А. | Проверка отображения текста уведомления об ошибке | Успех |
| 16.04.2025 | NotificationViewModel.cs | Подтелков В. В. | Проверка создания уведомления об ошибке | Успех |
| 16.04.2025 | NotificationViewModel.cs | Подтелков В. В. | Проверка автоматического исчезновения уведомления об ошибке | Успех |
| 20.04.2025 | DragDropBehavior.cs | Подтелков В. В. | Проверка начала перетаскивания задачи | Успех |
| 20.04.2025 | DragDropBehavior.cs | Подтелков В. В. | Проверка процесса перетаскивания задачи | Успех |
| 20.04.2025 | DragDropBehavior.cs | Подтелков В. В. | Проверка завершения перетаскивания задачи | Успех |
| 25.04.2025 | MainWindow.xaml | Маркелов С. А. | Проверка отображения полей редактирования задачи | Успех |
| 26.04.2025 | MainWindowViewModel.cs | Маркелов С. А. | Проверка работы кнопок редактирования задачи | Успех |
| 27.04.2025 | KanbanTaskViewModel.cs | Маркелов С. А. | Проверка создания задачи | Успех |

Продолжение табл. 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 27.04.2025 | KanbanTaskViewModel.cs | Маркелов С. А. | Проверка обновления задачи | Успех |
| 27.04.2025 | KanbanTaskViewModel.cs | Маркелов С. А. | Проверка изменения статуса задачи | Успех |

# Заключение

В ходе выполнения курсовой работы было разработано WPF-приложение на языке C#. Программа представляет собой task-менеджер, выполненный в виде канбан-доски. Он позволяет пользователю создавать задачи, изменять и удалять их, а также перемещать между колонками канбан-доски.

В рамках работы в команде в роли Frontend-разработчика был реализован интерфейс приложения. Также мною в роли тимлида проводилась организация работы, распределение задач между участниками команды и ведение репозитория на GitHub [14].

# Список литературы

1. Методика и организация самостоятельной работы студентов: учебно-методическое пособие / Е.В. Ершов, Л.Н. Виноградова, В.В. Селивановских [и др.]. – Череповец: ФГБОУ ВПО ЧГУ, 2015. – 243 с.
2. 5 сервисов для прототипирования и дизайна — Яндекс.Образование [электр.ресурс] https://education.yandex.ru/journal/5-servisov-dlya-prototipirovaniya-i-dizajna. Дата обращения: 06.06.2025.
3. Блок-схема: что это такое и как использовать — Skyeng [электр.ресурс] https://skyeng.ru/magazine/wiki/it-industriya/chto-takoe-blok-skhema/. Дата обращения: 06.06.2025.
4. Гайд по языкам программирования – ч.1: типы и парадигмы — Рефни [электр.ресурс] https://refni.vercel.app/blog/gayd-po-yazykam-programmirovaniya-ch1-tipy-i-paradigmy. Дата обращения: 06.06.2025.
5. Канбан: система управления задачами — Фоксфорд [электр.ресурс] https://media.foxford.ru/articles/kanban-tool. Дата обращения: 06.06.2025.
6. Лучшие планировщики задач 2025: обзор программ для планирования — Комьюнити [электр.ресурс] https://timeweb.com/ru/community/articles/luchshie-planirovshchiki-zadach-2025-obzor-programm-dlya-planirovaniya. Дата обращения: 06.06.2025.
7. Обзор Windows Presentation Foundation — Microsoft Learn [электр.ресурс] https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-9.0&viewFallbackFrom=netdesktop-5.0. Дата обращения: 06.06.2025.
8. Основы тестирования ПО для начинающих — Skypro [электр.ресурс] https://sky.pro/wiki/profession/osnovy-testirovaniya-po-dlya-nachinayushih/. Дата обращения: 06.06.2025.
9. Язык С#: где используют, что пишут, как появился и чем хорош — Skillbox [электр.ресурс] https://skillbox.ru/media/code/yazyk-s-gde-ispolzuyut-chto-pishut-kak-poyavilsya-i-chem-khorosh/. Дата обращения: 06.06.2025.
10. Git Guide — GitHub [электр.ресурс] https://github.com/git-guides. Дата обращения: 06.06.2025.
11. Introduction to Prism — Prism Library [электр.ресурс] https://prismlibrary.github.io/docs/. Дата обращения: 06.06.2025.
12. Model-View-ViewModel (MVVM) — Microsoft Learn [электр.ресурс] https://learn.microsoft.com/ru-ru/dotnet/architecture/maui/mvvm. Дата обращения: 06.06.2025.
13. SQLite — SkillFactory [электр.ресурс] https://blog.skillfactory.ru/glossary/sqlite/. Дата обращения: 06.06.2025.
14. task-manager — GitHub [электр.ресурс] https://github.com/SamiNiRo/task-manager. Дата обращения: 06.06.2025.

Приложение 1

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

Образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

                                     Институт информационных технологий

*наименование института (факультета)*

                           Математическое и программное обеспечение ЭВМ

*наименование кафедры*

                               C#-программирование

*Наименование дисциплины в соответствии с учебным планом*

УТВЕРЖДАЮ

Зав. кафедрой                  ,

д. т. н., профессор                    Ершов Е. В.

«      »                       2025 г.

Разработка программного обеспечения на языке C#

Техническое задание на курсовую работу

Листов 8

|  |  |
| --- | --- |
| Руководитель | Шаханов Н. И. |
|  | Ф.И.О. преподавателя |
| Исполнитель |  |
| студент | 1ПИб-02-3оп-22 |
|  | группа |
|  | Маркелов С. А. |
|  | Фамилия, имя, отчество |

2025 год

Введение

Курсовая работа направлена на разработку приложения на языке C#, представляющего собой task-менеджер в виде канбан-доски. Приложение позволит пользователю создавать, изменять и удалять задачи, а также перетаскивать их между колонками канбан-доски.

1. Основания для разработки

Основанием для разработки является задание на курсовую работу по дисциплине «Объектно-ориентированное программирование», выданное на кафедре МПО ЭВМ.

Дата утверждения: 22 февраля 2025 года.

Наименование темы разработки: «Разработка программного обеспечения на языке C#».

2. Назначение разработки

Назначением разработки является создание программы, позволяющей пользователю планировать задачи, разделяя их по различным колонкам в зависимости от степени готовности.

3. Требования к программе

* 1. Требования к функциональным характеристикам

К разрабатываемой программе предъявляются следующие требования:

* программа должна реализовывать структуру канбан-доски;
* канбан-доска должна состоять из 5 колонок:
* «Бэклог»;
* «Запросы»;
* «Выбрано»;
* «В работе»;
* «Готово»;
* программа должна предоставлять пользователю возможность добавлять, редактировать и удалять задачи;
* программа должна предоставлять пользователю возможность перемещать задачи между колонками с помощью drag-and-drop;
* задачи должны храниться в базе данных;
* в программе должны быть реализованы всплывающие уведомления об ошибках.
  1. Требования к надежности

К разрабатываемой программе предъявляются следующие требования:

* работа программы быть проверена и результаты проверки оформлены протоколом тестирования;
* должна быть реализована корректная запись задач в базу данных;
* должен быть реализован корректный вывод информации о задачах на экран.
  1. Условия эксплуатации

Для корректной работы программы необходимо:

* наличие любого устройства, на котором возможно запускать файлы формата .exe (компьютер, ноутбук);
* наличие на устройстве совместимой операционной системы.
  1. Требования к составу и параметрам технических средств

Для корректной работы программы необходимо:

* процессор не менее чем с 2 ядрами и тактовой частотой не ниже 1,6 ГГц;
* оперативная память: не менее 1 Гб для 32-разрядной ОС, 2 Гб для 64-разрядной ОС;
* 8 Гб свободного места на диске;
* разрешение экрана не менее 800 х 600;
* видеокарта с видеопамятью не менее 1 Гб;
* наличие мыши и клавиатуры.
  1. Требования к информационной и программной совместимости

Для корректной работы программы необходимо:

* установленная на устройстве операционная система Windows 7, 8, 8.1, 10 или 11;
* установленный на устройстве пакет библиотек «.NET Framework» версии 7.0 или выше.
  1. Требования к маркировке и упаковке

Требования не предъявляются.

* 1. Требования к транспортированию и хранению

Программа должна храниться на носителе информации объемом не менее 8 Гб (USB-флешка, CD-диск, HDD или SSD).

* 1. Специальные требования

Программа должна быть удобна для использования, в том числе для людей, плохо умеющих пользоваться компьютером.

4. Требования к программной документации

4.1. Содержание расчетно-пояснительной записки

Программная документация должна содержать расчётно-пояснительную записку с содержанием:

Титульный лист;

Оглавление;

Введение;

1. Изучение и описание предметной области;
2. Постановка задачи;
3. Выбор структур данных;
4. Логическое проектирование;
5. Физическое проектирование;
6. Проектирование интерфейса;
7. Тестирование;

Заключение;

Список литературы;

Приложения.

4.2. Требования к оформлению

Текстовые документы оформляются на белых листах формата А4, графический материал допускается представлять на листах формата A3. В соответствии с общими требованиями поля листа определяются следующим образом: левое – 30 мм, правое – 10 мм, верхнее – 20 мм, нижнее – 20 мм. Формат текста: Word for Windows, через полтора интервала (около тридцати строк на листе), шрифт – Times New Roman Cyr, размер шрифта – 14, отступ первой строки абзаца – 0,75 см. Количество знаков в строке, считая пробелы, – 60. Текст программы может быть расположен в две колонки, шрифт – Times New Roman Cyr, размер шрифта – 8. Нумерация всех страниц (в том числе и приложений) сквозная. Номер проставляется в середине верхнего поля страницы арабской цифрой. Номера страниц на титульном листе, аннотации и оглавлении не проставляются.

Наименование разделов, подразделов, пунктов должно быть кратким и соответствовать содержанию. Каждая новая глава печатается с новой страницы. Это же правило относится и к другим основным структурным частям работы: аннотации, оглавлению, введению, заключению, списку литературы, приложениям. Наименование разделов (основных частей) пишется прописными буквами по центру строки. Расстояние между заголовками и текстом, а также между заголовками разделов и подразделов должно быть равно двум интервалам. Наименования подразделов и пунктов размещаются с абзацного отступа (0,75 см) и печатаются с прописной буквы, без подчеркивания и без точки в конце. Расстояние между последней строкой текста предыдущего раздела и последующим заголовком при расположении их на одной странице должно быть равно трем интервалам. Разделы и подразделы нумеруются арабскими цифрами с точкой. Разделы имеют порядковые номера 1, 2 и т.д. Номер подраздела состоит из номера раздела и порядкового номера подраздела, входящего в данный раздел, разделенных точкой (например: 1.1, 2.5). При использовании ссылок на пункты, разделы и подразделы указывается порядковый номер раздела или пункта (например: «в разд. 2», «в п. 2.3.1»). Перечисления нумеруются арабскими цифрами со скобкой (например: 2), 3) и т. д.) с абзацного отступа. Для этого используются нумерованные списки. Допускается также применение маркированных списков одного вида. Таблицы в основном применяются для оформления цифрового материала. Шрифт – Times New Roman Cyr, размер шрифта – 14. Иногда возможен 10-й размер шрифта. Обозначения единиц физических величин необходимо применять в системе СИ. Номер таблицы размещается в правом верхнем углу над заголовком.

5. Стадии и этапы разработки

В данном пункте описаны стадии и этапы разработки программы (табл. П1.1).

Таблица П1.1

Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапа разработки | Сроки разработки | Результат выполнения | Отметка о выполнении |
| Оформление технического задания | 15.03.2025 | Оформленное техническое задание | Выполнено |
| Анализ предметной области | 29.03.2025 | Проведен анализ предметной области | Выполнено |
| Создание макета интерфейса | 12.04.2025 | Макет интерфейса | Выполнено |
| Проектирование приложения | 26.04.2025 | Выбраны структуры данных, построены блок-схемы | Выполнено |
| Создание приложения | 17.04.2025 | Готовая программа | Выполнено |
| Тестирование и отладка | 31.05.2025 | Конечный вариант программы | Выполнено |
| Оформление сопроводительной документации | 05.06.2025 | Оформленная сопроводительная документация | Выполнено |

6. Порядок контроля и приемки

В данном пункте описан порядок контроля и приемки курсовой работы (табл. П1.2).

Таблица П1.2

Порядок контроля и приемки

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование контрольного этапа выполнения курсовой работы | Сроки контроля | Результат выполнения | Отметка о приемке результата контрольного этапа |
| Сдача технического задания | 15.03.2025 | Согласованное техническое задания | Выполнено |
| Сдача расчетно-пояснительной записки | 05.06.2025 | Согласованная расчетно-пояснительная записка | Выполнено |
| Сдача курсовой работы | 11.06.2025 | Получение оценки за выполненную работу |  |

# Приложение 2

Текст программы

Текст модуля MainWindow.xaml представлен на рис. П2.1.

<?xml version="1.0" encoding="utf-8"?>

<Window x:Class="TaskManager.Views.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:prism="http://prismlibrary.com/"

xmlns:models="clr-namespace:TaskManager.Models"

xmlns:behaviors="clr-namespace:TaskManager.Behaviors"

xmlns:converters="clr-namespace:TaskManager.Converters"

xmlns:local="clr-namespace:TaskManager.Views"

Title="Канбан доска" Height="800" Width="1200"

prism:ViewModelLocator.AutoWireViewModel="True">

<Window.Resources>

<converters:GreaterThanConverter x:Key="GreaterThanConverter"/>

<!-- Цвета колонок -->

<SolidColorBrush x:Key="BacklogColumnBrush" Color="#3498db"/>

<SolidColorBrush x:Key="BacklogHeaderBrush" Color="#2980b9"/>

<SolidColorBrush x:Key="RequestColumnBrush" Color="#16a085"/>

<SolidColorBrush x:Key="RequestHeaderBrush" Color="#147a6c"/>

<SolidColorBrush x:Key="SelectedColumnBrush" Color="#9b59b6"/>

<SolidColorBrush x:Key="SelectedHeaderBrush" Color="#8e44ad"/>

<SolidColorBrush x:Key="InProgressColumnBrush" Color="#e67e22"/>

<SolidColorBrush x:Key="InProgressHeaderBrush" Color="#d35400"/>

<SolidColorBrush x:Key="CompletedColumnBrush" Color="#2ecc71"/>

<SolidColorBrush x:Key="CompletedHeaderBrush" Color="#1e8449"/>

<SolidColorBrush x:Key="DeleteButtonBrush" Color="#ff4444"/>

<SolidColorBrush x:Key="HoverBackgroundBrush" Color="#F8F8F8"/>

<Style x:Key="TaskBorderStyle" TargetType="Border">

<Setter Property="Background" Value="White"/>

<Setter Property="CornerRadius" Value="5"/>

<Setter Property="Margin" Value="2"/>

<Setter Property="Padding" Value="5"/>

<Style.Triggers>

<DataTrigger Binding="{Binding RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=ListViewItem}, Path=IsMouseOver}" Value="True">

<Setter Property="Background" Value="#F8F8F8"/>

</DataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="EditButtonStyle" TargetType="Button">

<Setter Property="Width" Value="70"/>

<Setter Property="Margin" Value="0,0,5,0"/>

<Setter Property="Height" Value="25"/>

</Style>

<Style x:Key="DeleteButtonStyle" TargetType="Button" BasedOn="{StaticResource EditButtonStyle}">

<Setter Property="Background" Value="#ff4444"/>

<Setter Property="Foreground" Value="White"/>

</Style>

<DataTemplate x:Key="TaskTemplate">

Рис П2.1. Текст модуля MainWindow.xaml

Рис. П2.1. Продолжение

<Border x:Name="TaskBorder" Background="White" CornerRadius="5" Margin="2" Padding="5">

<Grid>

<Grid x:Name="ViewMode">

<StackPanel>

<TextBlock Text="{Binding Title}"

TextWrapping="Wrap"

FontWeight="SemiBold"/>

<TextBlock Text="{Binding Description}"

TextWrapping="Wrap"

Margin="0,5,0,5"

Opacity="0.7"/>

</StackPanel>

</Grid>

<Grid x:Name="EditMode" Visibility="Collapsed">

<StackPanel>

<TextBox Text="{Binding Title, UpdateSourceTrigger=PropertyChanged}"

Margin="0,0,0,5"/>

<TextBox Text="{Binding Description, UpdateSourceTrigger=PropertyChanged}"

TextWrapping="Wrap"

AcceptsReturn="True"

MinHeight="60"

Margin="0,0,0,5"/>

<StackPanel Orientation="Horizontal"

HorizontalAlignment="Right"

Margin="0,0,0,5">

<Button x:Name="DeleteButton"

Content="Удалить"

Command="{Binding DataContext.DeleteTaskCommand, RelativeSource={RelativeSource AncestorType=Window}}"

CommandParameter="{Binding}"

Style="{StaticResource DeleteButtonStyle}"/>

<Button Content="Отмена"

Command="{Binding DataContext.CancelEditCommand, RelativeSource={RelativeSource AncestorType=Window}}"

CommandParameter="{Binding}"

Style="{StaticResource EditButtonStyle}"/>

<Button Content="Сохранить"

Command="{Binding DataContext.SaveTaskCommand, RelativeSource={RelativeSource AncestorType=Window}}"

CommandParameter="{Binding}"

Style="{StaticResource EditButtonStyle}"/>

</StackPanel>

</StackPanel>

</Grid>

</Grid>

</Border>

<DataTemplate.Triggers>

<DataTrigger Binding="{Binding IsEditing}" Value="True">

<Setter TargetName="ViewMode" Property="Visibility" Value="Collapsed"/>

<Setter TargetName="EditMode" Property="Visibility" Value="Visible"/>

<Setter TargetName="TaskBorder" Property="Background" Value="#F8F8F8"/>

</DataTrigger>

<MultiDataTrigger>

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsEditing}" Value="True"/>

<Condition Binding="{Binding Id}" Value="0"/>

</MultiDataTrigger.Conditions>

<Setter TargetName="DeleteButton" Property="Visibility" Value="Collapsed"/>

</MultiDataTrigger>

<MultiDataTrigger>

Рис. П2.1. Продолжение

<MultiDataTrigger.Conditions>

<Condition Binding="{Binding IsEditing}" Value="True"/>

<Condition Binding="{Binding Id, ConverterParameter=0, Converter={StaticResource GreaterThanConverter}}" Value="True"/>

</MultiDataTrigger.Conditions>

<Setter TargetName="DeleteButton" Property="Visibility" Value="Visible"/>

</MultiDataTrigger>

<DataTrigger Binding="{Binding RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=ListViewItem}, Path=IsMouseOver}" Value="True">

<Setter TargetName="TaskBorder" Property="Background" Value="#F8F8F8"/>

</DataTrigger>

</DataTemplate.Triggers>

</DataTemplate>

<Style TargetType="ListView">

<Setter Property="ScrollViewer.HorizontalScrollBarVisibility" Value="Disabled"/>

<Setter Property="ScrollViewer.VerticalScrollBarVisibility" Value="Auto"/>

<Setter Property="Background" Value="Transparent"/>

<Setter Property="BorderThickness" Value="0"/>

</Style>

<Style TargetType="ListViewItem">

<Setter Property="HorizontalContentAlignment" Value="Stretch"/>

<Setter Property="Margin" Value="0,2"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="ListViewItem">

<Border x:Name="Border"

Background="Transparent">

<VisualStateManager.VisualStateGroups>

<VisualStateGroup x:Name="CommonStates">

<VisualState x:Name="Normal"/>

<VisualState x:Name="MouseOver">

<Storyboard>

<DoubleAnimation Storyboard.TargetName="Border"

Storyboard.TargetProperty="(UIElement.RenderTransform).(ScaleTransform.ScaleX)"

To="1.02" Duration="0:0:0.1"/>

<DoubleAnimation Storyboard.TargetName="Border"

Storyboard.TargetProperty="(UIElement.RenderTransform).(ScaleTransform.ScaleY)"

To="1.02" Duration="0:0:0.1"/>

</Storyboard>

</VisualState>

</VisualStateGroup>

</VisualStateManager.VisualStateGroups>

<Border.RenderTransform>

<ScaleTransform ScaleX="1" ScaleY="1"/>

</Border.RenderTransform>

<ContentPresenter/>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="Border" Property="Effect">

<Setter.Value>

<DropShadowEffect BlurRadius="10" ShadowDepth="2" Opacity="0.3"/>

</Setter.Value>

</Setter>

</Trigger>

</ControlTemplate.Triggers>

Рис. П2.1. Продолжение

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="AddButtonStyle" TargetType="Button">

<Setter Property="Background" Value="Transparent"/>

<Setter Property="BorderThickness" Value="0"/>

<Setter Property="Width" Value="30"/>

<Setter Property="Height" Value="30"/>

<Setter Property="Margin" Value="0,0,5,0"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Grid>

<Ellipse Fill="White" Opacity="0.2"/>

<Path Fill="White"

Data="M19,13H13V19H11V13H5V11H11V5H13V11H19V13Z"

Width="16"

Height="16"

Stretch="Uniform"/>

</Grid>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="ColumnHeaderStyle" TargetType="TextBlock">

<Setter Property="Foreground" Value="White"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="FontWeight" Value="Bold"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style x:Key="ColumnIconStyle" TargetType="Path">

<Setter Property="Fill" Value="White"/>

<Setter Property="Width" Value="27"/>

<Setter Property="Height" Value="27"/>

<Setter Property="Margin" Value="0,0,5,0"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style x:Key="KanbanColumnStyle" TargetType="Border">

<Setter Property="Margin" Value="5"/>

<Setter Property="CornerRadius" Value="5"/>

</Style>

<Style x:Key="KanbanColumnHeaderStyle" TargetType="Border">

<Setter Property="DockPanel.Dock" Value="Top"/>

<Setter Property="CornerRadius" Value="5,5,0,0"/>

<Setter Property="Padding" Value="10,5"/>

</Style>

<Style x:Key="KanbanListViewStyle" TargetType="ListView" BasedOn="{StaticResource {x:Type ListView}}">

<Setter Property="AllowDrop" Value="True"/>

<Setter Property="Margin" Value="5"/>

<Setter Property="ItemTemplate" Value="{StaticResource TaskTemplate}"/>

<Setter Property="behaviors:DragDropBehavior.IsEnabled" Value="True"/>

Рис. П2.1. Продолжение

</Style>

<!-- Данные иконок для заголовков колонок (Material Design Icons) -->

<Geometry x:Key="BacklogIconData">M13,14H11V10H13V14M13,18H11V16H13V18M19,3H5C3.89,3 3,3.89 3,5V19C3,20.11 3.89,21 5,21H19C20.11,21 21,20.11 21,19V5C21,3.89 20.11,3 19,3M19,19H5V5H19V19Z</Geometry>

<Geometry x:Key="RequestIconData">M17,10H14V8H17V10M17,14H14V12H17V14M17,18H14V16H17V18M19,3H5C3.89,3 3,3.89 3,5V19C3,20.11 3.89,21 5,21H19C20.11,21 21,20.11 21,19V5C21,3.89 20.11,3 19,3M19,19H5V5H19V19M11,7H7V11H11V7M11,11.5H7V15.5H11V11.5M11,16H7V20H11V16Z</Geometry>

<Geometry x:Key="SelectedIconData">M12,15.39L16.24,18.55L14.71,13.4L18.81,10.23L13.66,10.1L12,5.12L10.34,10.1L5.19,10.23L9.29,13.4L7.76,18.55L12,15.39M12,2A10,10 0 0,0 2,12A10,10 0 0,0 12,22A10,10 0 0,0 22,12A10,10 0 0,0 12,2Z</Geometry>

<Geometry x:Key="InProgressIconData">M12,20A8,8 0 0,0 20,12A8,8 0 0,0 12,4A8,8 0 0,0 4,12A8,8 0 0,0 12,20M12,2A10,10 0 0,1 22,12A10,10 0 0,1 12,22A10,10 0 0,1 2,12A10,10 0 0,1 12,2M12,7V13L17.4,16.2L16.6,17.6L10.4,14V7H12Z</Geometry>

<Geometry x:Key="CompletedIconData">M12,2A10,10 0 0,0 2,12A10,10 0 0,0 12,22A10,10 0 0,0 22,12A10,10 0 0,0 12,2M12,4A8,8 0 0,1 20,12A8,8 0 0,1 12,20A8,8 0 0,1 4,12A8,8 0 0,1 12,4M10,17L5,12L6.41,10.59L10,14.17L17.59,6.58L19,8L10,17Z</Geometry>

</Window.Resources>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<!-- Бэклог -->

<Border Grid.Column="0" Background="#3498db" Margin="5">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!-- Заголовок колонки Бэклог -->

<Border Grid.Row="0" Background="{StaticResource BacklogHeaderBrush}" Style="{StaticResource KanbanColumnHeaderStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

<Path Grid.Column="0" Style="{StaticResource ColumnIconStyle}" Data="{StaticResource BacklogIconData}"/>

<StackPanel Grid.Column="1" Orientation="Horizontal" VerticalAlignment="Center">

<TextBlock Text="Бэклог" Style="{StaticResource ColumnHeaderStyle}"/>

<TextBlock Text="{Binding BacklogTaskCount}" Style="{StaticResource ColumnHeaderStyle}" Margin="5,0,0,0"/>

</StackPanel>

<Button Grid.Column="2" Style="{StaticResource AddButtonStyle}" Command="{Binding AddBacklogTaskCommand}"/>

</Grid>

</Border>

<ListView Grid.Row="1"

Рис. П2.1. Продолжение

ItemsSource="{Binding BacklogTasks}"

Tag="{x:Static models:KanbanTaskStatus.Backlog}"

Style="{StaticResource KanbanListViewStyle}"/>

</Grid>

</Border>

<!-- Запросы -->

<Border Grid.Column="1" Background="#16a085" Margin="5">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!-- Заголовок колонки Запросы -->

<Border Grid.Row="0" Background="{StaticResource RequestHeaderBrush}" Style="{StaticResource KanbanColumnHeaderStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

<Path Grid.Column="0" Style="{StaticResource ColumnIconStyle}" Data="{StaticResource RequestIconData}"/>

<StackPanel Grid.Column="1" Orientation="Horizontal" VerticalAlignment="Center">

<TextBlock Text="Запросы" Style="{StaticResource ColumnHeaderStyle}"/>

<TextBlock Text="{Binding RequestTaskCount}" Style="{StaticResource ColumnHeaderStyle}" Margin="5,0,0,0"/>

</StackPanel>

<Button Grid.Column="2" Style="{StaticResource AddButtonStyle}" Command="{Binding AddRequestTaskCommand}"/>

</Grid>

</Border>

<ListView Grid.Row="1"

ItemsSource="{Binding RequestTasks}"

Tag="{x:Static models:KanbanTaskStatus.Request}"

Style="{StaticResource KanbanListViewStyle}"/>

</Grid>

</Border>

<!-- Выбрано -->

<Border Grid.Column="2" Background="#9b59b6" Margin="5">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!-- Заголовок колонки Выбрано -->

<Border Grid.Row="0" Background="{StaticResource SelectedHeaderBrush}" Style="{StaticResource KanbanColumnHeaderStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

Рис. П2.1. Продолжение

<Path Grid.Column="0" Style="{StaticResource ColumnIconStyle}" Data="{StaticResource SelectedIconData}"/>

<StackPanel Grid.Column="1" Orientation="Horizontal" VerticalAlignment="Center">

<TextBlock Text="Выбрано" Style="{StaticResource ColumnHeaderStyle}"/>

<TextBlock Text="{Binding SelectedTaskCount}" Style="{StaticResource ColumnHeaderStyle}" Margin="5,0,0,0"/>

</StackPanel>

<Button Grid.Column="2" Style="{StaticResource AddButtonStyle}" Command="{Binding AddSelectedTaskCommand}"/>

</Grid>

</Border>

<ListView Grid.Row="1"

ItemsSource="{Binding SelectedTasks}"

Tag="{x:Static models:KanbanTaskStatus.Selected}"

Style="{StaticResource KanbanListViewStyle}"/>

</Grid>

</Border>

<!-- В работе -->

<Border Grid.Column="3" Background="#e67e22" Margin="5">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!-- Заголовок колонки В работе -->

<Border Grid.Row="0" Background="{StaticResource InProgressHeaderBrush}" Style="{StaticResource KanbanColumnHeaderStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

<Path Grid.Column="0" Style="{StaticResource ColumnIconStyle}" Data="{StaticResource InProgressIconData}"/>

<StackPanel Grid.Column="1" Orientation="Horizontal" VerticalAlignment="Center">

<TextBlock Text="В работе" Style="{StaticResource ColumnHeaderStyle}"/>

<TextBlock Text="{Binding InProgressTaskCount}" Style="{StaticResource ColumnHeaderStyle}" Margin="5,0,0,0"/>

</StackPanel>

<Button Grid.Column="2" Style="{StaticResource AddButtonStyle}" Command="{Binding AddInProgressTaskCommand}"/>

</Grid>

</Border>

<ListView Grid.Row="1"

ItemsSource="{Binding InProgressTasks}"

Tag="{x:Static models:KanbanTaskStatus.InProgress}"

Style="{StaticResource KanbanListViewStyle}"/>

</Grid>

</Border>

<!-- Готово -->

<Border Grid.Column="4" Background="#27ae60" Margin="5">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

Рис. П2.1. Продолжение

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!-- Заголовок колонки Готово -->

<Border Grid.Row="0" Background="{StaticResource CompletedHeaderBrush}" Style="{StaticResource KanbanColumnHeaderStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

<Path Grid.Column="0" Style="{StaticResource ColumnIconStyle}" Data="{StaticResource CompletedIconData}"/>

<StackPanel Grid.Column="1" Orientation="Horizontal" VerticalAlignment="Center">

<TextBlock Text="Готово" Style="{StaticResource ColumnHeaderStyle}"/>

<TextBlock Text="{Binding CompletedTaskCount}" Style="{StaticResource ColumnHeaderStyle}" Margin="5,0,0,0"/>

</StackPanel>

<Button Grid.Column="2" Style="{StaticResource AddButtonStyle}" Command="{Binding AddCompletedTaskCommand}"/>

</Grid>

</Border>

<ListView Grid.Row="1"

ItemsSource="{Binding CompletedTasks}"

Tag="{x:Static models:KanbanTaskStatus.Completed}"

Style="{StaticResource KanbanListViewStyle}"/>

</Grid>

</Border>

<!-- Контейнер для уведомлений в нижнем правом углу -->

<ItemsControl ItemsSource="{Binding Notifications}" Grid.ColumnSpan="5" IsHitTestVisible="False" HorizontalAlignment="Right" VerticalAlignment="Bottom" Margin="10">

<ItemsControl.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel Orientation="Vertical"/>

</ItemsPanelTemplate>

</ItemsControl.ItemsPanel>

<ItemsControl.ItemTemplate>

<DataTemplate>

<local:NotificationView Margin="0,5,0,0"/>

</DataTemplate>

</ItemsControl.ItemTemplate>

</ItemsControl>

</Grid>

</Window>

Текст модуля MainWindow.xaml.cs представлен на рис. П2.1.

using System.Windows;

using TaskManager.ViewModels;

namespace TaskManager.Views

{

// Главное окно приложения

Рис П2.2. Текст модуля MainWindow.xaml.cs

Рис. П2.2. Продолжение

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

Loaded += MainWindow\_Loaded;

}

private async void MainWindow\_Loaded(object sender, RoutedEventArgs e)

{

if (DataContext is MainWindowViewModel viewModel)

{

await viewModel.InitializeAsync();

}

}

}

}

Текст модуля MainWindowViewModel.cs представлен на рис. П2.3.

using Prism.Commands;

using Prism.Mvvm;

using System;

using System.Collections.ObjectModel;

using System.Linq;

using System.Threading.Tasks;

using TaskManager.Models;

using TaskManager.Services;

using System.Windows.Input;

using TaskManager.Behaviors;

namespace TaskManager.ViewModels

{

// Основной класс для управления задачами в главном окне

public class MainWindowViewModel : BindableBase, IDragDropHandler

{

// Сервис для работы с задачами

private readonly ITaskService \_taskService;

private readonly IDialogService \_dialogService;

private readonly INotificationService \_notificationService;

// Список задач в бэклоге

private ObservableCollection<KanbanTaskViewModel> \_backlogTasks;

public ObservableCollection<KanbanTaskViewModel> BacklogTasks

{

get => \_backlogTasks;

set => SetProperty(ref \_backlogTasks, value);

}

// Список запрошенных задач

private ObservableCollection<KanbanTaskViewModel> \_requestTasks;

public ObservableCollection<KanbanTaskViewModel> RequestTasks

{

get => \_requestTasks;

set => SetProperty(ref \_requestTasks, value);

Рис П2.3. Текст модуля MainWindowViewModel.cs

Рис. П2.3. Продолжение

}

// Список выбранных задач

private ObservableCollection<KanbanTaskViewModel> \_selectedTasks;

public ObservableCollection<KanbanTaskViewModel> SelectedTasks

{

get => \_selectedTasks;

set => SetProperty(ref \_selectedTasks, value);

}

// Список задач в работе

private ObservableCollection<KanbanTaskViewModel> \_inProgressTasks;

public ObservableCollection<KanbanTaskViewModel> InProgressTasks

{

get => \_inProgressTasks;

set => SetProperty(ref \_inProgressTasks, value);

}

// Список выполненных задач

private ObservableCollection<KanbanTaskViewModel> \_completedTasks;

public ObservableCollection<KanbanTaskViewModel> CompletedTasks

{

get => \_completedTasks;

set => SetProperty(ref \_completedTasks, value);

}

private KanbanTaskViewModel \_editingTask;

public KanbanTaskViewModel EditingTask

{

get => \_editingTask;

set => SetProperty(ref \_editingTask, value);

}

private bool \_isEditing;

public bool IsEditing

{

get => \_isEditing;

set => SetProperty(ref \_isEditing, value);

}

// Команды для добавления задач в разные колонки

public ICommand AddBacklogTaskCommand { get; private set; }

public ICommand AddRequestTaskCommand { get; private set; }

public ICommand AddSelectedTaskCommand { get; private set; }

public ICommand AddInProgressTaskCommand { get; private set; }

public ICommand AddCompletedTaskCommand { get; private set; }

public ICommand EditTaskCommand { get; private set; }

public ICommand SaveTaskCommand { get; private set; }

public ICommand CancelEditCommand { get; private set; }

public ICommand DeleteTaskCommand { get; private set; }

// Конструктор с инициализацией команд и загрузкой задач

public MainWindowViewModel(ITaskService taskService, IDialogService dialogService, INotificationService notificationService)

{

\_taskService = taskService ?? throw new ArgumentNullException(nameof(taskService));

\_dialogService = dialogService ?? throw new ArgumentNullException(nameof(dialogService));

\_notificationService = notificationService ?? throw new

ArgumentNullException(nameof(notificationService));

Рис. П2.3. Продолжение

InitializeCollections();

InitializeCommands();

}

private void InitializeCollections()

{

BacklogTasks = new ObservableCollection<KanbanTaskViewModel>();

RequestTasks = new ObservableCollection<KanbanTaskViewModel>();

SelectedTasks = new ObservableCollection<KanbanTaskViewModel>();

InProgressTasks = new ObservableCollection<KanbanTaskViewModel>();

CompletedTasks = new ObservableCollection<KanbanTaskViewModel>();

// Подписываемся на изменения коллекций для обновления счетчиков

BacklogTasks.CollectionChanged += (s, e) => RaisePropertyChanged(nameof(BacklogTaskCount));

RequestTasks.CollectionChanged += (s, e) => RaisePropertyChanged(nameof(RequestTaskCount));

SelectedTasks.CollectionChanged += (s, e) => RaisePropertyChanged(nameof(SelectedTaskCount));

InProgressTasks.CollectionChanged += (s, e) => RaisePropertyChanged(nameof(InProgressTaskCount));

CompletedTasks.CollectionChanged += (s, e) => RaisePropertyChanged(nameof(CompletedTaskCount));

}

private void InitializeCommands()

{

AddBacklogTaskCommand = new DelegateCommand(() => CreateNewTask(KanbanTaskStatus.Backlog));

AddRequestTaskCommand = new DelegateCommand(() => CreateNewTask(KanbanTaskStatus.Request));

AddSelectedTaskCommand = new DelegateCommand(() => CreateNewTask(KanbanTaskStatus.Selected));

AddInProgressTaskCommand = new DelegateCommand(() => CreateNewTask(KanbanTaskStatus.InProgress));

AddCompletedTaskCommand = new DelegateCommand(() => CreateNewTask(KanbanTaskStatus.Completed));

EditTaskCommand = new DelegateCommand<KanbanTaskViewModel>(StartEditingTask);

SaveTaskCommand = new DelegateCommand<KanbanTaskViewModel>(async (task) => await SaveTaskAsync(task));

CancelEditCommand = new DelegateCommand<KanbanTaskViewModel>(CancelEditing);

DeleteTaskCommand = new DelegateCommand<KanbanTaskViewModel>(async (task) => await DeleteTaskAsync(task));

}

public async Task InitializeAsync()

{

await LoadTasksAsync();

}

private void CreateNewTask(KanbanTaskStatus status)

{

var newTask = new KanbanTask

{

Status = status,

Title = "",

Description = "",

CreatedAt = DateTime.Now

};

var taskViewModel = new KanbanTaskViewModel(newTask) { IsEditing = true };

GetCollectionForStatus(status).Insert(0, taskViewModel);

}

private void StartEditingTask(KanbanTaskViewModel task)

{

Рис. П2.3. Продолжение

if (task == null) return;

task.IsEditing = true;

}

private async Task SaveTaskAsync(KanbanTaskViewModel task)

{

try

{

if (task == null) return;

if (string.IsNullOrWhiteSpace(task.Title))

{

\_notificationService.ShowNotification("Заголовок задачи не может быть пустым!");

return;

}

if (task.Id == 0)

{

// Новая задача

var addedTask = await \_taskService.AddTaskAsync(task.Model);

var collection = GetCollectionForStatus(task.Status);

var index = collection.IndexOf(task);

var newTaskViewModel = new KanbanTaskViewModel(addedTask);

collection[index] = newTaskViewModel;

}

else

{

// Обновление существующей задачи

await \_taskService.UpdateTaskAsync(task.Model);

}

task.IsEditing = false;

}

catch (Exception ex)

{

\_dialogService.ShowError($"Ошибка при сохранении задачи: {ex.Message}");

}

}

private void CancelEditing(KanbanTaskViewModel task)

{

if (task == null) return;

if (task.Id == 0) // Новая задача

{

var collection = GetCollectionForStatus(task.Status);

collection.Remove(task);

}

else

{

task.IsEditing = false;

}

}

private async Task DeleteTaskAsync(KanbanTaskViewModel task)

{

try

{

if (task == null || task.Id == 0) return;

Рис. П2.3. Продолжение

await \_taskService.DeleteTaskAsync(task.Id);

var collection = GetCollectionForStatus(task.Status);

collection.Remove(task);

}

catch (Exception ex)

{

\_dialogService.ShowError($"Ошибка при удалении задачи: {ex.Message}");

}

}

// Перемещение задачи в другую колонку

public async Task MoveTaskAsync(KanbanTaskViewModel task, KanbanTaskStatus newStatus)

{

if (task == null) return;

try

{

task.Status = newStatus;

await \_taskService.UpdateTaskAsync(task.Model);

BacklogTasks.Remove(task);

RequestTasks.Remove(task);

SelectedTasks.Remove(task);

InProgressTasks.Remove(task);

CompletedTasks.Remove(task);

GetCollectionForStatus(newStatus).Add(task);

}

catch (Exception ex)

{

\_dialogService.ShowError($"Ошибка при перемещении задачи: {ex.Message}");

await LoadTasksAsync();

}

}

// Получение коллекции задач по статусу

private ObservableCollection<KanbanTaskViewModel> GetCollectionForStatus(KanbanTaskStatus status)

{

return status switch

{

KanbanTaskStatus.Backlog => BacklogTasks,

KanbanTaskStatus.Request => RequestTasks,

KanbanTaskStatus.Selected => SelectedTasks,

KanbanTaskStatus.InProgress => InProgressTasks,

KanbanTaskStatus.Completed => CompletedTasks,

\_ => BacklogTasks

};

}

// Загрузка всех задач из базы данных

public async Task LoadTasksAsync()

{

try

{

var loadTasks = new[]

{

\_taskService.GetTasksByStatusAsync(KanbanTaskStatus.Backlog),

\_taskService.GetTasksByStatusAsync(KanbanTaskStatus.Request),

\_taskService.GetTasksByStatusAsync(KanbanTaskStatus.Selected),

\_taskService.GetTasksByStatusAsync(KanbanTaskStatus.InProgress),

Рис. П2.3. Продолжение

\_taskService.GetTasksByStatusAsync(KanbanTaskStatus.Completed)

};

var results = await Task.WhenAll(loadTasks);

// Очищаем существующие коллекции перед добавлением новых данных

BacklogTasks.Clear();

RequestTasks.Clear();

SelectedTasks.Clear();

InProgressTasks.Clear();

CompletedTasks.Clear();

// Добавляем загруженные задачи в коллекции

foreach (var task in results[0].Select(t => new KanbanTaskViewModel(t)))

{

BacklogTasks.Add(task);

}

foreach (var task in results[1].Select(t => new KanbanTaskViewModel(t)))

{

RequestTasks.Add(task);

}

foreach (var task in results[2].Select(t => new KanbanTaskViewModel(t)))

{

SelectedTasks.Add(task);

}

foreach (var task in results[3].Select(t => new KanbanTaskViewModel(t)))

{

InProgressTasks.Add(task);

}

foreach (var task in results[4].Select(t => new KanbanTaskViewModel(t)))

{

CompletedTasks.Add(task);

}

}

catch (Exception ex)

{

\_dialogService.ShowError($"Ошибка при загрузке задач: {ex.Message}");

}

}

public void OnDragStart(object item)

{

// Можно добавить визуальную обратную связь при начале перетаскивания

}

public void OnDragEnd(object item)

{

// Можно добавить визуальную обратную связь при окончании перетаскивания

}

public async void OnDrop(object item, KanbanTaskStatus targetStatus)

{

if (item is KanbanTaskViewModel task && task.Status != targetStatus)

{

await MoveTaskAsync(task, targetStatus);

}

}

public void OnItemDoubleClick(object item)

{

Рис. П2.3. Продолжение

if (item is KanbanTaskViewModel task)

{

StartEditingTask(task);

}

}

// Коллекция активных уведомлений для привязки в View

public ObservableCollection<NotificationViewModel> Notifications => (\_notificationService as NotificationService)?.Notifications;

// Свойства для счетчиков задач (отображаются в заголовках колонок)

public int BacklogTaskCount => BacklogTasks.Count;

public int RequestTaskCount => RequestTasks.Count;

public int SelectedTaskCount => SelectedTasks.Count;

public int InProgressTaskCount => InProgressTasks.Count;

public int CompletedTaskCount => CompletedTasks.Count;

}

}

Текст модуля NotificationView.xaml представлен на рис. П2.4.

<UserControl x:Class="TaskManager.Views.NotificationView"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:TaskManager.Views"

mc:Ignorable="d"

d:DesignHeight="50" d:DesignWidth="300">

<UserControl.Style>

<Style TargetType="UserControl">

<Setter Property="Opacity" Value="0"/>

<Setter Property="RenderTransform">

<Setter.Value>

<TranslateTransform Y="50"/>

</Setter.Value>

</Setter>

<Style.Triggers>

<DataTrigger Binding="{Binding IsVisible}" Value="True">

<DataTrigger.EnterActions>

<BeginStoryboard>

<Storyboard>

<DoubleAnimation Storyboard.TargetProperty="Opacity" To="0.9" Duration="0:0:0.3"/>

<DoubleAnimation Storyboard.TargetProperty="(RenderTransform).(TranslateTransform.Y)" To="0" Duration="0:0:0.3">

<DoubleAnimation.EasingFunction>

<QuadraticEase EasingMode="EaseOut"/>

</DoubleAnimation.EasingFunction>

</DoubleAnimation>

</Storyboard>

</BeginStoryboard>

</DataTrigger.EnterActions>

<DataTrigger.ExitActions>

<BeginStoryboard>

<Storyboard>

Рис П2.4. Текст модуля NotificationView.xaml

Рис. П2.4. Продолжение

<DoubleAnimation Storyboard.TargetProperty="Opacity" To="0" Duration="0:0:0.3"/>

<DoubleAnimation Storyboard.TargetProperty="(RenderTransform).(TranslateTransform.Y)" To="50" Duration="0:0:0.3">

<DoubleAnimation.EasingFunction>

<QuadraticEase EasingMode="EaseIn"/>

</DoubleAnimation.EasingFunction>

</DoubleAnimation>

</Storyboard>

</BeginStoryboard>

</DataTrigger.ExitActions>

</DataTrigger>

</Style.Triggers>

</Style>

</UserControl.Style>

<Border Background="Black" CornerRadius="5" Padding="10">

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center">

<TextBlock Text="❗" VerticalAlignment="Center" FontSize="20" Foreground="Yellow" Margin="0,0,10,0"/>

<TextBlock Text="{Binding Message}" VerticalAlignment="Center" Foreground="White" TextWrapping="Wrap" MaxWidth="250"/>

</StackPanel>

</Border>

</UserControl>

Текст модуля NotificationView.xaml.cs представлен на рис. П2.5.

using System.Collections.ObjectModel;

using TaskManager.ViewModels;

using System.Linq;

namespace TaskManager.Services

{

public class NotificationService : INotificationService

{

private readonly ObservableCollection<NotificationViewModel> \_notifications;

public NotificationService()

{

\_notifications = new ObservableCollection<NotificationViewModel>();

}

public ObservableCollection<NotificationViewModel> Notifications => \_notifications;

public void ShowNotification(string message)

{

// Удаляем старые скрытые уведомления, чтобы коллекция не росла бесконечно

var hidden = \_notifications.Where(n => !n.IsVisible).ToList();

foreach (var notification in hidden)

{

\_notifications.Remove(notification);

}

var notificationViewModel = new NotificationViewModel(message);

\_notifications.Add(notificationViewModel);

notificationViewModel.ShowNotification();

}

}

}

Рис П2.5. Текст модуля NotificationView.xaml

Текст модуля NotificationViewModel.cs представлен на рис. П2.6.

using Prism.Mvvm;

using System;

using System.Threading.Tasks;

using System.Windows.Input;

using Prism.Commands;

using System.Windows.Threading;

namespace TaskManager.ViewModels

{

public class NotificationViewModel : BindableBase

{

private string \_message;

public string Message

{

get => \_message;

set => SetProperty(ref \_message, value);

}

private bool \_isVisible;

public bool IsVisible

{

get => \_isVisible;

set => SetProperty(ref \_isVisible, value);

}

private DispatcherTimer \_timer;

public ICommand CloseCommand { get; }

public NotificationViewModel(string message)

{

Message = message;

IsVisible = false; // Изначально невидимо

CloseCommand = new DelegateCommand(HideNotification);

\_timer = new DispatcherTimer();

\_timer.Interval = TimeSpan.FromSeconds(3);

\_timer.Tick += (s, e) => HideNotification();

}

public void ShowNotification()

{

IsVisible = true;

\_timer.Start();

}

private void HideNotification()

{

\_timer.Stop();

IsVisible = false;

}

}

}

Рис П2.6. Текст модуля NotificationViewModel.cs

Текст модуля KanbanTaskViewModel.cs представлен на рис. П2.7.

using Prism.Mvvm;

using TaskManager.Models;

namespace TaskManager.ViewModels

{

public class KanbanTaskViewModel : BindableBase

{

private readonly KanbanTask \_task;

private bool \_isEditing;

public KanbanTaskViewModel(KanbanTask task)

{

\_task = task;

}

public int Id

{

get => \_task.Id;

set

{

\_task.Id = value;

RaisePropertyChanged();

}

}

public string Title

{

get => \_task.Title;

set

{

\_task.Title = value;

RaisePropertyChanged();

}

}

public string Description

{

get => \_task.Description;

set

{

\_task.Description = value;

RaisePropertyChanged();

}

}

public KanbanTaskStatus Status

{

get => \_task.Status;

set

{

\_task.Status = value;

RaisePropertyChanged();

}

}

public bool IsEditing

{

get => \_isEditing;

set => SetProperty(ref \_isEditing, value);

}

public KanbanTask Model => \_task;

}

}

Рис П2.7. Текст модуля KanbanTaskViewModel.cs

Текст модуля GreaterThanConverter.cs представлен на рис. П2.8.

using System;

using System.Globalization;

using System.Windows.Data;

namespace TaskManager.Converters

{

public class GreaterThanConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value == null || parameter == null)

return false;

if (value is int intValue && parameter is string stringParameter)

{

if (int.TryParse(stringParameter, out int compareValue))

{

return intValue > compareValue;

}

}

return false;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

Рис П2.8. Текст модуля GreaterThanConverter.cs

Текст модуля NumberToVisibilityConverter.cs представлен на рис. П2.9.

using System;

using System.Globalization;

using System.Windows;

using System.Windows.Data;

namespace TaskManager.Converters

{

public class NumberToVisibilityConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value is int number)

{

return number > 0 ? Visibility.Visible : Visibility.Collapsed;

}

return Visibility.Collapsed;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

Рис П2.9. Текст модуля NumberToVisibilityConverter.cs

Текст модуля DragDropBehavior.cs представлен на рис. П2.10.

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using Microsoft.Xaml.Behaviors;

using TaskManager.Models;

namespace TaskManager.Behaviors

{

public interface IDragDropHandler

{

void OnDragStart(object item);

void OnDragEnd(object item);

void OnDrop(object item, KanbanTaskStatus targetStatus);

void OnItemDoubleClick(object item);

}

public class DragDropBehavior : Behavior<ListView>

{

public static readonly DependencyProperty IsEnabledProperty =

DependencyProperty.RegisterAttached(

"IsEnabled",

typeof(bool),

typeof(DragDropBehavior),

new PropertyMetadata(false, OnIsEnabledChanged));

public static bool GetIsEnabled(DependencyObject obj)

{

return (bool)obj.GetValue(IsEnabledProperty);

}

public static void SetIsEnabled(DependencyObject obj, bool value)

{

obj.SetValue(IsEnabledProperty, value);

}

private static void OnIsEnabledChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)

{

if (d is ListView listView)

{

var behavior = new DragDropBehavior();

if ((bool)e.NewValue)

{

Interaction.GetBehaviors(listView).Add(behavior);

}

else

{

var behaviors = Interaction.GetBehaviors(listView);

for (int i = behaviors.Count - 1; i >= 0; i--)

{

if (behaviors[i] is DragDropBehavior)

{

behaviors.RemoveAt(i);

}

}

}

Рис П2.10. Текст модуля DragDropBehavior.cs

Рис. П2.10. Продолжение

}

}

private Point \_startPoint;

private bool \_isDragging;

private object \_draggedItem;

protected override void OnAttached()

{

base.OnAttached();

AssociatedObject.PreviewMouseLeftButtonDown += ListView\_PreviewMouseLeftButtonDown;

AssociatedObject.MouseMove += ListView\_MouseMove;

AssociatedObject.Drop += ListView\_Drop;

AssociatedObject.MouseDoubleClick += ListView\_MouseDoubleClick;

AssociatedObject.AllowDrop = true;

}

protected override void OnDetaching()

{

base.OnDetaching();

AssociatedObject.PreviewMouseLeftButtonDown -= ListView\_PreviewMouseLeftButtonDown;

AssociatedObject.MouseMove -= ListView\_MouseMove;

AssociatedObject.Drop -= ListView\_Drop;

AssociatedObject.MouseDoubleClick -= ListView\_MouseDoubleClick;

}

private void ListView\_PreviewMouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

\_startPoint = e.GetPosition(null);

var listView = sender as ListView;

var item = FindAncestor<ListViewItem>((DependencyObject)e.OriginalSource);

if (item != null)

{

\_draggedItem = item.DataContext;

\_isDragging = true;

if (AssociatedObject.DataContext is IDragDropHandler handler)

{

handler.OnDragStart(\_draggedItem);

}

}

}

private void ListView\_MouseMove(object sender, MouseEventArgs e)

{

if (\_isDragging && e.LeftButton == MouseButtonState.Pressed && \_draggedItem != null)

{

Point mousePos = e.GetPosition(null);

Vector diff = \_startPoint - mousePos;

if (Math.Abs(diff.X) > SystemParameters.MinimumHorizontalDragDistance ||

Math.Abs(diff.Y) > SystemParameters.MinimumVerticalDragDistance)

{

var listView = sender as ListView;

var dragData = new DataObject("KanbanTask", \_draggedItem);

DragDrop.DoDragDrop(listView, dragData, DragDropEffects.Move);

\_isDragging = false;

if (AssociatedObject.DataContext is IDragDropHandler handler)

Рис. П2.10. Продолжение

{

handler.OnDragEnd(\_draggedItem);

}

\_draggedItem = null;

}

}

}

private void ListView\_Drop(object sender, DragEventArgs e)

{

if (e.Data.GetDataPresent("KanbanTask"))

{

var item = e.Data.GetData("KanbanTask");

var targetListView = sender as ListView;

var targetStatus = (KanbanTaskStatus)targetListView.Tag;

if (item != null && AssociatedObject.DataContext is IDragDropHandler handler)

{

handler.OnDrop(item, targetStatus);

}

}

}

private void ListView\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

if (AssociatedObject.SelectedItem != null &&

AssociatedObject.DataContext is IDragDropHandler handler)

{

handler.OnItemDoubleClick(AssociatedObject.SelectedItem);

}

}

private static T FindAncestor<T>(DependencyObject current) where T : DependencyObject

{

do

{

if (current is T ancestor)

{

return ancestor;

}

current = VisualTreeHelper.GetParent(current);

}

while (current != null);

return null;

}

}

}

Текст модуля App.xaml представлен на рис. П2.11.

<?xml version="1.0" encoding="utf-8"?>

<prism:PrismApplication x:Class="TaskManager.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Рис П2.11. Текст модуля App.xaml

Рис. П2.11. Продолжение

xmlns:prism="http://prismlibrary.com/"

xmlns:converters="clr-namespace:TaskManager.Converters">

<Application.Resources>

<ResourceDictionary>

<converters:NumberToVisibilityConverter x:Key="NumberToVisibilityConverter"/>

<converters:GreaterThanConverter x:Key="GreaterThanConverter"/>

<ResourceDictionary.MergedDictionaries>

<!-- Здесь будут стили -->

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Application.Resources>

</prism:PrismApplication>

Текст модуля App.xaml.cs представлен на рис. П2.12.

using Prism.DryIoc;

using Prism.Ioc;

using System;

using System.Windows;

using System.Windows.Threading;

using TaskManager.Data;

using TaskManager.Services;

using TaskManager.Views;

using Microsoft.EntityFrameworkCore;

using TaskManager.ViewModels;

namespace TaskManager

{

// Главный класс приложения

public partial class App : PrismApplication

{

public App()

{

DispatcherUnhandledException += App\_DispatcherUnhandledException;

AppDomain.CurrentDomain.UnhandledException += CurrentDomain\_UnhandledException;

}

// Обработчик необработанных исключений в UI потоке

private void App\_DispatcherUnhandledException(object sender, DispatcherUnhandledExceptionEventArgs e)

{

MessageBox.Show($"Необработанное исключение: {e.Exception.Message}\n\nПодробности: {e.Exception.StackTrace}",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

e.Handled = true;

}

// Обработчик необработанных исключений в других потоках

private void CurrentDomain\_UnhandledException(object sender, UnhandledExceptionEventArgs e)

{

if (e.ExceptionObject is Exception ex)

{

MessageBox.Show($"Критическая ошибка: {ex.Message}\n\nПодробности: {ex.StackTrace}",

"Критическая ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

Рис П2.12. Текст модуля App.xaml.cs

Рис. П2.12. Продолжение

}

// Создание главного окна приложения

protected override Window CreateShell()

{

try

{

var window = Container.Resolve<MainWindow>();

if (window == null)

{

throw new Exception("Не удалось создать главное окно");

}

return window;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при создании главного окна: {ex.Message}",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

Current.Shutdown();

return null;

}

}

// Регистрация сервисов и зависимостей

protected override void RegisterTypes(IContainerRegistry containerRegistry)

{

try

{

// Настройка подключения к базе данных

var dbPath = System.IO.Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "TaskManager", "TaskManager.db");

System.IO.Directory.CreateDirectory(System.IO.Path.GetDirectoryName(dbPath));

var options = new DbContextOptionsBuilder<TaskManagerDbContext>()

.UseSqlite($"Data Source={dbPath}")

.EnableSensitiveDataLogging()

.Options;

// Регистрация контекста базы данных

containerRegistry.RegisterInstance(options);

containerRegistry.Register<TaskManagerDbContext>();

// Регистрация сервиса для работы с задачами

containerRegistry.RegisterSingleton<ITaskService, TaskService>();

containerRegistry.RegisterSingleton<IDialogService, DialogService>();

containerRegistry.RegisterSingleton<INotificationService, NotificationService>();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при регистрации сервисов: {ex.Message}",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

throw;

}

}

// Инициализация приложения

protected override async void OnStartup(StartupEventArgs e)

{

Try

Рис. П2.12. Продолжение

{

base.OnStartup(e);

var context = Container.Resolve<TaskManagerDbContext>();

await context.Database.EnsureCreatedAsync();

// Загрузка данных будет происходить в обработчике Loaded окна MainWindow

var mainWindow = Container.Resolve<MainWindow>();

// var viewModel = mainWindow.DataContext as MainWindowViewModel;

// if (viewModel != null)

// {

// await viewModel.InitializeAsync();

// }

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при инициализации приложения: {ex.Message}",

"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

Current.Shutdown();

}

}

protected override void ConfigureViewModelLocator()

{

base.ConfigureViewModelLocator();

}

}

}

# Приложение 3

Руководство пользователя

1. Общие сведения о программе

Файл программы называется «TaskManager.exe» (рис. П2.1). Данная программа представляет собой task-менеджер в виде канбан-доски. Она позволяет создавать, изменять и удалять задачи, а также перемещать их между колонками канбан-доски.

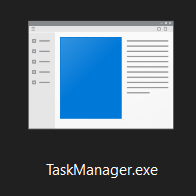


Рис. П2.1. Файл программы

1. Описание установки

Установка программы не требуется, достаточно лишь запустить exe-файл. Сторонних программ для работы также не требуется.

1. Описание запуска

Для запуска программы необходимо дважды нажать на файл TaskManager.exe, либо на его ярлык.

1. Инструкции по работе

Интерфейс программы представлен на рис. П2.2.



Рис. П2.2. Интерфейс программы

Канбан-доска состоит из 5 колонок:

* «Бэклог» – задачи, которые ставит команда;
* «Запросы» – запросы от пользователей;
* «Выбрано» – задачи, которые планируется выполнить (берутся из «Бэклога» и «Запросов»);
* «В работе» – задачи, над которыми сейчас ведется работа;
* «Готово» – выполненные задачи.

Для создания новой задачи необходимо нажать на кнопку «+» в верхней части любой из колонок (рис. П2.3).



Рис. П2.3. Кнопка для создания новой задачи

После этого в выбранной колонке появится карточка новой задачи. На ней будут отображаться 2 поля ввода. Верхнее поле – заголовок задачи, нижнее поле – ее описание. Под карточкой 2 кнопки – «Сохранить» для сохранения задачи и «Отмена» для удаления задачи (рис. П2.4).

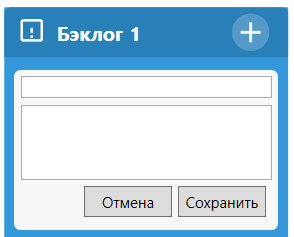


Рис П2.4. Создание новой задачи

Необходимо заполнить соответствующие поля (рис. П2.5).

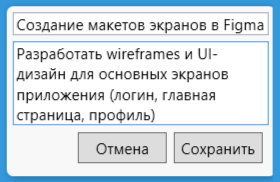


Рис П2.5. Написание текста задачи

После ввода заголовка и описания задачи для ее сохранения необходимо нажать на кнопку «Сохранить» (рис. П2.6). Чтобы стереть написанную задачу, необходимо нажать на кнопку «Отмена».

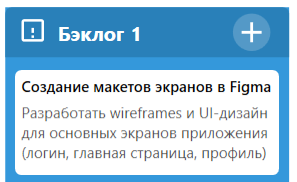


Рис. П2.6. Созданная задача

Для того, чтобы отредактировать или удалить задачу, необходимо дважды по ней нажать левой клавишей мыши. На карточке задачи отобразятся поля и кнопки, аналогично созданию задачи. Однако помимо кнопок «ОК» и «Отмена» также появится третья кнопка «Удалить» (рис. П2.7).

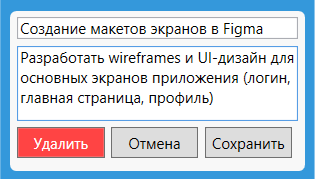


Рис. П2.7. Редактирование задачи

После изменения заголовка или описания задачи для сохранения обновленной задачи необходимо нажать на кнопку «Сохранить». Чтобы отменить внесение изменений, необходимо нажать на кнопку «Отмена». Чтобы удалить задачу, необходимо нажать на кнопку «Удалить».

Для перемещения задачи из одной колонки в другую нужно зажать карточку задачи левой кнопкой мыши и перетащить в нужную колонку, затем отпустить клавишу мыши.

Для выхода из программы необходимо нажать на кнопку с иконкой крестика в правом верхнем углу (рис. П2.8).



Рис. П2.8. Кнопка закрытия программы

1. Сообщения пользователю

Если при создании или редактировании задачи оставить заголовок пустым (рис. П2.9). В правом нижнем углу всплывет оповещение об ошибке с текстом «Заголовок задачи не может быть пустым!» (рис. П2.10).

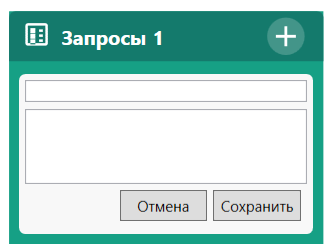


Рис. П2.9. Попытка создания задачи с пустым заголовком

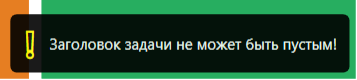


Рис. П2.10. Сообщение об ошибке при создании задачи с пустым заголовком