1. Обоснование выбора

1.1 В качестве системы контроля версий для WPF C# проекта, часто используются Git,. Git поддерживает ветвление и слияние, что позволяет удобно управлять различными функциональными изменениями, экспериментировать с новыми функциями без воздействия на основной код и объединять изменения в основную ветку разработки. Выбор для нашего проекта обоснован тем, что в целом, Git обеспечивает удобство и гибкость, позволяя эффективно управлять версиями кода и совместно работать над проектами.

1.2 Декомпиляция C# кода может быть выполнена с использованием инструмента, таким как dotPeek. Данный инструмент может быть полезен для работы с уже существующими программами, позволяя проанализировать исходный код, извлечь ресурсы или провести обратную инженерию. Понимание декомпиляции также помогает защитить код от нежелательного доступа и вмешательства. DotPeek обладает удобным интерфейсом и поддерживает множество функций, включая поиск по коду, навигацию по проекту, анализ зависимостей, быструю загрузку исходного кода и поддержку различных типов .NET-сборок. В данном случае, наш выбор обснован в следующем, DotPeek представляет собой мощный инструмент, который облегчает работу с .NET-кодом и позволяет более подробно изучать внутреннюю реализацию приложений.

2. Создание среды

2.2 Тестовое приложение

2.2.1 Техническое задание

Цель проекта:  
- Создать игру "Морской бой" с использованием Windows Presentation Foundation (WPF) при помощи паттерна MVVM, в которой игрок будет соревноваться с компьютером в уничтожении кораблей.  
  
Основные функции игры:  
1. Начало игры:  
   - Игрок должен иметь возможность создать свою игровую доску размером 10x10 клеток и разместить на ней свои корабли.  
   - Корабли имеют различную длину: от 1 до 4 клеток.  
   - Игра должна поддерживать возможность случайной генерации расположения кораблей на доске игрока.  
  
2. Отображение игровых досок:  
   - Для отображения игровых досок будет использоваться графическое окно WPF.  
   - Каждая клетка доски будет представлена визуально с помощью элементов WPF, чтобы отразить ее состояние (пустая, попадание, промах).  
   - Визуальное отображение будет меняться в зависимости от хода игры.

3. Ход игры:  
   - Игрок и компьютер будут ходить поочередно.  
   - Игрок с помощью интерфейса WPF выбирает клетку для атаки на доске компьютера.  
   - Компьютер генерирует случайные координаты клетки для атаки на доске игрока.  
   - При попадании в корабль, клетка окрашивается в другой цвет или отображается другим элементом интерфейса.  
   - При промахе, клетка также будет визуально отображаться по-разному (в виде крестика).  
  
4. Конец игры:  
   - Игра продолжается до полного уничтожения всех кораблей либо на доске игрока, либо на доске компьютера.  
   - После окончания игры должна быть предоставлена возможность начать новую игру или выйти из приложения с помощью интерфейса WPF.  
  
5. Управление игрой:  
   - Игра будет иметь навигационное меню с кнопками для начала новой игры, сохранения и загрузки игры, а также выхода из приложения.  
   - Возможность изменения настроек игры.  
  
Требования к реализации:  
- Игра должна быть разработана на языке C# с использованием Windows Presentation Foundation (WPF) для создания графического интерфейса.  
- Игровые доски, клетки и элементы интерфейса должны быть стилизованы с помощью XAML.  
- Код должен быть хорошо структурирован, модульный и использовать объектно-ориентированный подход для обеспечения легкости сопровождения и расширяемости.  
  
Дополнительные требования:  
- Обеспечить валидацию ходов игрока, чтобы игрок не мог совершать некорректные действия (например, выбирать уже атакованные клетки).  
- Обработка случаев, когда игрок пытается атаковать за пределами игровой доски или при попытке сохранения или загрузки игры в несуществующий файл.  
- Возможность ввода имени игрока в интерфейсе WPF, и его имя должно отображаться в игре.