МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«Новосибирский национальный исследовательский государственный университет»

(Новосибирский государственный университет, НГУ)

Структурное подразделение Новосибирского государственного университета – Высший колледж информатики Университета (ВКИ НГУ)

КАФЕДРА ИНФОРМАТИКИ

РАЗРАБОТКА ДЕСКТОПНОГО ПРИЛОЖЕНИЯ ДЛЯ ОС WINDOWS «Система администрирования для НРИ D&D»

Квалификация программист

|  |  |
| --- | --- |
| Руководитель | Пауль С.А.  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2023 г. |
| Студент 3 курса  гр. 107в2 | Сидоров А.А.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2023 г. |

Новосибирск

2023

**СОДЕРЖАНИЕ**

Оглавление

[ВВЕДЕНИЕ 5](#_Toc154784492)

[ПОСТАНОВКА ЗАДАЧИ ВКР 9](#_Toc154784493)

[Пользовательские требования 10](#_Toc154784494)

[Системные требования 12](#_Toc154784495)

[Требования к графическому пользовательскому интерфейсу 12](#_Toc154784496)

[План-график выполнения ВКР 13](#_Toc154784497)

[АНАЛИЗ ТРЕБОВАНИЙ И ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ 15](#_Toc154784498)

[Описание предметной области задачи ВКР 15](#_Toc154784499)

[Информационные объекты предметной области и взаимосвязи между ними 15](#_Toc154784500)

[Информационные и функциональные потребности пользователей разрабатываемой ПС (ПМ) 16](#_Toc154784501)

[Обзор существующих программных реализаций решения задачи 17](#_Toc154784502)

[Концептуальное обоснование разработки 18](#_Toc154784503)

[Классы и характеристики пользователей 19](#_Toc154784504)

[Функциональные требования 20](#_Toc154784505)

[Описание прецедентов 21](#_Toc154784506)

[ВЫБОР ПРОГРАММНЫХ СРЕД И СРЕДСТВ РАЗРАБОТКИ 24](#_Toc154784507)

[Сравнительный анализ имеющихся возможностей по выбору средств разработки 24](#_Toc154784508)

[Характеристика выбранных программных сред и средств 26](#_Toc154784509)

[АЛГОРИТМ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ 28](#_Toc154784510)

[Этапы реализации ПС (ПМ) 28](#_Toc154784511)

[Пользовательский интерфейс ПС (ПМ) 29](#_Toc154784512)

[Взаимодействие пользователей с ПС (ПМ) 29](#_Toc154784513)

[Проектирование пользовательских сценариев 30](#_Toc154784514)

[Определение операций пользователей 31](#_Toc154784515)

[Проектирование структуры экранов и схема функционирования экранов 32](#_Toc154784516)

[Входные, выходные и промежуточные данные 33](#_Toc154784517)

[Архитектура ПС (ПМ) 35](#_Toc154784518)

[ТЕСТИРОВАНИЕ И ОПТИМИЗАЦИЯ 51](#_Toc154784519)

[План тестирования 51](#_Toc154784520)

[Результаты тестирования 52](#_Toc154784521)

[Оптимизация ПС (ПМ) 52](#_Toc154784522)

[РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 54](#_Toc154784523)

[ЗАКЛЮЧЕНИЕ 59](#_Toc154784524)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 60](#_Toc154784525)

[ПРИЛОЖЕНИЯ 61](#_Toc154784526)

[Приложение А: MainWindow.xaml.cs 61](#_Toc154784527)

[Приложение Б: MainWindow.xaml 61](#_Toc154784528)

[Приложение Е: Template. Cs 95](#_Toc154784529)

[Приложение Ж: Character. cs 102](#_Toc154784530)

[Приложение З: Dice. cs 106](#_Toc154784531)

[Приложение И: Skill. cs 106](#_Toc154784532)

[Приложение К: Stat. cs 110](#_Toc154784533)

[Приложение М: Dexterity. cs 114](#_Toc154784534)

[Приложение Н: Constitution. cs 116](#_Toc154784535)

[Приложение О: Intelligence. cs 116](#_Toc154784536)

[Приложение П: Wisdom. cs 118](#_Toc154784537)

[Приложение Р: Charisma. cs 120](#_Toc154784538)

[Приложение С: Elf. cs 122](#_Toc154784539)

[Приложение Т: Goblin. cs 123](#_Toc154784540)

[Приложение У: HalfOrc. cs 123](#_Toc154784541)

[Приложение Ф: HighElf 123](#_Toc154784542)

[Приложение Х: WoodElf 124](#_Toc154784543)

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫХ ОБОЗНАЧЕНИЙ И ТЕРМИНОВ

Программное средство (ПС) – это совокупность программных продуктов, которые выполняют определенные функции или решают конкретные задачи.

Операционная система (ОС) – это программное обеспечение, которое управляет аппаратными и программными ресурсами компьютера или другого устройства.

C Sharp (C#) – это объектно-ориентированный язык программирования общего назначения.

Геймер – это термин, который используется для обозначения человека, увлекающегося компьютерными или видеоиграми.

Кастомизация – это процесс настройки и изменения внешнего вида, характеристик или поведения объекта в игре в соответствии с предпочтениями и желаниями игрока.

Игровой персонаж – это виртуальный персонаж, созданный для взаимодействия с игровым миром.

Игровой мир – это виртуальная среда или область, в которой развивается действие видеоигры.

Компьютерные видеоигры – это форма интерактивного развлечения, предназначенная для запуска и воспроизведения на компьютере.

Dungeons and Dragons (D&D) – это настольная ролевая игра, созданная Гари Гайгексом и Дэйвом Арнесоном.

НРИ – Настольная ролевая игра.

Информационный объект – класс, построенный на основе обобщения различных объектов, процессов и моделей, связанных с информационными описаниями, процессами и технологиями, и отношений между ними.

ВВЕДЕНИЕ

Современный мир проживает эпоху стремительного развития цифровых технологий, в которой компьютерные приложения, а особенно видеоигры, играют важную роль в формировании культуры и самовыражении личности.

С ростом популярности компьютерных игр особенно заметен тренд к кастомизации персонажей. Игроки стремятся не только пройти увлекательные сценарии, но и взаимодействовать с игровым миром через уникальные и персонализированные персонажи. Кастомизация открывает дополнительные возможности для самовыражения, позволяя игрокам воплотить свои творческие идеи и фантазии в виртуальной реальности.

Одним из ключевых аспектов актуальности является индивидуальность геймера в виртуальном пространстве видеоигр. В современном обществе игровая культура стала неотъемлемой частью личной идентичности многих людей. Виртуальные миры видеоигр предоставляют возможность создать уникального персонажа, выразить свою индивидуальность и представить себя таким, каким хочется быть в виртуальной реальности.

Еще одной из актуальных сфер в информационной технологии является разработка приложений для управления персонажами в играх, особенно в контексте популярных настольных ролевых игр, таких как Dungeons & Dragons (D&D). Отслеживание инициативы персонажей в таких играх – это важный этап, которым позволяет отслеживать ход боя.

Актуальность данной работы обусловлена не только удовлетворением запросов любителей настольных ролевых игр, но и более широким влиянием цифровых инноваций на формирование современной культуры. Интеграция технологий в развлекательные процессы становится неотъемлемой частью нашей жизни, и разработка программ, улучшающих игровой опыт, имеет существенное значение для современного общества.

В этом контексте создание десктопного приложения для создания персонажей в настольной ролевой игре Dungeons & Dragons (D&D) не только соответствует техническим трендам, но и обладает высокой социокультурной актуальностью. Создание приложения, позволяющего пользователю отслеживать инициативу персонажей в игре D&D. Это не только предоставляет возможность правильно распределить ресурсы пользователя на более важные аспекты игры.

Данный курсовой проект направлен на разработку десктопного приложения для операционной системы Windows, которое позволит упростить и сделать более увлекательным процесс отслеживания боя в игре D&D.

Однако, за пределами узкоспециализированной темы отслеживания инициативы персонажей, стоит заметить, что разработка подобных программ имеет глубокие корни в общем контексте роста цифровых технологий и их влияния на развлекательную и игровую индустрию.

Данная работа также рассматривает вопросы удобства пользовательского интерфейса, что подчеркивает общую тенденцию к созданию приложений, ориентированных на интуитивное взаимодействие с пользователем.

Наконец, анализ предметной области и обоснование необходимости разработки данного приложения требует рассмотрения существующих подходов и решений в данной области. В ходе выполнения курсовой работы будут проанализированы существующие продукты и определены преимущества предлагаемого приложения.

Цель курсовой работы – разработка десктопного приложения для ОС Windows «Система администрирования для НРИ D&D». Следовательно, чтобы добиться достижения поставленной цели нужно выполнить определённый ряд работ, среди которых можно выделить:

1. Изучение основных аспектов предметной области, осуществление исследования для более полного понимания требований и характеристик, связанных с отслеживанием инициативы.
2. Проведение анализа существующих аналогичных приложений с целью выявления их особенностей, преимуществ и недостатков. Подробное изучение функциональности и интерфейса, с целью выявления наилучших практик и уникальных черт для использования в разрабатываемом приложении.
3. Разработка прототипов пользовательского интерфейса с учетом удобства использования. Исследование современных дизайн-тенденций и применение их в разработке интерфейса для повышения привлекательности и интуитивной понятности.
4. Написание кода на языке программирования C# с использованием технологии WPF. Этот код должен включать в себя логику отслеживания состояния персонажей их порядка по инициативе. Также, в коде будет предусмотрено взаимодействие с пользовательским интерфейсом для отображения этих характеристик.
5. Тестирование и отладка. Планирование и проведение тестирования приложения на различных этапах разработки. Выявление и устранение ошибок и недочетов в функциональности и интерфейсе приложения.
6. Написание руководства пользователя. Подготовка подробного руководства, описывающего процесс использования разработанного приложения. Руководство должно включать в себя шаги по управлением интерфейса. Также, следует предоставить информацию об основных возможностях приложения и способах взаимодействия с интерфейсом.
7. Написание пояснительной записки к курсовой работе. Создание подробной пояснительной записки, в которой освещаются все этапы разработки приложения. Это включает в себя описание выбранных технологий, методов и средств разработки, анализ предметной области, обоснование принятых решений по проектированию интерфейса и функционала, а также результаты тестирования и отладки. Также, в пояснительной записке следует выделить основные преимущества разработанного приложения, анализировать полученный опыт и предлагать возможные направления для дальнейшего улучшения.

Таким образом, создание десктопного приложения для отображения и управление инициативой персонажа в игре D&D является не только техническим проектом, но и вкладом в общий контекст развития цифровых технологий, повышения пользовательского опыта и удовлетворения запросов современного поколения, активно взаимодействующего с цифровым миром. Создание приложения для управления персонажами в D&D выходит за рамки простой технической задачи. Это важный шаг в поддержке индивидуальности и самовыражении геймеров в виртуальных мирах, что в современном мире приобретает значимость и влияет на формирование цифровой культуры.

ПОСТАНОВКА ЗАДАЧИ ВКР

Данный проект нацелен на создание удобного десктопного приложения "Система администрирования для НРИ D&D" на операционную систему Windows. Основной задачей разработки является предоставление игрокам быстрого и простого инструмента для эффективного управления игрой в реальном мире и устранения возможных барьеров для новых участников.

Ключевые задачи, направленные на достижение этой цели, включают в себя:

1. **Отображение показателей персонажа:**
   1. Разработать логику для отображения информации о персонаже на вкладке "Character".
   2. Реализовать отображение значений характеристик на вкладке "Stat".
2. **Инициализация броска инициативы для вхождения в бой:**
   1. Создать логику для броска инициативы на основе характеристик персонажа при нажатии кнопки "бросок инициативы".
3. **Управление ходом:**
   1. Реализовать функционал кнопки "Закончить ход" для завершения хода персонажа.
4. **Обработка ошибок:**
   1. Выводить информативные сообщения об ошибках или подсказки для улучшения пользовательского опыта.
5. **Оформление и стиль:**
   1. Работать над стилем и дизайном интерфейса для повышения его привлекательности и интуитивной понятности.
   2. Разработать логику обновления и отображения элементов интерфейса, чтобы сделать их интуитивно понятными и легко воспринимаемыми.
6. **Тестирование и отладка:**
   1. Провести тестирование на различных этапах разработки для выявления и устранения ошибок в функциональности и интерфейсе.
   2. Обеспечить стабильную работу приложения.
7. **Разработка руководства пользователя:**
   1. Создать подробное руководство, содержащее инструкции по использованию приложения.

Проект стремится не только к технической реализации функционала, но и к созданию приложения, которое соответствует современным требованиям пользователя, предоставляя удовлетворительный опыт игры в мире Dungeons & Dragons.

Пользовательские требования

Пользовательские требования определяют функциональности, которые пользователи ожидают от приложения «Система администрирования для НРИ D&D» с целью достижения своих целей в контексте управления персонажами в игре Dungeons & Dragons.

В эту область также включаются характеристики приложения, существенные для обеспечения удовлетворения потребностей пользователей и обеспечения им приятного и продуктивного взаимодействия с приложением.

1. **Выбор персонажа:**
   1. Пользователь должен иметь возможность выбирать персонажей из представленного списка.
   2. При выборе персонажа, приложение должно отображать подробную информацию о нем.
2. **Инициатива в бою:**
   1. Предоставить пользователю возможность проводить броски инициативы для персонажей.
   2. Отображать результаты бросков инициативы для каждого персонажа.
3. **Управление ходом боя:**
   1. Реализовать функционал для завершения хода персонажа.
   2. При завершении хода персонажа, обновлять порядок персонажей в бою.
4. **Логирование событий:**
   1. Предоставить возможность вести журнал событий боя.
   2. Отображать важные события и броски костей в журнале.
5. **Добавление в бой новых персонажей:**
   1. Реализовать возможность добавления новых персонажей в бой.
   2. При добавлении персонажа, обновлять порядок персонажей в бою.
6. **Учет состояния персонажей:**
   1. Вести учет текущих хит-поинтов и состояния каждого персонажа в бою.
   2. В случае смерти персонажа, предоставить соответствующее уведомление.
7. **Обновление данных персонажей:**
   1. Пользователь должен иметь возможность обновлять данные персонажей, такие как их характеристики и текущие хит-поинты.
   2. Обновление данных должно автоматически отражаться в порядке персонажей в бою.
8. **Логирование смерти персонажей:**
   1. Вести учет смертей персонажей и отображать соответствующие уведомления.
   2. При наступлении смерти, предоставить опции для обработки этого события.

Разработка десктопного приложения «Система администрирования для НРИ D&D» представляет собой значимый шаг в направлении улучшения пользовательского опыта и облегчения процесса создания персонажей для игроков настольной ролевой игры Dungeons & Dragons (D&D).

Системные требования

В контексте приложения «Система администрирования для НРИ D&D», системные требования определяют технические характеристики и условия, которые необходимы для корректной установки, функционирования и использования вашего программного продукта.

Эти требования включают в себя:

1. Платформа. Приложение разработано для операционной системы Windows 8 и больше.
2. Память. Рекомендуемый объем оперативной памяти для стабильной работы приложения – не менее 4 ГБ.
3. Процессор. Процессор с тактовой частотой не менее 2.0 ГГц.
4. Дисковое пространство. Свободное дисковое пространство для установки приложения и сохранения данных – не менее 200 МБ.
5. Разрешение экрана. Рекомендуемое разрешение экрана – не менее 1280x720 пикселей.
6. Надежность. Приложение должно обеспечивать стабильную работу без существенных сбоев и зависаний.
7. Поддержка разрядности. Приложение поддерживает 64-разрядные версии операционной системы Windows.

Требования к графическому пользовательскому интерфейсу

В контексте приложения «Система администрирования для НРИ D&D», требования к графическому пользовательскому интерфейсу определяют, каким образом пользователи будут взаимодействовать с приложением. Эти требования включают в себя:

1. Внешний вид интерфейса:
   1. Описание дизайна и стиля интерфейса, который должен быть интуитивно понятным для пользователей D&D, включая опытных игроков и новичков.
2. Навигация:
   1. Обеспечение удобной навигации по приложению, чтобы пользователи могли легко находить нужные функции.
   2. Размещение элементов управления (кнопки, поля ввода) в логичных и интуитивно понятных местах.
3. Взаимодействие:
   1. Определение способов взаимодействия пользователя с приложением (например, клики).
   2. Предоставление обратной связи пользователю при выполнении определенных действий.
4. Доступность функциональности:
   1. Гарантирование, что все функциональные возможности приложения доступны через пользовательский интерфейс.
   2. Понятные и информативные метки, подсказки и инструкции для облегчения понимания возможностей приложения.

Таким образом, пользовательский интерфейс приложения разрабатывается с учетом легкости освоения, что делает его доступным как для опытных игроков, так и для новичков.

План-график выполнения ВКР

Ниже продемонстрирован план-график выполнения курсовой работы (табл.1).

Таблица 1 – План-график выполнения курсовой работы

| Дата | Задача |
| --- | --- |
| 20.11.2023 | Проведение анализа предметной области и изучение требований. |
| 26.11.2023 | Исследование аналогичных приложений и выявление их особенностей. |
| 01.12.2023 | Разработка прототипов пользовательского интерфейса. |
| 07.12.2023 | Начало написания кода на C# с использованием WPF. |
| 14.12.2023 | Разделение основных Характеристик на отдельные классы, для обеспечения быстрого доступа к ним. |
| 15.12.2023 | Внедрение привязок данных между объектами MainWindow и элементами управления. |
|  |  |
| 20.12.2023 | Удаление возможности создавать Персонажа.  Причина: Неправильная реализация работы с классами классов персонажей и рас и некорректная работа передачи экземпляра из CreateWindow в MainWindow. |
| 24.12.2023 | Полная Реализация функций теркера инициативы. |
| 25.12.2023 | Тестирование и отладка приложения. |
| 26.12.2023 | Подготовка пояснительной записки и руководства пользователя. |

Таким образом, план-график предполагает последовательное выполнение основных этапов работы над проектом в течение декабря 2023 года. Задачи включают в себя анализ, исследование, разработку, программирование, тестирование и завершение проекта с подготовкой документации.

АНАЛИЗ ТРЕБОВАНИЙ И ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИЙ

Описание предметной области задачи ВКР

Современные исследования подтверждают, что компьютерные игры играют важную роль в жизни многих людей, предоставляя развлечение, средство самовыражения и социализации. В этом контексте настольные ролевые игры, такие как Dungeons & Dragons (D&D), занимают особенное место, предоставляя игрокам уникальный опыт управления персонажами в вымышленных мирах.

Одним из ключевых аспектов в предметной области проектируемого приложения является управление боевыми сценами в D&D, особенно отслеживание инициативы персонажей. Это важный этап, влияющий на ход боя и требующий эффективного и удобного инструмента для игроков.

Проект направлен на решение проблем, с которыми сталкиваются игроки D&D при проведении боевых сцен, и предоставление им интуитивного трекера инициативы. Важными аспектами будут не только отслеживание порядка хода персонажей, но и создание удобного интерфейса для визуализации и управления инициативой. Это позволит игрокам более полно наслаждаться процессом сражений, делая его более динамичным и погружающим.

Проект также учитывает тенденции в развитии игровой индустрии и растущий интерес к настольным ролевым играм, что подчеркивает актуальность разработки трекера инициативы для проведения боевых сцен в D&D5e.

Информационные объекты предметной области и взаимосвязи между ними

В предметной области создания программного средства, направленного на кастомизацию и индивидуализацию игроков в настольных играх, информационные объекты представляют собой разнообразные элементы, процессы и модели. Эти элементы тесно связаны с описаниями, процессами и технологиями, активно применяемыми в сфере визуальных решений и развития креативного мышления. В рамках предметной области «Система администрирования для НРИ D&D» выделяются разнообразные информационные объекты, взаимосвязанные между собой.

Выделим следующие информационные объекты:

1. Графические компоненты:
   1. Визуальные элементы, предоставляющие пользователю показания персонажей.
   2. Визуальные элементы, предоставляющие пользователю показания в ленте инициативы.
2. Технологии визуализации:
   1. Специфические методы и средства для визуализации персонажей и их характеристик.
3. Интерфейс взаимодействия:
   1. Элементы, обеспечивающие взаимодействие пользователя с программным средством.
   2. Примеры: Удобные панели управления, интуитивные элементы интерфейса.

Эти информационные объекты формируют сложную систему, обеспечивая качественную индивидуализацию игрового опыта в настольных играх.

Информационные и функциональные потребности пользователей разрабатываемой ПС (ПМ)

Исходя из предметной области, пользователи разрабатываемой программной системы (ПС) имеют следующие информационные и функциональные потребности:

1. Информационные потребности:
   1. Получение информации о доступных характеристиках, расах и классах.
   2. Ознакомление с описаниями персонажей.
   3. Возможность просмотра его характеристик и текущего состояния.
2. Функциональные потребности:
   1. Проброс его инициативы.
   2. Визуализация ленты инициативы боя, в котором участвуют персонажи.
   3. Возможность редактирования персонажа.

Эти потребности обуславливают функциональность приложения, которая должна быть интуитивной и удовлетворять требованиям разнообразных пользователей.

Обзор существующих программных реализаций решения задачи

В настоящее время существует одно широко известное программное решение, направленных на Администрацию игрового опыта в настольных играх.

Ниже представлен обзор одного аналога с указанием их достоинств и недостатков.

Tentaculus – сайт для создания отслеживания инициативы.

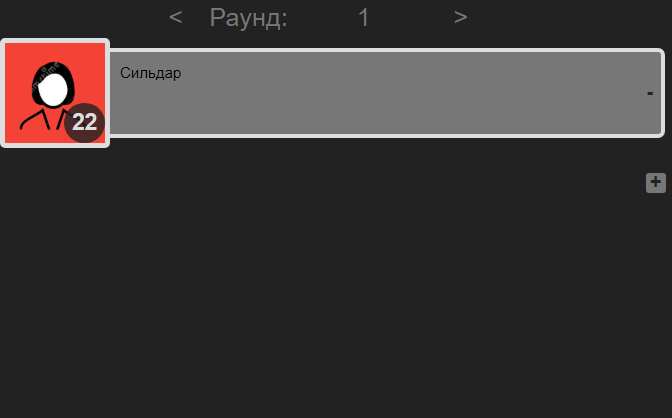


Рисунок №1 – интерфейс Tentaculus

Достоинства:

1. Минималистичный интуитивно понятный интерфейс.

Недостатки:

1. Слишком маленький функционал.
2. Неудобный интерфейс.

Существующие программное реализация предлагает односторонний подход к управлению инициативой персонажей в бою. Новое программное средство, разрабатываемое в рамках ВКР, будет уникальным за счет комбинации удобства взаимодействия, обширных функциональных возможностей и отсутствия ограничений в использовании основных функций.

Концептуальное обоснование разработки

Концептуальное обоснование разработки программного средства для администрации игрового опыта в настольных играх базируется на тщательном анализе проблем в предметной области. После изучения вышеописанных решений, выявляются определенные ограничения и недостатки, которые могут быть решены через новый подход.

Программное средство, разрабатываемое в данной работе, стремится объединить преимущества управления состояния персонажа во время боевых действий. Ключевой идеей является расширение функциональности.

Разработка нового программного средства обоснована потребностью в более гибком, разнообразном и индивидуализированном подходе к взаимодействию с персонажами.

Классы и характеристики пользователей

В рамках разработки программного средства, направленного на администрирование и управление состоянием персонажей в бою в настольных играх, выделяются следующие пользовательские роли:

1. Администраторы:
   1. Опыт в администрировании и управлении боями в настольных играх.
   2. Знание правил и механик боевых действий.
   3. Опыт в настройке параметров и характеристик персонажей.
2. Мастера игры:
   1. Умение организовывать и вести боевые сценарии.
   2. Творческий подход к созданию уникальных боевых ситуаций.
   3. Опыт в адаптации правил под конкретные игровые миры.
3. Разработчики:
   1. Навыки программирования и разработки ПО.
   2. Понимание технических аспектов создания приложений для  
      управления состоянием персонажей в бою.

Каждая из этих пользовательских ролей обладает своими уникальными характеристиками и выполняет определенные задачи в контексте использования разрабатываемого программного средства. Это разнообразие ролей позволит удовлетворить потребности широкого круга пользователей и сделать программу более привлекательной для различных категорий игровых мастеров и администраторов боевых сценариев.

Функциональные требования

Функциональные требования определяют поведение программной системы и набор задач, которые она должна выполнять. В контексте разрабатываемого приложения для администрации и управления состоянием персонажей в бою, определяются следующие функциональные требования.

Актеры:

Администратор: Лицо, использующее приложение для управления персонажами в бою.

Система: Приложение для администрации и управления состоянием персонажей в бою.

Варианты использования:

Добавление персонажа в бой:

Описание: Администратор выбирает персонажа и добавляет его в текущий бой.

Действие системы: Добавление персонажа в список участников боя.

Управление характеристиками в бою:

Описание: Администратор вносит изменения в характеристики персонажей во время боя.

Действие системы: Обновление характеристик персонажей в реальном времени.

Завершение хода персонажа:

Описание: Администратор завершает ход текущего персонажа.

Действие системы: Переход к следующему персонажу в порядке инициативы.

Отслеживание состояния персонажей:

Описание: Система автоматически отслеживает текущее состояние персонажей в бою.

Действие системы: Визуализация текущего состояния персонажей, их здоровья и других характеристик.

Управление списком персонажей:

Описание: Администратор добавляет, удаляет или редактирует персонажей в общем списке.

Действие системы: Обновление списка персонажей в соответствии с внесенными изменениями.

Выполнение вышеперечисленных функциональных требований играет ключевую роль в обеспечении функциональности и удовлетворении потребностей администраторов при управлении состоянием персонажей в бою. Создание системы, обеспечивающей удобное взаимодействие с персонажами, позволит эффективно администрировать игровые сценарии и обеспечивать качественный игровой опыт.

Описание прецедентов

1. Создание боевого состояния персонажа:

Администратор запускает процесс управления боевым состоянием персонажей, необходимым для эффективного ведения боевых сценариев в приложении.

Основной сценарий:

* 1. Администратор открывает раздел управления боевым состоянием персонажей в приложении.
  2. Администратор выбирает персонажа из списка.
  3. Администратор осуществляет необходимые действия, такие как изменение текущих хит-поинтов, статуса жив/мертв и других характеристик персонажа.

1. Обновление состояния персонажа в бою:

Администратор актуализирует состояние персонажей в боевых сценариях, обновляя информацию в приложении.

Основной сценарий:

* 1. Администратор выбирает персонажа в списке участников боя.
  2. Администратор вносит изменения в текущее состояние персонажа,  
     такие как урон, изменение статуса и другие параметры.
  3. Система автоматически обновляет интерфейс, отображая актуальную информацию о состоянии персонажа в бою.

1. Управление инициативой:

Администратор осуществляет управление порядком действий персонажей в боевой сцене, определяя инициативу.

Основной сценарий:

* 1. Администратор просматривает список персонажей в боевой сцене.
  2. Администратор назначает инициативу каждому персонажу в соответствии с правилами игры.
  3. Система автоматически обновляет порядок хода персонажей в интерфейсе.

1. Завершение хода персонажа:

Администратор осуществляет завершение хода персонажа, учитывая возможные изменения состояния, такие как урон и спасброски от смерти.

Основной сценарий:

* 1. Администратор выбирает персонажа, чей ход завершается.
  2. Администратор вносит необходимые изменения в состояние персонажа после завершения хода.
  3. Система автоматически обновляет интерфейс, отображая актуальную информацию о состоянии персонажа.

1. Обработка смерти и спасбросков:

Администратор управляет смертью персонажей, принимая решения о спасбросках и обработке смертельных исходов.

Основной сценарий:

* 1. Администратор выявляет персонажа с нулевыми хит-поинтами.
  2. Администратор принимает решение о спасбросках от смерти в соответствии с правилами игры.
  3. Система автоматически обрабатывает результаты спасбросков и обновляет состояние персонажа в интерфейсе.

Все прецеденты предоставляют администратору функциональность по управлению состоянием персонажей в боевых сценариях приложения.

Начало формы

ВЫБОР ПРОГРАММНЫХ СРЕД И СРЕДСТВ РАЗРАБОТКИ

Для разработки десктопного приложения «Система администрирования для НРИ D&D» были выбраны следующие программные средства и фреймворки: Visual Studio, C#, WPF, а также XAML. Выбор программных сред разработки оправдан использованием Visual Studio. Это обширное и мощное инструментальное средство, которое обеспечивает высокую производительность разработки, интеграцию с WPF и XAML, а также широкие возможности для проектирования и отладки приложений. Важными факторами при выборе были также богатая документация и поддержка со стороны сообщества разработчиков.

Сравнительный анализ имеющихся возможностей по выбору средств разработки

Visual Studio. Среда разработки, сделанная компанией Microsoft, Visual Studio предоставляет разработчикам широкий набор инструментов и возможностей для написания, отладки, тестирования и развертывания программ.

Достоинства:

1. Обширные возможности разработки.
2. Поддержка C# и XAML с высоким уровнем интеграции.
3. Большое сообщество и обширная документация.

Недостатки:

1. Занимает много места на диске.
2. Может потребовать времени для освоения всех функций.

C# (C Sharp). Язык программирования, разработанный компанией Microsoft.

Достоинства:

1. Объектно-ориентированный подход: Удобство организации кода и взаимодействия с объектами.
2. Интеграция с платформой .NET: Возможность использования библиотек и фреймворков, упрощающих разработку.

Недостатки:

1. Ограничение на платформу Microsoft: Язык ориентирован на платформу Microsoft, что может создать зависимость от её технологий.

XAML (Extensible Application Markup Language). Предоставляет декларативный способ создания графических пользовательских интерфейсов, позволяя разработчикам описывать структуру и внешний вид UI, а также взаимодействие между элементами.

Достоинства:

1. Декларативный подход: Удобство создания и изменения интерфейса без необходимости программирования.
2. Интеграция с WPF: Возможность создания богатого и гибкого пользовательского интерфейса.

Недостатки:

1. Не предоставляет функциональности программирования.
2. XAML не является языком программирования, и для логики приложения все равно требуется использование C# или другого языка.

WPF (Windows Presentation Foundation). WPF является частью платформы .NET и предоставляет разработчикам мощные инструменты для создания интерактивных и стильных приложений.

Достоинства:

1. Гибкий пользовательский интерфейс:
2. Возможность создания сложных и красочных интерфейсов.
3. Интеграция с XAML и C#:
4. Способствует разделению дизайна и логики приложения.

Недостатки:

1. Требует изучения.
2. WPF имеет свои особенности, и разработчику может потребоваться время для их освоения.

Характеристика выбранных программных сред и средств

Visual Studio – это мощная интегрированная среда разработки, предоставляющая все необходимые инструменты для создания WPF-приложений на языке C# с использованием XAML.

C# (C Sharp) – это объектно-ориентированный язык программирования, разработанный для создания приложений на платформе Microsoft. В контексте приложения "Система администрирования для НРИ D&D ", C# используется для программирования логики приложения, взаимодействия с интерфейсом и обработки данных.

Windows Presentation Foundation (WPF) – это фреймворк для построения приложений с богатым графическим интерфейсом пользователя. Он предоставляет средства для разработки инновационных и стильных приложений под Windows, а также интегрируется с языком C# и языком разметки XAML. Это технология, предоставляющая средства для создания десктопных приложений с богатым и гибким пользовательским интерфейсом. В контексте приложения " Система администрирования для НРИ D&D ", WPF используется для реализации визуальной части приложения, взаимодействия с XAML и обеспечения пользовательского опыта.

XAML (Extensible Application Markup Language) – это декларативный язык разметки, используемый для создания пользовательского интерфейса в приложениях, разработанных под управлением технологии WPF. В контексте приложения «Система администрирования для НРИ D&D», XAML используется для описания структуры и внешнего вида пользовательского интерфейса.

Таким образом, выбор программных сред и средств (Visual Studio, C#, XAML, WPF) обоснован стремлением к созданию десктопного приложения с современным и привлекательным интерфейсом для удобства пользователя. Visual Studio предоставляет всесторонние возможности разработки и интеграции с технологиями, а C#, XAML и WPF обеспечивают эффективные средства для создания функционального и графически привлекательного приложения. Сочетание этих инструментов позволяет достичь баланса между производительностью разработки, мощью функционала и качеством пользовательского интерфейса.

АЛГОРИТМ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

Этапы реализации ПС (ПМ)

Для реализации приложения «Система администрирования для НРИ D&D» можно выделить следующие этапы:

1. Анализ требований и проектирование:
   1. Изучение бизнес-требований и пользовательских запросов.
   2. Проектирование базы данных для хранения информации о персонажах.
   3. Разработка структуры пользовательского интерфейса в XAML.
   4. Проектирование архитектуры приложения.
2. Настройка окружения разработки:
   1. Установка Visual Studio и необходимых компонентов.
   2. Создание проекта WPF.
3. Разработка интерфейса:
   1. Создание главного окна приложения в XAML.
   2. Добавление элементов управления для выбора расы, класса и генерации характеристик.
   3. Визуализация списка доступных персонажей.
4. Логика бросков Характеристик и навыков:
   1. Реализация логирования и вывода результата броска.
   2. Обновление соответствующих текстовых полей в интерфейсе с новыми значениями.
5. Логика переключения персонажа в ленте в инициативы:
   1. Отображение перемещения персонажей в ленте.
   2. Реализация поражения персонажа, когда его здоровье понижается до 0.
6. Тестирование:
   1. Проведение модульных тестов для проверки корректности работы каждого функционала.
   2. Проведение интеграционных тестов для оценки взаимодействия компонентов.
7. Оптимизация и доработка:
   1. Выявление узких мест и оптимизация производительности.
   2. Доработка пользовательского интерфейса и функционала на основе обратной связи.
8. Документация:
   1. Подготовка технической документации.
   2. Создание руководства пользователя.

Каждый этап включает в себя ряд конкретных задач и шагов, необходимых для успешной реализации приложения.

Пользовательский интерфейс ПС (ПМ)

Взаимодействие пользователей с ПС (ПМ)

Алгоритмы взаимодействия:

1. Вход в систему:
   1. Администратор (Мастер игры) запускает приложение общим списком игровых персонажей.
2. Управление боем:
   1. Администратор запускает приложение для управления состоянием персонажей в бою путем «Броска инициативы, после сего они отправляются в ленту инициативы.
   2. Система предоставляет интерфейс для отслеживания текущих персонажей в бою с их характеристиками и статусами.
   3. При необходимости, администратор может вносить изменения в характеристики персонажей, их состояние (жив/мертв), инициативу и другие параметры.
   4. Система отображает актуальное состояние боя в режиме реального времени.

Проектирование пользовательских сценариев

При проектировании пользовательского интерфейса для администрации и управления состоянием персонажей в бою, следующие ключевые сценарии могут быть учтены:

1. Добавление нового персонажа в бой:
   1. Описание: Администратор запускает приложение и выбирает опцию "Добавить нового персонажа в бой". Система предоставляет форму для ввода основных данных персонажа.
   2. Взаимодействие: Администратор вводит необходимую информацию, такую как имя персонажа, раса, класс и характеристики, затем подтверждает добавление персонажа в бой.
2. Управление состоянием персонажей в бою:
   1. Описание: Администратор выбирает опцию "Управление боем", где отображается текущий список персонажей в бою с их характеристиками и статусами (жив/мертв).
   2. Взаимодействие: Администратор может вносить изменения в характеристики персонажей, их состояние, инициативу и другие параметры. Система отображает изменения в режиме реального времени.
3. Просмотр и завершение боя:
   1. Описание: После завершения боя, администратор имеет возможность просмотреть итоги и состояние персонажей. Система предоставляет возможность завершить бой и сохранить результаты.
   2. Взаимодействие: Администратор анализирует результаты боя, подтверждает завершение и, при необходимости, сохраняет данные о бое.

Проектирование данных сценариев поможет обеспечить легкость использования приложения для администрации и эффективное управление состоянием персонажей в бою.

Определение операций пользователей

При проектировании пользовательского интерфейса приложения "Система администрирования для НРИ D&D" определены следующие операции, которые пользователи могут выполнять в рамках прецедентов:

1. Управление боем:

Операции:

* 1. Выбор персонажа для управления в бою.
  2. Отображение текущего состояния персонажа в бою, включая характеристики, текущие очки здоровья и другие важные параметры.
  3. Изменение текущих характеристик персонажа, таких как очки здоровья, состояния и дополнительные параметры.
  4. Подтверждение и сохранение изменений в состоянии персонажа в бою.

1. Управление списком персонажей в бою:

Операции:

* 1. Просмотр списка персонажей, участвующих в текущем бою.
  2. Добавление новых персонажей в бой.
  3. Удаление персонажей из боя.
  4. Переупорядочение списка персонажей в зависимости от инициативы.

1. Лог боя:

Операции:

* 1. Отображение лога событий в текущем бою.
  2. Ведение записей о действиях персонажей, полученных уронах, успешных или неудачных спасбросках и других событиях в бою.

1. Управление раундами:

Операции:

* 1. Переход к следующему раунду и автоматическое обновление инициативы.

1. Просмотр подробной информации о персонаже:

Операции:

* 1. Выбор персонажа для просмотра подробной информации.
  2. Отображение основных характеристик, навыков и других данных о персонаже.
  3. Возможность внесения изменений в характеристики персонажа вне боя.

Все взаимодействия с интерфейсом осуществляются в режиме реального времени, обеспечивая пользователю максимальное удобство и эффективность при администрировании и управлении состоянием персонажа в бою.

Проектирование структуры экранов и схема функционирования экранов

На основе определенных операций пользователей была разработана структура экрана. Вся информация о персонаже, включая основные и дополнительные характеристики, отображается на единственном экране. Пользователь имеет возможность в интерактивном режиме вводить и изменять данные, а также выполнять различные действия.

На основе определенных операций пользователей и сценариев взаимодействия, была разработана структура экранов системы:

Ввод и редактирование данных:

Пользователь взаимодействует с текстовыми полями и элементами выбора для ввода основных и дополнительных характеристик.

Интерактивные элементы обеспечивают удобство ввода и выбора параметров.

На рисунке 5 схема функционирования экранов приложения «Система администрирования для НРИ D&D»

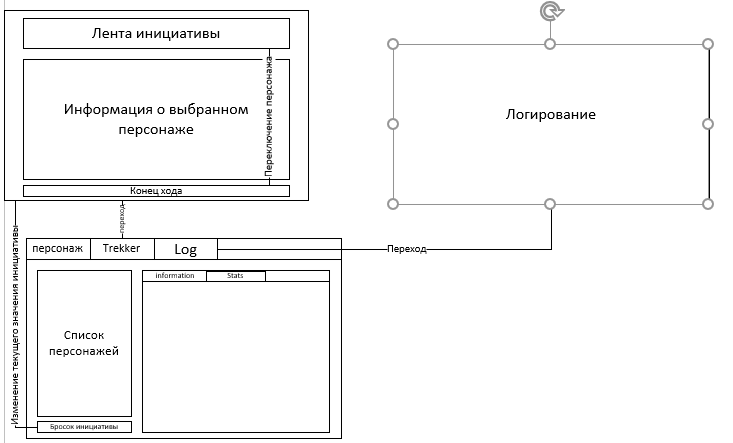


Рисунок 2 – Схема функционирования экранов

Такой подход создает единое пространство, где пользователь может управлять всей информацией о персонаже на одном экране, что способствует удобству использования приложения.

Входные, выходные и промежуточные данные

В данном разделе представлена структура данных для взаимодействия с состоянием персонажа в бою в приложении "Система администрирования для НРИ D&D". Этот раздел описывает входные, промежуточные и выходные данные, необходимые для эффективного управления персонажами в рамках боевых сценариев Dungeons & Dragons (D&D).

Входные данные:

Пользователь взаимодействует с приложением, указывая различные параметры, такие как действия персонажа, изменения состояния (например, получение урона). Ввод осуществляется через удобный интерфейс, предоставляющий элементы управления для интуитивного понимания пользовательского ввода.

Форма для ввода пользовательских действий, изменений состояния и выбора целей.

Интерфейс приложения с элементами управления для комфортного взаимодействия пользователя.

Промежуточные данные:

На этапе промежуточной обработки осуществляется анализ и обновление данных, связанных с состоянием персонажа в бою. Здесь происходит обработка пользовательских действий, расчет изменений в характеристиках персонажа и временное хранение результатов для обеспечения целостности боевого процесса.

Процесс анализа и обновления данных, связанных с состоянием персонажа в бою.

Временные данные, используемые для хранения результатов промежуточной обработки в рамках боевого сценария.

Выходные данные:

Обновленное состояние персонажа отображается на пользовательском интерфейсе в соответствии с проведенными действиями. Эти выходные данные предоставляют полную информацию о текущем состоянии персонажа в бою, включая изменения характеристик, статусы и другие важные детали.

Отображение актуального состояния персонажа на пользовательском интерфейсе.

Структура данных для сохранения и загрузки информации о состоянии персонажа, соответствующая определенным схемам данных.

В общем, данный раздел предоставляет обзор ключевых этапов взаимодействия приложения с данными в рамках управления состоянием персонажа в бою.

Архитектура ПС (ПМ)

Класс ApplicationViewModel представляет ViewModel для создания персонажа в приложении. Давайте рассмотрим каждое из свойств и методов класса:

private Random Random = new Random();

Тип: Random

Описание: Приватное поле, представляющее класс генерации случайных значений.

private Character selectedCharacter;

Тип: Character

Описание: Приватное поле, представляющее выбранного персонажа в списке.

public Character SelectedCharacter;

Тип: Character

Описание: Публичное свойство для доступа к выбранному персонажу. Имеет метод уведомления об изменении.

private Template selectedInBattle;

Тип: Template

Описание: Приватное поле, представляющее выбранного персонажа в битве.

public Template SelectedInBattle;

Тип: Template

Описание: Публичное свойство для доступа к выбранному персонажу в битве. Имеет метод уведомления об изменении.

public ObservableCollection<Character> Characters { get; set; }

Тип: ObservableCollection<Character>

Описание: Публичное свойство, представляющее коллекцию персонажей. Используется для отображения списка персонажей в приложении.

public ObservableCollection<Template> InBattle { get; set; }

Тип: ObservableCollection<Template>

Описание: Публичное свойство, представляющее коллекцию персонажей в битве. Используется для отображения списка персонажей, участвующих в битве.

public static ObservableCollection<string> Log { get; set; }

Тип: ObservableCollection<string>

Описание: Публичное статическое свойство, представляющее коллекцию строк для логирования событий в приложении.

private Dice D20 = new Dice() { Count = 1, Type = 20 };

Тип: Dice

Описание: Приватное поле, представляющее объект класса Dice для броска 20-гранных костей.

private RelayCommand initiativeCatsCommand;

Тип: RelayCommand

Описание: Приватное поле, представляющее команду для выполнения действий при броске инициативы в битве.

public RelayCommand InitiativeCatsCommand;

Тип: RelayCommand

Описание: Публичное свойство для доступа к команде броска инициативы. Имеет метод уведомления об изменении.

private RelayCommand endTurnCommand;

Тип: RelayCommand

Описание: Приватное поле, представляющее команду для выполнения действий при завершении хода в битве.

public RelayCommand EndTurnCommand;

Тип: RelayCommand

Описание: Публичное свойство для доступа к команде завершения хода. Имеет метод уведомления об изменении.

public ApplicationViewModel();

Тип: Конструктор

Описание: Публичный конструктор класса. Инициализирует объект ApplicationViewModel, заполняя коллекции персонажей и персонажей в битве, а также устанавливает начального выбранного персонажа.

Template – это класс, представляющий основную модель персонажа в приложении. Этот класс содержит различные характеристики и свойства, необходимые для отображения и управления персонажем в контексте игрового процесса. Данный класс также реализует интерфейс PropPropertyChanged, предназначенный для обработки уведомлений об изменениях свойств.private string name;

Тип: string

Описание: Приватное поле, представляющее имя объекта Template.

public string Name;

Тип: string

Описание: Публичное свойство для доступа к имени объекта Template. Имеет метод уведомления об изменении.

private string race;

Тип: string

Описание: Приватное поле, представляющее расу персонажа объекта Template.

public string Race;

Тип: string

Описание: Публичное свойство для доступа к расе персонажа объекта Template. Имеет метод уведомления об изменении.

private bool isFriend;

Тип: bool

Описание: Приватное поле, определяющее, является ли персонаж другом (true) или врагом (false).

public bool IsFriend;

Тип: bool

Описание: Публичное свойство для доступа к статусу друга/врага персонажа объекта Template. Имеет метод уведомления об изменении.

private int level;

Тип: int

Описание: Приватное поле, представляющее уровень персонажа объекта Template.

public int Level;

Тип: int

Описание: Публичное свойство для доступа к уровню персонажа объекта Template. Имеет метод уведомления об изменении.

private int hitPoints;

Тип: int

Описание: Приватное поле, представляющее общее количество хитов персонажа объекта Template.

public int HitPoints;

Тип: int

Описание: Публичное свойство для доступа к общему количеству хитов персонажа объекта Template. Имеет метод уведомления об изменении.

private int currentHitPoints;

Тип: int

Описание: Приватное поле, представляющее текущее количество хитов персонажа объекта Template.

public int CurrentHitPoints;

Тип: int

Описание: Публичное свойство для доступа к текущему количеству хитов персонажа объекта Template. Имеет метод уведомления об изменении.

private int deathSaveTrue = 0;

Тип: int

Описание: Приватное поле, представляющее количество успешных спасбросков от смерти.

public int DeathSaveTrue;

Тип: int

Описание: Публичное свойство для доступа к количеству успешных спасбросков от смерти персонажа объекта Template. Имеет метод уведомления об изменении.

private int deathSaveFalse = 0;

Тип: int

Описание: Приватное поле, представляющее количество неудачных спасбросков от смерти.

public int DeathSaveFalse;

Тип: int

Описание: Публичное свойство для доступа к количеству неудачных спасбросков от смерти персонажа объекта Template. Имеет метод уведомления об изменении.

private int armorClass;

Тип: int

Описание: Приватное поле, представляющее класс брони персонажа объекта Template.

public int ArmorClass;

Тип: int

Описание: Публичное свойство для доступа к классу брони персонажа объекта Template. Имеет метод уведомления об изменении.

private int masteryBonus;

Тип: int

Описание: Приватное поле, представляющее бонус мастерства персонажа объекта Template.

public int MasteryBonus;

Тип: int

Описание: Публичное свойство для доступа к бонусу мастерства персонажа объекта Template. Имеет метод уведомления об изменении.

private int speed;

Тип: int

Описание: Приватное поле, представляющее скорость передвижения персонажа объекта Template.

public int Speed;

Тип: int

Описание: Публичное свойство для доступа к скорости передвижения персонажа объекта Template. Имеет метод уведомления об изменении.

private int passivePerception;

Тип: int

Описание: Приватное поле, представляющее навык пассивного восприятия персонажа объекта Template.

public int PassivePerception;

Тип: int

Описание: Публичное свойство для доступа к навыку пассивное восприятие.

private int initiative;

Тип: int

Описание: Приватное поле, представляющее инициативу персонажа.

private int initiativeCast;

Тип: int

Описание: Приватное поле, представляющее модифицированную инициативу с учетом броска кубика.

private Strength strength;

Тип: Strength

Описание: Приватное поле, представляющее характеристику "Сила" персонажа.

private int strengthValue;

Тип: int

Описание: Приватное поле, представляющее значение характеристики "Сила".

public int StrengthValue;

Тип: int

Описание: Публичное свойство для доступа к значению характеристики "Сила".

public Strength Strength;

Тип: Strength

Описание: Публичное свойство для доступа к характеристике "Сила".

private Dexterity dexterity;

Тип: Dexterity

Описание: Приватное поле, представляющее характеристику "Ловкость" персонажа.

private int dexterityValue;

Тип: int

Описание: Приватное поле, представляющее значение характеристики "Ловкость".

public int DexterityValue;

Тип: int

Описание: Публичное свойство для доступа к значению характеристики "Ловкость".

public Dexterity Dexterity;

Тип: Dexterity

Описание: Публичное свойство для доступа к характеристике "Ловкость".

... (аналогично для характеристик: Constitution, Intelligence, Wisdom, Charisma)

private Dice hitPointDice;

Тип: Dice

Описание: Приватное поле, представляющее кубик для определения количества очков здоровья.

public Dice HitPointDice;

Тип: Dice

Описание: Публичное свойство для доступа к кубику для определения очков здоровья.

public Template(

int strengthValue = 10,

int dexterityValue = 10,

int constitutionValue = 10,

int intelligenceValue = 10,

int wisdomValue = 10,

int charismaValue = 10);

Тип: Конструктор

Описание: Публичный конструктор, инициализирующий объект Template с заданными значениями характеристик по умолчанию.

public void UpdateMasteryBonus();

Тип: void

Описание: Публичный метод для обновления бонуса мастерства в зависимости от уровня персонажа.

Класс Skill предоставляет свойства для отображения навыков персонажа и содержит логику их использования. Давайте рассмотрим каждое поле, свойство и метод:

name

Тип: string

Описание: Приватное поле, представляющее имя навыка.

proficient

Тип: bool

Описание: Приватное поле, указывающее, обладает ли персонаж профициентностью в данном навыке.

modifier

Тип: int

Описание: Приватное поле, представляющее модификатор навыка.

value

Тип: int

Описание: Приватное поле, представляющее базовое значение навыка.

stat

Тип: Stat

Описание: Приватное поле, представляющее связанную статистику, влияющую на навык.

dice

Тип: Dice

Описание: Приватное поле, представляющее объект для броска костей в рамках навыка.

SkillCastCommand (свойство)

Тип: RelayCommand

Описание: Публичное свойство, представляющее команду для использования навыка. Создает всплывающее окно с результатом броска и добавляет информацию в лог.

SkillBackground (свойство)

Тип: SolidColorBrush

Описание: Публичное свойство для доступа к цвету фона навыка.

Name (свойство)

Тип: string

Описание: Публичное свойство для доступа к имени навыка.

Proficient (свойство)

Тип: bool

Описание: Публичное свойство для доступа к информации о профициентности в навыке.

Modifier (свойство)

Тип: int

Описание: Публичное свойство для доступа к модификатору навыка, учитывая профициентность.

Value (свойство)

Тип: int

Описание: Публичное свойство для доступа к базовому значению навыка.

Stat (свойство)

Тип: Stat

Описание: Публичное свойство для доступа к связанной статистике навыка.

Dice (свойство)

Тип: Dice

Описание: Публичное свойство для доступа к объекту для броска костей в рамках навыка.

MasteryBonus (свойство)

Тип: int

Описание: Публичное свойство для доступа к бонусу мастерства, зависящему от связанной статистики.

Skill (конструктор)

Параметры: string name, Stat stat

Описание: Конструктор, устанавливающий имя навыка и связанную статистику, и инициализирующий объект для броска костей.

Cast (метод)

Тип: string

Описание: Метод для проведения броска костей и формирования строки с результатом броска и модификатором.

Этот класс предоставляет функциональность для работы с навыками персонажа, включая их использование и отображение результатов. Класс Dice предоставляет свойства и метод для работы с костями (броска кубика) в рамках игровой системы. Давайте рассмотрим каждое поле, свойство и метод:

count

Тип: int

Описание: Приватное поле, представляющее количество костей (бросков) в рамках одного действия.

type

Тип: int

Описание: Приватное поле, представляющее количество граней на каждой кости.

Count (свойство)

Тип: int

Описание: Публичное свойство для доступа к значению количества костей. Имеет метод уведомления об изменении.

Type (свойство)

Тип: int

Описание: Публичное свойство для доступа к значению количества граней на кости. Имеет метод уведомления об изменении.

Cast (метод)

Тип: int

Описание: Публичный метод для проведения броска костей. Генерирует случайные значения в соответствии с заданным количеством костей и граней. Возвращает сумму результатов бросков.

Этот класс предоставляет функциональность для генерации случайных чисел, эмулируя броски костей, и используется в рамках игровой системы для различных игровых механик.

Character.cs – класс, представляющий персонажа с характеристиками и связанными с ними методами. Это класс, представляющий персонажа в контексте игры Dungeons & Dragons. Класс содержит характеристики персонажа, а также связанные методы для работы с этими характеристиками. Давайте подробно рассмотрим свойства и методы класса:

private int exhaustion; Тип: int Описание: приватное поле для хранения уровня истощения персонажа.

public int Exhaustion Тип: int Описание: публичное свойство для доступа к уровню истощения. Имеет логику уведомления об изменении.

private int level; Тип: int Описание: приватное поле для хранения уровня персонажа.

public new int Level Тип: int Описание: публичное свойство для доступа к уровню персонажа. При изменении происходит обновление хитпоинтов и мастерства. Имеет логику уведомления об изменении.

private int hitPoints; Тип: int Описание: приватное поле для хранения значения хитпоинтов персонажа.

public new int HitPoints Тип: int Описание: публичное свойство для доступа к хитпоинтам персонажа. Имеет логику уведомления об изменении.

private int currentHitPointDice; Тип: int Описание: приватное поле для хранения количества костей хитпоинтов на текущем уровне.

public int CurrentHitPointDice Тип: int Описание: публичное свойство для доступа к текущему количеству костей хитпоинтов. Имеет логику уведомления об изменении.

private string classCharacter; Тип: string Описание: приватное поле для хранения класса персонажа.

public string ClassCharacter  
Тип: string Описание: публичное свойство для доступа к классу персонажа. При изменении происходит обновление мастерства. Имеет логику уведомления об изменении.

private ObservableCollection<string> abilities; Тип: ObservableCollection<string> Описание: приватная коллекция для хранения способностей персонажа.

public ObservableCollection<string> Abilities Тип: ObservableCollection<string> Описание: публичное свойство для доступа к способностям персонажа. Имеет логику уведомления об изменении.

private Race race; Тип: Race Описание: приватное поле для хранения расы персонажа.

public new Race Race Тип: Race Описание: публичное свойство для доступа к расе персонажа. Имеет логику уведомления об изменении.

public Character() Тип: конструктор Описание: конструктор по умолчанию. Устанавливает истощение в 0.

Давайте представим все описанные выше нами классы в виде диаграммы классов для наглядности (рис.6).

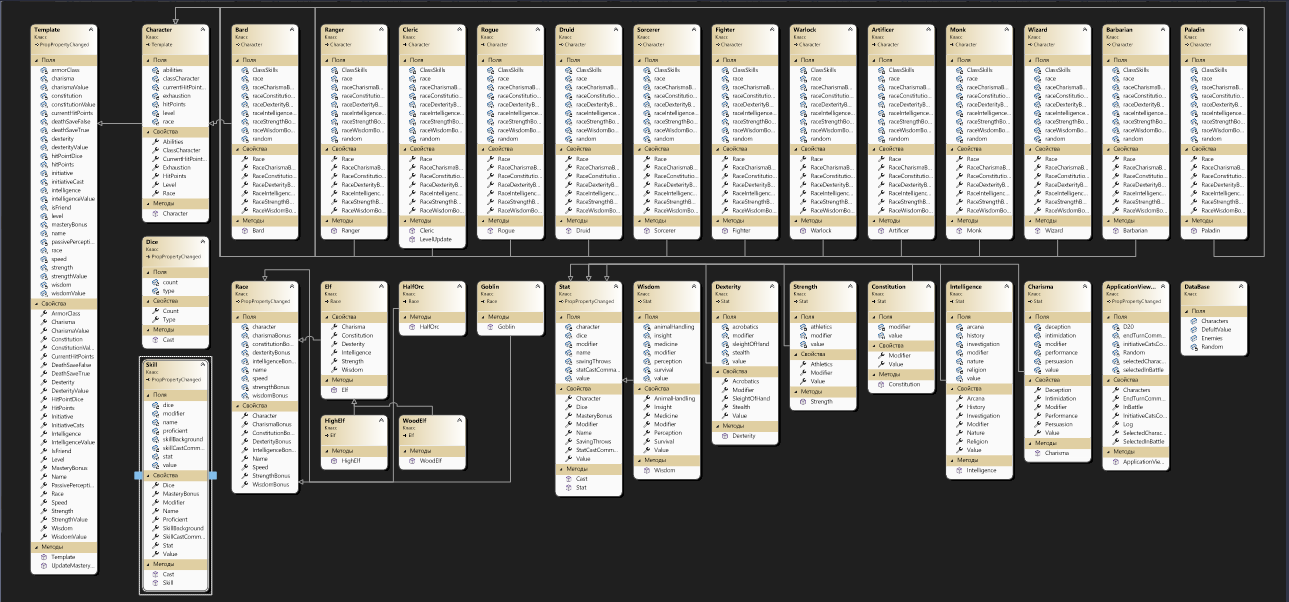


Рисунок 3 – Диаграмма классов

ТЕСТИРОВАНИЕ И ОПТИМИЗАЦИЯ

План тестирования

Тестирование является важной частью жизненного цикла разработки программного обеспечения, направленной на обеспечение качества и надежности создаваемой системы.

В контексте приложения для администрации и управления состоянием персонажа в бою тестирование остается важным этапом разработки, направленным на обеспечение надежности и функциональности системы. Ручное тестирование выбрано из-за его гибкости, способности адаптироваться к изменениям в коде и учета человеческого фактора в оценке удобства использования. Это позволяет оперативно реагировать на изменения и поддерживать достаточный уровень проверки качества при ограниченных ресурсах проекта.

План тестирования приложения для управления состоянием персонажа в бою:

1. Проверка функциональности управления персонажами:
   1. Проверка корректности выбора персонажа для управления.
   2. Проверка возможности изменения параметров персонажа (инициатива, текущие очки здоровья) через интерфейс.
   3. Проверка обновления списка персонажей в бою после внесения изменений.
2. Интеграционное тестирование:
   1. Проверка взаимодействия между компонентами приложения (выбор персонажа, управление состоянием, обновление интерфейса).
   2. Выполнение различных действий (выбор, обновление параметров) и проверка корректности взаимодействия компонентов.
3. Тестирование производительности:
   1. Оценка времени отклика приложения при обновлении состояния персонажа.
   2. Проверка производительности при работе с большим количеством персонажей в бою.
4. Тестирование совместимости:
   1. Проверка работы приложения на различных операционных системах (Windows).
   2. Установка и тестирование на различных версиях операционной системы.
5. Тестирование удобства использования:
   1. Оценка интуитивности интерфейса управления состоянием персонажа в бою.
   2. Проведение тестов с участием различных администраторов для оценки удобства использования при реальном взаимодействии.

Результаты тестирования

В ходе тестирования приложения «Система администрирования для НРИ D&D» было проведено ручное тестирование.

В результате тестов обнаружена ошибка при вводе символов, отличных от чисел, в поля характеристик.

Интерфейс обновляется в соответствии с выбранными параметрами, и приложение отвечает на действия пользователя в течение заданных временных рамок.

Тестирование также включило проверку работы на операционной системе Windows, обеспечивая совместимость с различными версиями.

Оптимизация ПС (ПМ)

Процесс оптимизации приложения «Система администрирования для НРИ D&D» включал тщательный анализ результатов тестирования с целью выявления узких мест и определения возможных улучшений. Оптимизация была направлена на улучшение производительности, устранение возможных ошибок и повышение эффективности работы приложения.

Этот процесс включал идентификацию время затратных операций, оптимизацию алгоритмов, улучшение структуры приложения, оптимизацию интерфейса и проверку вводимых значений.

Результатом стало существенное улучшение общей производительности и отзывчивости приложения, что подтверждено успешным прохождением последующих тестов.

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Приложение «Система администрирования для НРИ D&D» предоставляет интуитивно понятный интерфейс, где пользователь может задавать инициативу персонажу, редактировать характеристики, тем самым кастомизируя персонажей и повышать свою индивидуальность в игровом мире.

Чтобы войти в приложение, пользователь должен зайти в загрузки или на рабочий стол и найти ярлык с приложением, нажав на него правой кнопкой мыши несколько раз.

Когда приложение запуститься, то пользователь окажется в основном экране приложения с основным его функционалом (рис.4).

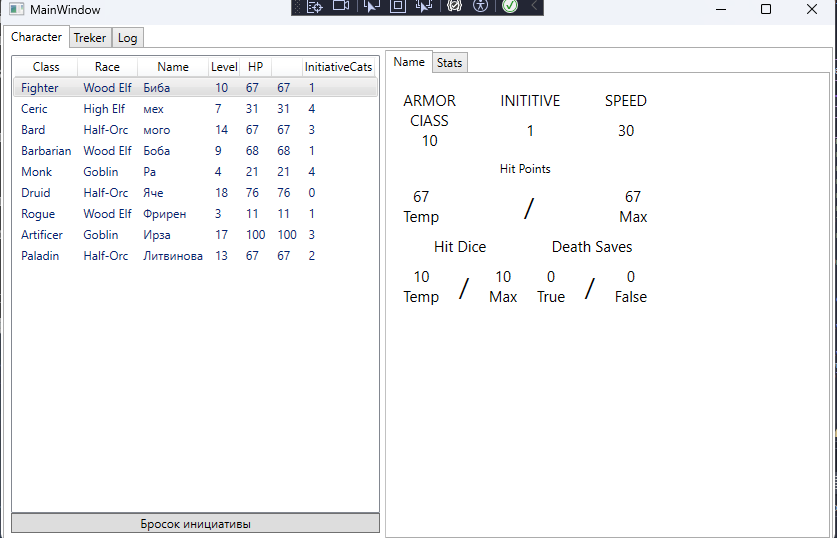


Рисунок 4 – Общий вид главного экрана приложения

Пользователь может выбрать персонажа и поменять набор выбранных навыков, для этого надо перейти в вкладку «Stats» (рис.5).

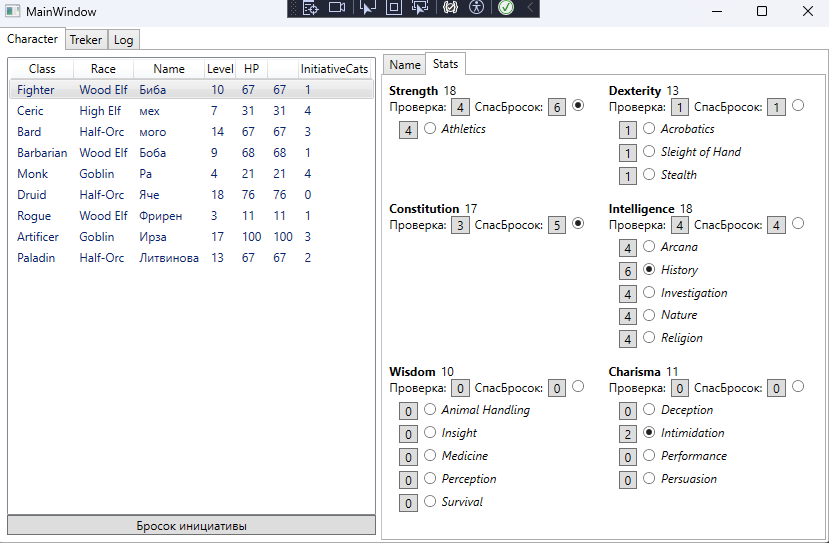


Рисунок 5 – Общий вид главного экрана приложения

Пользователь может выбрать персонажа и пробросить значения его инициативы, которое добавит данного персонажа в список персонажей, которые находятся в бою. Для этого необходимо нажать кнопку «Бросок Инициативы». Значение всех характеристик будут отображаться в отдельном поле (рис.6-7).

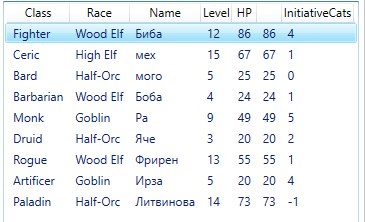




Рисунок 6-7 – Кнопка «Выбор персонажа и Бросок Инициативы»

И тогда в ленте инициативы отобразятся выбранный персонаж (рис.8).

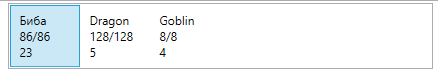


Рисунок 8 – выбранный персонаж в ленте инициативы

Для того чтобы нанести урон участнику ленты инициативы, надо бросить кубик D20 в реальности и просмотреть класс защиты в отображаемой информации выбранного персонажа.

Например, персонаж Биба хочет атаковать Гоблина игрок бросает свой D20 и смотрит класс защиты Гоблина.

Бибе выпало 11, а класс защиты Гоблина 15, соответственно Биба не попадает по Гоблину, если бы Бибе выпало больше или равно 15, то мастер бы ему назначил кубик урона, который бы он пробросил и нанес бы урон. (рис. 9-10)



Рисунок 9-10 Бросок игрока и класс защиты Гоблина

После того как ход закончен, то пользователю надо нажать на кнопку «Закончить Ход».

  
Рисунок 11 – кнопка «Закончить Ход»

После этого система переместит указатель на следующего в ленте инициативе. (рис. 12)

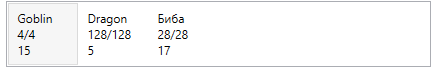


Рисунок 12 – перетасованная лента инициативы

Когда очки здоровья персонажа опускаются до нуля, то он каждый раз когда будет подходить его очередь он автоматически будет пробрасывать бросок против смерти, если он будет больше 10, то в его поле успешных бросков станет больше на 1, а значение, которое надо будет пробросить увеличится на 1, если в поле успешных бросков станет 3, то персонаж встанет с одним очком здоровья . Если персонаж не прошёл проверку, то в его поле неуспешных бросков против смерти увеличится на 1, если в поле неуспешных бросков станет 3, то персонаж исключится из ленты инициативы.  
Приложение «Система администрирования для НРИ D&D» предоставляет удобный способ администрирования и управления персонажами в бою на партии в реальности для игры в Dungeons & Dragons, а его интуитивно понятный интерфейс обеспечивает легкость в использовании.

ЗАКЛЮЧЕНИЕ

Успешно завершено разработка десктопного приложения "Управление состоянием персонажа в бою" для операционной системы Windows. Приложение полностью соответствует поставленным задачам и предоставляет интуитивно понятный интерфейс для эффективного управления состоянием персонажей в боевых сценах.

Процесс разработки включал использование Visual Studio с языками программирования C# и XAML, а также фреймворка WPF, обеспечивая высокую производительность и интуитивное взаимодействие с пользователем. Приложение ориентировано на отслеживание и управление состоянием персонажей в реальном времени.

Разработка включала в себя проектирование и реализацию функционала отслеживания состояния персонажей в бою. Ручное тестирование подтвердило стабильную и корректную работу приложения, а оптимизация обеспечила высокую производительность.

Созданное приложение предоставляет удобный способ администрирования и отслеживания состояния персонажей в боевых сценах, что сокращает время подготовки к боям и повышает общий опыт пользователей. Современный дизайн и простой пользовательский интерфейс создают приятное впечатление и делают приложение привлекательным для администраторов и участников игры.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Wizards of the Coast. URL: https://dnd.wizards.com
2. Игромания. "Что такое Dungeons & Dragons и как в это играть. Краткий справочник для новичков." URL: https://www.igromania.ru/article/31235/Chto\_takoe\_Dungeons\_and\_Dragons\_i\_kak\_v\_yeto\_igrat.\_Kratkiy\_spravochnik\_dlya\_novichkov.html
3. D&D.SU. "Дварф" URL: https://dnd.su/race/78-dwarf/
4. DM Stuff. Каталог рас. URL: http://dm-stuff.ru/race-catalog/
5. Microsoft Docs. "Model-View-ViewModel (MVVM) architecture for .NET MAUI" URL: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm
6. Metanit. "WPF Tutorial" URL: https://metanit.com/sharp/wpf/22.1.php
7. TV Besedka. "Шаблон MVVM - просто о сложном. Пример на C#" URL: https://tvbesedka.com/article/pattern\_mvvm\_-\_prosto\_o\_slozhnom\_primer\_na\_c\_644
8. WebDelphi. "Первое приложение WPF" URL: <https://csharp.webdelphi.ru/pervoe-prilozhenie-wpf/>
9. Chat GPT
10. TTG Club https://ttg.club

ПРИЛОЖЕНИЯ

## Приложение А: MainWindow.xaml.cs

using DndHelper.ViewModel;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace DndHelper

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

DataContext = new ApplicationViewModel();

}

}

}

## Приложение Б: MainWindow.xaml

<Window x:Class="DndHelper.MainWindow"

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

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

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

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

xmlns:local="clr-namespace:DndHelper"

mc:Ignorable="d"

Title="MainWindow"

Height="auto"

Width="850"

MinHeight="400"

MinWidth="850">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TabControl Grid.RowSpan="1" Margin="2,0,2,0">

<TabItem Header="Character">

<DockPanel>

<DockPanel DockPanel.Dock="Left"

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

<StackPanel DockPanel.Dock="Bottom">

<Button Content="Бросок инициативы"

Command="{Binding InitiativeCatsCommand}" />

</StackPanel>

<ListView Name="UserList"

ItemsSource="{Binding Characters}"

SelectedItem="{Binding SelectedCharacter}"

ScrollViewer.CanContentScroll="True"

ScrollViewer.VerticalScrollBarVisibility="Hidden">

<ListView.View>

<GridView>

<GridViewColumn Header="Class" DisplayMemberBinding="{Binding Path=ClassCharacter}"/>

<GridViewColumn Header="Race" DisplayMemberBinding="{Binding Path=Race.Name}"/>

<GridViewColumn Header="Name" DisplayMemberBinding="{Binding Path=Name}"/>

<GridViewColumn Header="Level" DisplayMemberBinding="{Binding Path=Level}"/>

<GridViewColumn Header="HP" DisplayMemberBinding="{Binding Path=CurrentHitPoints}"/>

<GridViewColumn Header="" DisplayMemberBinding="{Binding Path=HitPoints}"/>

<GridViewColumn Header="InitiativeCats" DisplayMemberBinding="{Binding Path=InitiativeCats}" />

</GridView>

</ListView.View>

</ListView>

</DockPanel>

<TabControl>

<TabItem Header="Name">

<DockPanel>

<StackPanel DataContext="{Binding SelectedCharacter}"

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

<DockPanel>

<DockPanel DockPanel.Dock="Left"

Margin="10,10,10,10"

VerticalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

Text="ARMOR"

HorizontalAlignment="Center"

FontSize="15"/>

<TextBlock DockPanel.Dock="Bottom"

Text="{Binding ArmorClass}"

HorizontalAlignment="Center"

FontSize="15"/>

<TextBlock Text="ClASS"

HorizontalAlignment="Center"

FontSize="15"/>

</DockPanel>

<DockPanel DockPanel.Dock="Right"

Margin="10,10,10,10"

HorizontalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

Text="SPEED"

HorizontalAlignment="Center"

FontSize="15"/>

<TextBlock Text="{Binding Speed}"

HorizontalAlignment="Center"

VerticalAlignment="Center"

FontSize="15"/>

</DockPanel>

<DockPanel Margin="10,10,10,10">

<TextBlock DockPanel.Dock="Top"

Text="INITITIVE"

HorizontalAlignment="Center"

FontSize="15"/>

<TextBlock Text="{Binding Initiative}"

HorizontalAlignment="Center"

Padding="25,10,25,10"

FontSize="15"/>

</DockPanel>

</DockPanel>

<DockPanel>

<TextBlock DockPanel.Dock="Top"

Text="Hit Points"

HorizontalAlignment="Center"/>

<DockPanel DockPanel.Dock="Left"

Margin="10,10,10,10"

HorizontalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

Text="{Binding CurrentHitPoints}"

HorizontalAlignment="Center"

FontSize="15"/>

<TextBlock Text="Temp"

FontSize="15"/>

</DockPanel>

<DockPanel DockPanel.Dock="Right"

Margin="10,10,10,10"

HorizontalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

Text="{Binding HitPoints}"

HorizontalAlignment="Center"

FontSize="15"/>

<TextBlock Text="Max"

FontSize="15"/>

</DockPanel>

<DockPanel Margin="10,10,10,10"

HorizontalAlignment="Center"

VerticalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

HorizontalAlignment="Center"

FontSize="25"

Text="/"/>

</DockPanel>

</DockPanel>

<DockPanel>

<DockPanel DockPanel.Dock="Left">

<TextBlock DockPanel.Dock="Top"

HorizontalAlignment="Center"

Text="Hit Dice"

FontSize="15"/>

<DockPanel DockPanel.Dock="Left"

Margin="10,10,10,10"

HorizontalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

Text="{Binding CurrentHitPointDice}"

HorizontalAlignment="Center"

FontSize="15"/>

<TextBlock Text="Temp"

FontSize="15"/>

</DockPanel>

<DockPanel DockPanel.Dock="Right"

Margin="10,10,10,10"

HorizontalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

Text="{Binding HitPointDice.Count }"

HorizontalAlignment="Center"

FontSize="15"/>

<TextBlock Text="Max"

FontSize="15"/>

</DockPanel>

<DockPanel Margin="10,10,10,10"

HorizontalAlignment="Center"

VerticalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

HorizontalAlignment="Center"

FontSize="25"

Text="/"/>

</DockPanel>

</DockPanel>

<DockPanel DockPanel.Dock="Right">

<TextBlock DockPanel.Dock="Top"

HorizontalAlignment="Center"

Text="Death Saves"

FontSize="15"/>

<DockPanel DockPanel.Dock="Left"

Margin="10,10,10,10"

HorizontalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

Text="{Binding DeathSaveTrue}"

HorizontalAlignment="Center"

FontSize="15"/>

<TextBlock Text="True"

FontSize="15"/>

</DockPanel>

<DockPanel DockPanel.Dock="Right"

Margin="10,10,10,10"

HorizontalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

Text="{Binding DeathSaveFalse}"

HorizontalAlignment="Center"

FontSize="15"/>

<TextBlock Text="False"

FontSize="15"/>

</DockPanel>

<DockPanel Margin="10,10,10,10"

HorizontalAlignment="Center"

VerticalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

HorizontalAlignment="Center"

FontSize="25"

Text="/"/>

</DockPanel>

</DockPanel>

<DockPanel></DockPanel>

</DockPanel>

</StackPanel>

<DockPanel>

</DockPanel>

</DockPanel>

</TabItem>

<TabItem Header="Stats">

<Grid DataContext="{Binding SelectedCharacter}">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<!--Сила-->

<StackPanel Grid.Column="0"

Grid.Row="0"

Margin="5,5,0,10">

<!--Имя характеристики и ее значение-->

<WrapPanel>

<TextBlock Text="{Binding Strength.Name}"

Margin="0,0,5,0"

FontWeight="Bold"/>

<TextBlock Text="{Binding Strength.Value}"

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

</WrapPanel>

<!--Проверка и Спасбросок-->

<WrapPanel>

<WrapPanel>

<TextBlock Text="Проверка:"

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

<Button Content ="{Binding Strength.Modifier}"

Command="{Binding Strength.StatCastCommand}"

Padding="5,0,5,0"

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

</WrapPanel>

<WrapPanel>

<TextBlock Text="СпасБросок:" Margin="0,0,5,0"/>

<WrapPanel>

<Button Content="{Binding Strength.SavingThrows.Modifier}"

Command="{Binding Strength.SavingThrows.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Strength.SavingThrows.Proficient, Mode=TwoWay}"

FontStyle="Italic"

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

</WrapPanel>

</WrapPanel>

</WrapPanel>

<!--Навыки-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Strength.Athletics.Modifier}"

Command="{Binding Strength.Athletics.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Strength.Athletics.Proficient, Mode=TwoWay}"

Content="{Binding Strength.Athletics.Name}"

FontStyle="Italic"

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

</WrapPanel>

</StackPanel>

<!--Ловкость-->

<StackPanel Grid.Column="1"

Grid.Row="0"

Margin="5,5,0,10">

<!--Имя характеристики и ее значение-->

<WrapPanel>

<TextBlock Text="{Binding Dexterity.Name}"

Margin="0,0,5,0"

FontWeight="Bold"/>

<TextBlock Text="{Binding Dexterity.Value}"

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

</WrapPanel>

<!--Проверка и Спасбросок-->

<WrapPanel>

<WrapPanel>

<TextBlock Text="Проверка:"

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

<Button Content ="{Binding Dexterity.Modifier}"

Command="{Binding Dexterity.StatCastCommand}"

Padding="5,0,5,0"

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

</WrapPanel>

<WrapPanel>

<TextBlock Text="СпасБросок:" Margin="0,0,5,0"/>

<WrapPanel>

<Button Content="{Binding Dexterity.SavingThrows.Modifier}"

Command="{Binding Dexterity.SavingThrows.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Dexterity.SavingThrows.Proficient, Mode=TwoWay}"

FontStyle="Italic"

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

</WrapPanel>

</WrapPanel>

</WrapPanel>

<!--Навыки-->

<!--Акробатика-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Dexterity.Acrobatics.Modifier}"

Command="{Binding Dexterity.Acrobatics.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Dexterity.Acrobatics.Proficient, Mode=TwoWay}"

Content="{Binding Dexterity.Acrobatics.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Ловкость рук-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Dexterity.SleightOfHand.Modifier}"

Command="{Binding Dexterity.SleightOfHand.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Dexterity.SleightOfHand.Proficient, Mode=TwoWay}"

Content="{Binding Dexterity.SleightOfHand.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Скрытность-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Dexterity.Stealth.Modifier}"

Command="{Binding Dexterity.Stealth.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Dexterity.Stealth.Proficient, Mode=TwoWay}"

Content="{Binding Dexterity.Stealth.Name}"

FontStyle="Italic"

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

</WrapPanel>

</StackPanel>

<!--Телосложение-->

<StackPanel Grid.Column="0"

Grid.Row="1"

Margin="5,5,0,10">

<!--Имя характеристики и ее значение-->

<WrapPanel>

<TextBlock Text="{Binding Constitution.Name}"

FontWeight="Bold"

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

<TextBlock Text="{Binding Constitution.Value}"

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

</WrapPanel>

<!--Проверка и Спасбросок-->

<WrapPanel>

<WrapPanel>

<TextBlock Text="Проверка:"

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

<Button Content ="{Binding Constitution.Modifier}"

Command="{Binding Constitution.StatCastCommand}"

Padding="5,0,5,0"

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

</WrapPanel>

<WrapPanel>

<TextBlock Text="СпасБросок:"

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

<WrapPanel>

<Button Content="{Binding Constitution.SavingThrows.Modifier}"

Command="{Binding Constitution.SavingThrows.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Constitution.SavingThrows.Proficient, Mode=TwoWay}"

FontStyle="Italic"

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

</WrapPanel>

</WrapPanel>

</WrapPanel>

</StackPanel>

<!--Интелект-->

<StackPanel Grid.Column="1"

Grid.Row="1"

Margin="5,5,0,10">

<!--Имя характеристики и ее значение-->

<WrapPanel>

<TextBlock Text="{Binding Intelligence.Name}"

Margin="0,0,5,0"

FontWeight="Bold"/>

<TextBlock Text="{Binding Intelligence.Value}"

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

</WrapPanel>

<!--Проверка и Спасбросок-->

<WrapPanel>

<WrapPanel>

<TextBlock Text="Проверка:"

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

<Button Content ="{Binding Intelligence.Modifier}"

Command="{Binding Intelligence.StatCastCommand}"

Padding="5,0,5,0"

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

</WrapPanel>

<WrapPanel>

<TextBlock Text="СпасБросок:" Margin="0,0,5,0"/>

<WrapPanel>

<Button Content="{Binding Intelligence.SavingThrows.Modifier}"

Command="{Binding Intelligence.SavingThrows.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Intelligence.SavingThrows.Proficient, Mode=TwoWay}"

FontStyle="Italic"

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

</WrapPanel>

</WrapPanel>

</WrapPanel>

<!--Навыки-->

<!--Магия-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Intelligence.Arcana.Modifier}"

Command="{Binding Intelligence.Arcana.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Intelligence.Arcana.Proficient, Mode=TwoWay}"

Content="{Binding Intelligence.Arcana.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--История-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Intelligence.History.Modifier}"

Command="{Binding Intelligence.History.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Intelligence.History.Proficient, Mode=TwoWay}"

Content="{Binding Intelligence.History.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Анализ-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Intelligence.Investigation.Modifier}"

Command="{Binding Intelligence.Investigation.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Intelligence.Investigation.Proficient, Mode=TwoWay}"

Content="{Binding Intelligence.Investigation.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Природа-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Intelligence.Nature.Modifier}"

Command="{Binding Intelligence.Nature.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Intelligence.Nature.Proficient, Mode=TwoWay}"

Content="{Binding Intelligence.Nature.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Религия-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Intelligence.Religion.Modifier}"

Command="{Binding Intelligence.Religion.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Intelligence.Religion.Proficient, Mode=TwoWay}"

Content="{Binding Intelligence.Religion.Name}"

FontStyle="Italic"

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

</WrapPanel>

</StackPanel>

<!--Мудрость-->

<StackPanel Grid.Column="0"

Grid.Row="2"

Margin="5,5,0,10">

<!--Имя характеристики и ее значение-->

<WrapPanel>

<TextBlock Text="{Binding Wisdom.Name}"

Margin="0,0,5,0"

FontWeight="Bold"/>

<TextBlock Text="{Binding Wisdom.Value}"

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

</WrapPanel>

<!--Проверка и Спасбросок-->

<WrapPanel>

<WrapPanel>

<TextBlock Text="Проверка:"

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

<Button Content ="{Binding Wisdom.Modifier}"

Command="{Binding Wisdom.StatCastCommand}"

Padding="5,0,5,0"

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

</WrapPanel>

<WrapPanel>

<TextBlock Text="СпасБросок:" Margin="0,0,5,0"/>

<WrapPanel>

<Button Content="{Binding Wisdom.SavingThrows.Modifier}"

Command="{Binding Wisdom.SavingThrows.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Wisdom.SavingThrows.Proficient, Mode=TwoWay}"

FontStyle="Italic"

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

</WrapPanel>

</WrapPanel>

</WrapPanel>

<!--Навыки-->

<!--Уход за животными-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Wisdom.AnimalHandling.Modifier}"

Command="{Binding Wisdom.AnimalHandling.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Wisdom.AnimalHandling.Proficient, Mode=TwoWay}"

Content="{Binding Wisdom.AnimalHandling.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Проницательность-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Wisdom.Insight.Modifier}"

Command="{Binding Wisdom.Insight.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Wisdom.Insight.Proficient, Mode=TwoWay}"

Content="{Binding Wisdom.Insight.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Медицина-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Wisdom.Medicine.Modifier}"

Command="{Binding Wisdom.Medicine.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Wisdom.Medicine.Proficient, Mode=TwoWay}"

Content="{Binding Wisdom.Medicine.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Восприятие-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Wisdom.Perception.Modifier}"

Command="{Binding Wisdom.Perception.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Wisdom.Perception.Proficient, Mode=TwoWay}"

Content="{Binding Wisdom.Perception.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Выживание-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Wisdom.Survival.Modifier}"

Command="{Binding Wisdom.Survival.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Wisdom.Survival.Proficient, Mode=TwoWay}"

Content="{Binding Wisdom.Survival.Name}"

FontStyle="Italic"

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

</WrapPanel>

</StackPanel>

<!--Харизма-->

<StackPanel Grid.Column="1"

Grid.Row="2"

Margin="5,5,0,10">

<!--Имя характеристики и ее значение-->

<WrapPanel>

<TextBlock Text="{Binding Charisma.Name}"

Margin="0,0,5,0"

FontWeight="Bold"/>

<TextBlock Text="{Binding Charisma.Value}"

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

</WrapPanel>

<!--Проверка и Спасбросок-->

<WrapPanel>

<WrapPanel>

<TextBlock Text="Проверка:"

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

<Button Content ="{Binding Charisma.Modifier}"

Command="{Binding Charisma.StatCastCommand}"

Padding="5,0,5,0"

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

</WrapPanel>

<WrapPanel>

<TextBlock Text="СпасБросок:" Margin="0,0,5,0"/>

<WrapPanel>

<Button Content="{Binding Charisma.SavingThrows.Modifier}"

Command="{Binding Charisma.SavingThrows.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Charisma.SavingThrows.Proficient, Mode=TwoWay}"

FontStyle="Italic"

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

</WrapPanel>

</WrapPanel>

</WrapPanel>

<!--Навыки-->

<!--Обман-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Charisma.Deception.Modifier}"

Command="{Binding Charisma.Deception.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Charisma.Deception.Proficient, Mode=TwoWay}"

Content="{Binding Charisma.Deception.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Устрашение-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Charisma.Intimidation.Modifier}"

Command="{Binding Charisma.Intimidation.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Charisma.Intimidation.Proficient, Mode=TwoWay}"

Content="{Binding Charisma.Intimidation.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Представление-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Charisma.Performance.Modifier}"

Command="{Binding Charisma.Performance.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Charisma.Performance.Proficient, Mode=TwoWay}"

Content="{Binding Charisma.Performance.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Убеждение-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Charisma.Persuasion.Modifier}"

Command="{Binding Charisma.Persuasion.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Charisma.Persuasion.Proficient, Mode=TwoWay}"

Content="{Binding Charisma.Persuasion.Name}"

FontStyle="Italic"

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

</WrapPanel>

</StackPanel>

</Grid>

</TabItem>

</TabControl>

</DockPanel>

</TabItem>

<TabItem Header="Treker">

<DockPanel>

<!-- Upper Initiative Bar -->

<WrapPanel DockPanel.Dock="Top"

Orientation="Horizontal"

HorizontalAlignment="Center"

VerticalAlignment="Center">

<ListBox Name="InitiativeList"

ItemsSource="{Binding InBattle}"

SelectedItem="{Binding SelectedInBattle}"

Width="425"

Height="auto">

<!-- Define ListBox ItemTemplate for each character -->

<ListBox.ItemsPanel>

<ItemsPanelTemplate>

<StackPanel Orientation="Horizontal" />

</ItemsPanelTemplate>

</ListBox.ItemsPanel>

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Margin="5,5,5,5"

Width="50">

<TextBlock Text="{Binding Name}" />

<WrapPanel>

<TextBlock Text="{Binding CurrentHitPoints, UpdateSourceTrigger=PropertyChanged}" />

<TextBlock Text="/" />

<TextBlock Text="{Binding HitPoints}" />

</WrapPanel>

<TextBlock Text="{Binding InitiativeCats}" />

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</WrapPanel>

<!-- Bottom "End Turn" Button -->

<StackPanel DockPanel.Dock="Bottom"

Orientation="Horizontal"

VerticalAlignment="Bottom"

Height="50"

HorizontalAlignment="Center">

<Button Content="End Turn"

Command="{Binding EndTurnCommand}"

Width="100"

Height="30"

Margin="10" />

</StackPanel>

<!-- Middle Character Panel -->

<DockPanel VerticalAlignment="Center"

HorizontalAlignment="Center"

Width="auto"

Height="auto">

<!-- Add controls for character details here -->

<TabControl>

<TabItem Header="Information">

<DockPanel>

<StackPanel DataContext="{Binding SelectedInBattle}"

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

<DockPanel>

<DockPanel DockPanel.Dock="Left"

Margin="10,10,10,10"

VerticalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

Text="ARMOR"

HorizontalAlignment="Center"

FontSize="15" />

<TextBlock DockPanel.Dock="Bottom"

Text="{Binding ArmorClass}"

HorizontalAlignment="Center"

FontSize="15" />

<TextBlock Text="ClASS"

HorizontalAlignment="Center"

FontSize="15" />

</DockPanel>

<DockPanel DockPanel.Dock="Right"

Margin="10,10,10,10"

HorizontalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

Text="SPEED"

HorizontalAlignment="Center"

FontSize="15" />

<TextBlock Text="{Binding Speed}"

HorizontalAlignment="Center"

VerticalAlignment="Center"

FontSize="15" />

</DockPanel>

<DockPanel Margin="10,10,10,10">

<TextBlock DockPanel.Dock="Top"

Text="INITITIVE CAST"

HorizontalAlignment="Center"

FontSize="15" />

<TextBlock Text="{Binding InitiativeCats}"

HorizontalAlignment="Center"

Padding="25,10,25,10"

FontSize="15" />

</DockPanel>

</DockPanel>

<DockPanel>

<TextBlock DockPanel.Dock="Top"

Text="Hit Points"

HorizontalAlignment="Center" />

<DockPanel DockPanel.Dock="Left"

Margin="10,10,10,10"

HorizontalAlignment="Center">

<TextBox DockPanel.Dock="Top"

Text="{Binding CurrentHitPoints}"

HorizontalAlignment="Center"

FontSize="15" />

<TextBlock Text="Temp"

FontSize="15" />

</DockPanel>

<DockPanel DockPanel.Dock="Right"

Margin="10,10,10,10"

HorizontalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

Text="{Binding HitPoints}"

HorizontalAlignment="Center"

FontSize="15" />

<TextBlock Text="Max"

FontSize="15" />

</DockPanel>

<DockPanel Margin="10,10,10,10"

HorizontalAlignment="Center"

VerticalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

HorizontalAlignment="Center"

FontSize="25"

Text="/" />

</DockPanel>

</DockPanel>

<DockPanel>

<DockPanel DockPanel.Dock="Left">

<TextBlock DockPanel.Dock="Top"

HorizontalAlignment="Center"

Text="Hit Dice"

FontSize="15" />

<DockPanel DockPanel.Dock="Left"

Margin="10,10,10,10"

HorizontalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

Text="{Binding CurrentHitPointDice}"

HorizontalAlignment="Center"

FontSize="15" />

<TextBlock Text="Temp"

FontSize="15" />

</DockPanel>

<DockPanel DockPanel.Dock="Right"

Margin="10,10,10,10"

HorizontalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

Text="{Binding HitPointDice.Count }"

HorizontalAlignment="Center"

FontSize="15" />

<TextBlock Text="Max"

FontSize="15" />

</DockPanel>

<DockPanel Margin="10,10,10,10"

HorizontalAlignment="Center"

VerticalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

HorizontalAlignment="Center"

FontSize="25"

Text="/" />

</DockPanel>

</DockPanel>

<DockPanel DockPanel.Dock="Right">

<TextBlock DockPanel.Dock="Top"

HorizontalAlignment="Center"

Text="Death Saves"

FontSize="15" />

<DockPanel DockPanel.Dock="Left"

Margin="10,10,10,10"

HorizontalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

Text="{Binding DeathSaveTrue}"

HorizontalAlignment="Center"

FontSize="15" />

<TextBlock Text="True"

FontSize="15" />

</DockPanel>

<DockPanel DockPanel.Dock="Right"

Margin="10,10,10,10"

HorizontalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

Text="{Binding DeathSaveFalse}"

HorizontalAlignment="Center"

FontSize="15" />

<TextBlock Text="False"

FontSize="15" />

</DockPanel>

<DockPanel Margin="10,10,10,10"

HorizontalAlignment="Center"

VerticalAlignment="Center">

<TextBlock DockPanel.Dock="Top"

HorizontalAlignment="Center"

FontSize="25"

Text="/" />

</DockPanel>

</DockPanel>

<DockPanel></DockPanel>

</DockPanel>

</StackPanel>

<DockPanel></DockPanel>

</DockPanel>

</TabItem>

<TabItem Header="Stat">

<Grid DataContext="{Binding SelectedInBattle}">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="auto" />

<RowDefinition Height="auto" />

</Grid.RowDefinitions>

<!--Сила-->

<StackPanel Grid.Column="0"

Grid.Row="0"

Margin="5,5,0,10">

<!--Имя характеристики и ее значение-->

<WrapPanel>

<TextBlock Text="{Binding Strength.Name}"

Margin="0,0,5,0"

FontWeight="Bold" />

<TextBlock Text="{Binding Strength.Value}"

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

</WrapPanel>

<!--Проверка и Спасбросок-->

<WrapPanel>

<WrapPanel>

<TextBlock Text="Проверка:"

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

<Button Content="{Binding Strength.Modifier}"

Command="{Binding Strength.StatCastCommand}"

Padding="5,0,5,0"

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

</WrapPanel>

<WrapPanel>

<TextBlock Text="СпасБросок:"

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

<WrapPanel>

<Button Content="{Binding Strength.SavingThrows.Modifier}"

Command="{Binding Strength.SavingThrows.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Strength.SavingThrows.Proficient, Mode=TwoWay}"

FontStyle="Italic"

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

</WrapPanel>

</WrapPanel>

</WrapPanel>

<!--Навыки-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Strength.Athletics.Modifier}"

Command="{Binding Strength.Athletics.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Strength.Athletics.Proficient, Mode=TwoWay}"

Content="{Binding Strength.Athletics.Name}"

FontStyle="Italic"

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

</WrapPanel>

</StackPanel>

<!--Ловкость-->

<StackPanel Grid.Column="1"

Grid.Row="0"

Margin="5,5,0,10">

<!--Имя характеристики и ее значение-->

<WrapPanel>

<TextBlock Text="{Binding Dexterity.Name}"

Margin="0,0,5,0"

FontWeight="Bold" />

<TextBlock Text="{Binding Dexterity.Value}"

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

</WrapPanel>

<!--Проверка и Спасбросок-->

<WrapPanel>

<WrapPanel>

<TextBlock Text="Проверка:"

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

<Button Content="{Binding Dexterity.Modifier}"

Command="{Binding Dexterity.StatCastCommand}"

Padding="5,0,5,0"

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

</WrapPanel>

<WrapPanel>

<TextBlock Text="СпасБросок:"

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

<WrapPanel>

<Button Content="{Binding Dexterity.SavingThrows.Modifier}"

Command="{Binding Dexterity.SavingThrows.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Dexterity.SavingThrows.Proficient, Mode=TwoWay}"

FontStyle="Italic"

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

</WrapPanel>

</WrapPanel>

</WrapPanel>

<!--Навыки-->

<!--Акробатика-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Dexterity.Acrobatics.Modifier}"

Command="{Binding Dexterity.Acrobatics.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Dexterity.Acrobatics.Proficient, Mode=TwoWay}"

Content="{Binding Dexterity.Acrobatics.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Ловкость рук-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Dexterity.SleightOfHand.Modifier}"

Command="{Binding Dexterity.SleightOfHand.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Dexterity.SleightOfHand.Proficient, Mode=TwoWay}"

Content="{Binding Dexterity.SleightOfHand.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Скрытность-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Dexterity.Stealth.Modifier}"

Command="{Binding Dexterity.Stealth.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Dexterity.Stealth.Proficient, Mode=TwoWay}"

Content="{Binding Dexterity.Stealth.Name}"

FontStyle="Italic"

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

</WrapPanel>

</StackPanel>

<!--Телосложение-->

<StackPanel Grid.Column="0"

Grid.Row="1"

Margin="5,5,0,10">

<!--Имя характеристики и ее значение-->

<WrapPanel>

<TextBlock Text="{Binding Constitution.Name}"

FontWeight="Bold"

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

<TextBlock Text="{Binding Constitution.Value}"

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

</WrapPanel>

<!--Проверка и Спасбросок-->

<WrapPanel>

<WrapPanel>

<TextBlock Text="Проверка:"

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

<Button Content="{Binding Constitution.Modifier}"

Command="{Binding Constitution.StatCastCommand}"

Padding="5,0,5,0"

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

</WrapPanel>

<WrapPanel>

<TextBlock Text="СпасБросок:"

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

<WrapPanel>

<Button Content="{Binding Constitution.SavingThrows.Modifier}"

Command="{Binding Constitution.SavingThrows.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Constitution.SavingThrows.Proficient, Mode=TwoWay}"

FontStyle="Italic"

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

</WrapPanel>

</WrapPanel>

</WrapPanel>

</StackPanel>

<!--Интелект-->

<StackPanel Grid.Column="2"

Grid.Row="0"

Margin="5,5,0,10">

<!--Имя характеристики и ее значение-->

<WrapPanel>

<TextBlock Text="{Binding Intelligence.Name}"

Margin="0,0,5,0"

FontWeight="Bold" />

<TextBlock Text="{Binding Intelligence.Value}"

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

</WrapPanel>

<!--Проверка и Спасбросок-->

<WrapPanel>

<WrapPanel>

<TextBlock Text="Проверка:"

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

<Button Content="{Binding Intelligence.Modifier}"

Command="{Binding Intelligence.StatCastCommand}"

Padding="5,0,5,0"

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

</WrapPanel>

<WrapPanel>

<TextBlock Text="СпасБросок:"

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

<WrapPanel>

<Button Content="{Binding Intelligence.SavingThrows.Modifier}"

Command="{Binding Intelligence.SavingThrows.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Intelligence.SavingThrows.Proficient, Mode=TwoWay}"

FontStyle="Italic"

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

</WrapPanel>

</WrapPanel>

</WrapPanel>

<!--Навыки-->

<!--Магия-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Intelligence.Arcana.Modifier}"

Command="{Binding Intelligence.Arcana.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Intelligence.Arcana.Proficient, Mode=TwoWay}"

Content="{Binding Intelligence.Arcana.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--История-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Intelligence.History.Modifier}"

Command="{Binding Intelligence.History.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Intelligence.History.Proficient, Mode=TwoWay}"

Content="{Binding Intelligence.History.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Анализ-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Intelligence.Investigation.Modifier}"

Command="{Binding Intelligence.Investigation.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Intelligence.Investigation.Proficient, Mode=TwoWay}"

Content="{Binding Intelligence.Investigation.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Природа-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Intelligence.Nature.Modifier}"

Command="{Binding Intelligence.Nature.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Intelligence.Nature.Proficient, Mode=TwoWay}"

Content="{Binding Intelligence.Nature.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Религия-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Intelligence.Religion.Modifier}"

Command="{Binding Intelligence.Religion.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Intelligence.Religion.Proficient, Mode=TwoWay}"

Content="{Binding Intelligence.Religion.Name}"

FontStyle="Italic"

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

</WrapPanel>

</StackPanel>

<!--Мудрость-->

<StackPanel Grid.Column="2"

Grid.Row="1"

Margin="5,5,0,10">

<!--Имя характеристики и ее значение-->

<WrapPanel>

<TextBlock Text="{Binding Wisdom.Name}"

Margin="0,0,5,0"

FontWeight="Bold" />

<TextBlock Text="{Binding Wisdom.Value}"

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

</WrapPanel>

<!--Проверка и Спасбросок-->

<WrapPanel>

<WrapPanel>

<TextBlock Text="Проверка:"

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

<Button Content="{Binding Wisdom.Modifier}"

Command="{Binding Wisdom.StatCastCommand}"

Padding="5,0,5,0"

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

</WrapPanel>

<WrapPanel>

<TextBlock Text="СпасБросок:"

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

<WrapPanel>

<Button Content="{Binding Wisdom.SavingThrows.Modifier}"

Command="{Binding Wisdom.SavingThrows.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Wisdom.SavingThrows.Proficient, Mode=TwoWay}"

FontStyle="Italic"

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

</WrapPanel>

</WrapPanel>

</WrapPanel>

<!--Навыки-->

<!--Уход за животными-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Wisdom.AnimalHandling.Modifier}"

Command="{Binding Wisdom.AnimalHandling.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Wisdom.AnimalHandling.Proficient, Mode=TwoWay}"

Content="{Binding Wisdom.AnimalHandling.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Проницательность-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Wisdom.Insight.Modifier}"

Command="{Binding Wisdom.Insight.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Wisdom.Insight.Proficient, Mode=TwoWay}"

Content="{Binding Wisdom.Insight.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Медицина-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Wisdom.Medicine.Modifier}"

Command="{Binding Wisdom.Medicine.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Wisdom.Medicine.Proficient, Mode=TwoWay}"

Content="{Binding Wisdom.Medicine.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Восприятие-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Wisdom.Perception.Modifier}"

Command="{Binding Wisdom.Perception.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Wisdom.Perception.Proficient, Mode=TwoWay}"

Content="{Binding Wisdom.Perception.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Выживание-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Wisdom.Survival.Modifier}"

Command="{Binding Wisdom.Survival.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Wisdom.Survival.Proficient, Mode=TwoWay}"

Content="{Binding Wisdom.Survival.Name}"

FontStyle="Italic"

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

</WrapPanel>

</StackPanel>

<!--Харизма-->

<StackPanel Grid.Column="1"

Grid.Row="2"

Margin="5,5,0,10">

<!--Имя характеристики и ее значение-->

<WrapPanel>

<TextBlock Text="{Binding Charisma.Name}"

Margin="0,0,5,0"

FontWeight="Bold" />

<TextBlock Text="{Binding Charisma.Value}"

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

</WrapPanel>

<!--Проверка и Спасбросок-->

<WrapPanel>

<WrapPanel>

<TextBlock Text="Проверка:"

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

<Button Content="{Binding Charisma.Modifier}"

Command="{Binding Charisma.StatCastCommand}"

Padding="5,0,5,0"

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

</WrapPanel>

<WrapPanel>

<TextBlock Text="СпасБросок:"

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

<WrapPanel>

<Button Content="{Binding Charisma.SavingThrows.Modifier}"

Command="{Binding Charisma.SavingThrows.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Charisma.SavingThrows.Proficient, Mode=TwoWay}"

FontStyle="Italic"

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

</WrapPanel>

</WrapPanel>

</WrapPanel>

<!--Навыки-->

<!--Обман-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Charisma.Deception.Modifier}"

Command="{Binding Charisma.Deception.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Charisma.Deception.Proficient, Mode=TwoWay}"

Content="{Binding Charisma.Deception.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Устрашение-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Charisma.Intimidation.Modifier}"

Command="{Binding Charisma.Intimidation.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Charisma.Intimidation.Proficient, Mode=TwoWay}"

Content="{Binding Charisma.Intimidation.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Представление-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Charisma.Performance.Modifier}"

Command="{Binding Charisma.Performance.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Charisma.Performance.Proficient, Mode=TwoWay}"

Content="{Binding Charisma.Performance.Name}"

FontStyle="Italic"

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

</WrapPanel>

<!--Убеждение-->

<WrapPanel Margin="10,5,0,0">

<Button Content="{Binding Charisma.Persuasion.Modifier}"

Command="{Binding Charisma.Persuasion.SkillCastCommand}"

Padding="5,0,5,0"

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

<RadioButton IsChecked="{Binding Charisma.Persuasion.Proficient, Mode=TwoWay}"

Content="{Binding Charisma.Persuasion.Name}"

FontStyle="Italic"

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

</WrapPanel>

</StackPanel>

</Grid>

</TabItem>

</TabControl>

</DockPanel>

</DockPanel>

</TabItem>

<TabItem Header="Log">

<DockPanel>

<ListBox Name="LogList"

ItemsSource="{Binding Log}">

<ListBox.ItemTemplate>

<DataTemplate>

<TextBlock Text="{Binding }"/>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</DockPanel>

</TabItem>

</TabControl>

</Grid>

</Window>

**Приложение В: MainVM.cs**

using dnd\_char\_create.model;

using dnd\_char\_create.model.Base;

using dnd\_char\_create.View;

using Labrary.Model.Base;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

namespace dnd\_char\_create.VIewModel

{

public class MainVM : Notify

{

private string str\_start\_box;

public string Str\_start\_box

{

get { return str\_start\_box; }

set

{

str\_start\_box = value;

OnPropertyChanged(nameof(Str\_start\_box));

}

}

private string dex\_start\_box;

public string Dex\_start\_box

{

get { return dex\_start\_box; }

set

{

dex\_start\_box = value;

OnPropertyChanged(nameof(Dex\_start\_box));

}

}

private string con\_start\_box;

public string Con\_start\_box

{

get { return con\_start\_box; }

set

{

con\_start\_box = value;

OnPropertyChanged(nameof(Con\_start\_box));

}

}

private string inte\_start\_box;

public string Inte\_start\_box

{

get { return inte\_start\_box; }

set

{

inte\_start\_box = value;

OnPropertyChanged(nameof(Inte\_start\_box));

}

}

private string wis\_start\_box;

public string Wis\_start\_box

{

get { return wis\_start\_box; }

set

{

wis\_start\_box = value;

OnPropertyChanged(nameof(Wis\_start\_box));

}

}

private string cha\_start\_box;

public string Cha\_start\_box

{

get { return cha\_start\_box; }

set

{

cha\_start\_box = value;

OnPropertyChanged(nameof(Cha\_start\_box));

}

}

private Character character\_make;

public Character Character\_make

{

get { return character\_make; }

set

{

character\_make = value;

OnPropertyChanged(nameof(Character\_make));

}

}

private Rase selectedRase;

public Rase SelectedRase

{

get { return selectedRase; }

set

{

selectedRase = value;

OnPropertyChanged(nameof(SelectedRase));

}

}

private dndClass selectedClass;

public dndClass SelectedClass

{

get { return selectedClass; }

set

{

selectedClass = value;

OnPropertyChanged(nameof(SelectedClass));

}

}

private string lab;

public string Lab

{

get { return lab; }

set { lab = value; OnPropertyChanged(nameof(Lab)); }

}

private RelayCommand stats\_create;

public RelayCommand Stats\_create

{

get

{

return stats\_create ??

(stats\_create = new RelayCommand(obj =>

{

int[] arr = { 0, 0, 0, 0, 0, 0 };

int min = 7;

int a = 0;

Random rnd = new Random();

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 4; j++)

{

a = rnd.Next(1, 6);

arr[i] += a;

if (a < min) min = a;

}

arr[i] -= min;

min = 7;

}

Lab = arr[0].ToString() + " " + arr[1].ToString() + " " + arr[2].ToString() + " " + arr[3].ToString() + " " + arr[4].ToString() + " " + arr[5].ToString();

}));

}

}

private RelayCommand char\_create;

public RelayCommand Char\_create

{

get

{

return char\_create ??

(char\_create = new RelayCommand(obj =>

{

if ((SelectedRase != null) && (SelectedClass != null) && (Dex\_start\_box != null) && (Con\_start\_box != null) && (Inte\_start\_box != null) && (Wis\_start\_box != null) && (Cha\_start\_box != null) && (Str\_start\_box != null))

{

string check\_cha="1";

bool check\_t\_f = true;

for (int i=0;i<6; i++)

{

if (i == 0)

{

check\_cha = Str\_start\_box;

}

if (i == 1)

{

check\_cha = Dex\_start\_box;

}

if (i == 2)

{

check\_cha = Con\_start\_box;

}

if (i == 3)

{

check\_cha = Inte\_start\_box;

}

if (i == 4)

{

check\_cha = Wis\_start\_box;

}

if (i == 5)

{

check\_cha = Cha\_start\_box;

}

for (int j = 0; j < check\_cha.Length; j++)

{

if (!(check\_cha[j] >= '0' && check\_cha[j] <= '9'))

{

check\_t\_f=false;

break;

}

}

if (!check\_t\_f)

{

break;

}

}

if (check\_t\_f)

{

Character\_make = new Character(Convert.ToInt32(Str\_start\_box), Convert.ToInt32(Dex\_start\_box), Convert.ToInt32(Con\_start\_box), Convert.ToInt32(Inte\_start\_box), Convert.ToInt32(Wis\_start\_box), Convert.ToInt32(Cha\_start\_box), SelectedRase, SelectedClass);

Char\_Show ch = new Char\_Show(Character\_make);

ch.Show();

}

else

{

MessageBox.Show("Поля заполнены некорректно");

}

}

else

{

MessageBox.Show("Заполнены не все поля");

}

}));

}

}

}

}

**Приложение Г: MainVM2.cs**

using dnd\_char\_create.model;

using dnd\_char\_create.model.Base;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace dnd\_char\_create.VIewModel

{

public class MainVM2 :Notify

{

Character character;

public MainVM2(Character cr)

{

character = cr;

}

private string str\_cha;

public string Str\_cha

{

get { return character.stats.Str.ToString(); }

}

private string dex\_cha;

public string Dex\_cha

{

get { return character.stats.Dex.ToString(); }

}

private string con\_cha;

public string Con\_cha

{

get { return character.stats.Con.ToString(); }

}

private string inte\_cha;

public string Inte\_cha

{

get { return character.stats.Inte.ToString(); }

}

private string wis\_cha;

public string Wis\_cha

{

get { return character.stats.Wis.ToString(); }

}

private string cha\_cha;

public string Cha\_cha

{

get { return character.stats.Cha.ToString(); }

}

private string rase\_cha;

public string Rase\_cha

{

get { return character.stats.Rase.Name; }

}

private string class\_cha;

public string Class\_cha

{

get { return character.stats.Game\_class.Name; }

}

private string str\_dise\_cha;

public string Str\_dise\_cha

{

get { return character.stats.Str\_dise.ToString(); }

}

private string dex\_dise\_cha;

public string Dex\_dise\_cha

{

get { return character.stats.Dex\_dise.ToString(); }

}

private string con\_dise\_cha;

public string Con\_dise\_cha

{

get { return character.stats.Con\_dise.ToString(); }

}

private string inte\_dise\_cha;

public string Inte\_dise\_cha

{

get { return character.stats.Inte\_dise.ToString(); }

}

private string wis\_dise\_cha;

public string Wis\_dise\_cha

{

get { return character.stats.Wis\_dise.ToString(); }

}

private string cha\_dise\_cha;

public string Cha\_dise\_cha

{

get { return character.stats.Cha\_dise.ToString(); }

}

}

}

**Приложение Д: Char\_Show.xaml**

<Window x:Class="dnd\_char\_create.View.Char\_Show"

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

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

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

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

xmlns:local="clr-namespace:dnd\_char\_create.View" xmlns:viewmodel="clr-namespace:dnd\_char\_create.VIewModel" d:DataContext="{d:DesignInstance Type=viewmodel:MainVM}"

mc:Ignorable="d"

Title="MainWindow" Height="400" Width="300" ResizeMode="NoResize">

<Window.Resources>

<!-- Стиль для ListView -->

<Style TargetType="ListView">

<Setter Property="Background" Value="#8a3333"/>

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

<Setter Property="BorderBrush" Value="#FF2B2B"/>

<Setter Property="BorderThickness" Value="1"/>

<Setter Property="Margin" Value="20"/>

<Setter Property="FontFamily" Value="Times New Roman"/>

<Style.Triggers>

<!-- Эффект при наведении -->

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

<Setter Property="Background" Value="#8a3333"/>

</Trigger>

</Style.Triggers>

</Style>

<!-- Стиль для GridViewColumnHeader -->

<Style TargetType="GridViewColumnHeader">

<Setter Property="Background" Value="#293133"/>

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

<Setter Property="BorderBrush" Value="#FF2B2B"/>

<Setter Property="BorderThickness" Value="1"/>

<Setter Property="Height" Value="40"/>

<Setter Property="FontFamily" Value="Times New Roman"/>

<Style.Triggers>

<!-- Эффект при наведении -->

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

<Setter Property="Background" Value="#2A363B"/>

</Trigger>

</Style.Triggers>

</Style>

<!-- Стиль для кнопки -->

<Style TargetType="Button">

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

<Setter Property="Background" Value="#FF2B2B"/>

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

<Setter Property="Margin" Value="10"/>

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

<Setter Property="FontSize" Value="18"/>

<Setter Property="FontFamily" Value="Times New Roman"/>

<Setter Property="BorderThickness" Value="1"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" CornerRadius="5">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

<Window.Background>

<!-- Добавление фоновой картинки -->

<ImageBrush ImageSource="C:\Users\79526\OneDrive\Рабочий стол\Для лекций по истории\dnd\_char\_create\ac2ea54e87f2d81783143d8b5c4e1e26.jpg"/>

</Window.Background>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal">

<ListView Name="charStats">

<StackPanel Orientation="Horizontal">

<Label Content="{Binding Str\_cha}"/>

<Label>Str</Label>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="{Binding Str\_dise\_cha}"/>

<Label>StrDise</Label>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="{Binding Dex\_cha}"/>

<Label>Dex</Label>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="{Binding Dex\_dise\_cha}"/>

<Label>DexDise</Label>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="{Binding Con\_cha}"/>

<Label>Con</Label>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="{Binding Con\_dise\_cha}"/>

<Label>ConDise</Label>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="{Binding Inte\_cha}"/>

<Label>Inte</Label>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="{Binding Inte\_dise\_cha}"/>

<Label>InteDise</Label>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="{Binding Wis\_cha}"/>

<Label>Wis</Label>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="{Binding Cha\_cha}"/>

<Label>Cha</Label>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="{Binding Cha\_dise\_cha}"/>

<Label>ChaDise</Label>

</StackPanel>

</ListView>

<StackPanel Orientation="Vertical">

<ListView>

<Label Content="{Binding Rase\_cha}"></Label>

</ListView>

<ListView>

<Label Content="{Binding Class\_cha}"></Label>

</ListView>

</StackPanel>

</StackPanel>

</Grid>

</Window>

## Приложение Е: Template. Cs

using DndHelper.Model.Stats;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Diagnostics;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Security.AccessControl;

using System.Security.Permissions;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Controls.Primitives;

using System.Windows.Controls;

using System.Windows.Media;

using System.Windows.Threading;

using System.Windows;

namespace DndHelper.Model

{

public class Template : PropPropertyChanged

{

private string name;

public string Name

{

get { return name; }

set

{

name = value;

OnPropertyChanged(nameof(Name));

}

}

private string race;

public string Race

{

get { return race; }

set

{

race = value;

OnPropertyChanged(nameof(Race));

}

}

private bool isFriend;

public bool IsFriend

{

get { return isFriend; }

set

{

isFriend = value;

OnPropertyChanged(nameof(IsFriend));

}

}

private int level;

public int Level

{

get { return level; }

set

{

level = value;

OnPropertyChanged(nameof(Level));

UpdateMasteryBonus();

}

}

private int hitPoints;

public int HitPoints

{

get { return hitPoints; }

set

{

hitPoints = value;

CurrentHitPoints = value;

OnPropertyChanged(nameof(HitPoints));

}

}

private int currentHitPoints;

public int CurrentHitPoints

{

get { return currentHitPoints; }

set

{

currentHitPoints = value;

OnPropertyChanged(nameof(CurrentHitPoints));

}

}

private int deathSaveTrue = 0;

public int DeathSaveTrue

{

get { return deathSaveTrue; }

set

{

deathSaveTrue = value;

OnPropertyChanged(nameof(DeathSaveTrue));

}

}

private int deathSaveFalse = 0;

public int DeathSaveFalse

{

get { return deathSaveFalse; }

set

{

deathSaveFalse = value;

OnPropertyChanged(nameof(DeathSaveFalse));

}

}

private int armorClass;

public int ArmorClass

{

get { return armorClass; }

set

{

armorClass = value;

OnPropertyChanged(nameof(ArmorClass));

}

}

private int masteryBonus;

public int MasteryBonus

{

get { return masteryBonus; }

set

{

masteryBonus = value;

OnPropertyChanged(nameof(MasteryBonus));

OnPropertyChanged(nameof(Strength));

OnPropertyChanged(nameof(Dexterity));

OnPropertyChanged(nameof(Constitution));

OnPropertyChanged(nameof(Intelligence));

OnPropertyChanged(nameof(Wisdom));

OnPropertyChanged(nameof(Charisma));

}

}

private int speed;

public int Speed

{

get { return speed; }

set

{

speed = value;

OnPropertyChanged(nameof(Speed));

}

}

private int passivePerception;

public int PassivePerception

{

get { return passivePerception + Wisdom.Modifier; }

set

{

passivePerception = value;

OnPropertyChanged(nameof(PassivePerception));

}

}

private int initiative;

public int Initiative

{

get { return initiative + Dexterity.Modifier; }

set

{

initiative = value;

OnPropertyChanged(nameof(InitiativeCats));

}

}

private int initiativeCast;

public int InitiativeCats

{

get { return initiativeCast + Initiative; }

set

{

initiativeCast = value;

OnPropertyChanged(nameof(InitiativeCats));

}

}

private Strength strength;

private int strengthValue;

public int StrengthValue

{

get { return strengthValue; }

set

{

strengthValue = value;

OnPropertyChanged(nameof(StrengthValue));

}

}

public Strength Strength

{

get { return strength; }

set

{

strength = value;

strength.Value = StrengthValue;

strength.Character = this;

OnPropertyChanged(nameof(Strength));

}

}

private Dexterity dexterity;

private int dexterityValue;

public int DexterityValue

{

get { return dexterityValue; }

set

{

dexterityValue = value;

OnPropertyChanged(nameof(DexterityValue));

}

}

public Dexterity Dexterity

{

get { return dexterity; }

set

{

dexterity = value;

dexterity.Value = DexterityValue;

dexterity.Character = this;

OnPropertyChanged(nameof(Dexterity));

}

}

private Constitution constitution;

private int constitutionValue;

public int ConstitutionValue

{

get { return constitutionValue; }

set

{

constitutionValue = value;

OnPropertyChanged(nameof(ConstitutionValue));

}

}

public Constitution Constitution

{

get { return constitution; }

set

{

constitution = value;

constitution.Value = ConstitutionValue;

constitution.Character = this;

OnPropertyChanged(nameof(Constitution));

}

}

private Intelligence intelligence;

private int intelligenceValue;

public int IntelligenceValue

{

get { return intelligenceValue; }

set

{

intelligenceValue = value;

OnPropertyChanged(nameof(IntelligenceValue));

}

}

public Intelligence Intelligence

{

get { return intelligence; }

set

{

intelligence = value;

intelligence.Value = IntelligenceValue;

intelligence.Character = this;

OnPropertyChanged(nameof(Intelligence));

}

}

private Wisdom wisdom;

private int wisdomValue;

public int WisdomValue

{

get { return wisdomValue; }

set

{

wisdomValue = value;

OnPropertyChanged(nameof(WisdomValue));

}

}

public Wisdom Wisdom

{

get { return wisdom; }

set

{

wisdom = value;

wisdom.Value = WisdomValue;

wisdom.Character = this;

OnPropertyChanged(nameof(Wisdom));

}

}

private Charisma charisma;

private int charismaValue;

public int CharismaValue

{

get { return charismaValue; }

set

{

charismaValue = value;

OnPropertyChanged(nameof(CharismaValue));

}

}

public Charisma Charisma

{

get { return charisma; }

set

{

charisma = value;

charisma.Value = CharismaValue;

charisma.Character = this;

OnPropertyChanged(nameof(Charisma));

}

}

private Dice hitPointDice;

public Dice HitPointDice

{

get { return hitPointDice; }

set

{

hitPointDice = value;

HitPoints = hitPointDice.Cast();

OnPropertyChanged(nameof(HitPointDice));

}

}

public Template(

int strengthValue = 10,

int dexterityValue = 10,

int constitutionValue = 10,

int intelligenceValue = 10,

int wisdomValue = 10,

int charismaValue = 10)

{

StrengthValue = strengthValue;

DexterityValue = dexterityValue;

ConstitutionValue = constitutionValue;

IntelligenceValue = intelligenceValue;

WisdomValue = wisdomValue;

CharismaValue = charismaValue;

Strength = new Strength();

Dexterity = new Dexterity();

Constitution = new Constitution();

Intelligence = new Intelligence();

Wisdom = new Wisdom();

Charisma = new Charisma();

ArmorClass = 10 + Dexterity.Modifier;

UpdateMasteryBonus();

Speed = 30;

PassivePerception = 10;

Initiative = 0;

}

public void UpdateMasteryBonus()

{

if (Level <= 4)

{

MasteryBonus = 2;

}

else if (Level <= 8)

{

MasteryBonus = 3;

}

else if (Level <= 12)

{

MasteryBonus = 4;

}

else if (Level <= 16)

{

MasteryBonus = 5;

}

else if (Level <= 20)

{

MasteryBonus = 6;

}

}

}

}

## Приложение Ж: Character. cs

using DndHelper.Model.Races;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Diagnostics;

using System.Linq;

using System.Security.Policy;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Markup;

using System.Xml.Linq;

namespace DndHelper.Model

{

public class Character : Template

{

private int exhaustion = 0;

public int Exhaustion

{

get { return exhaustion; }

set

{

exhaustion = value;

OnPropertyChanged(nameof(Exhaustion));

}

}

private int level;

public new int Level

{

get { return level; }

set

{

level = value;

switch (value)

{

case 1:

HitPointDice.Count = 1;

HitPoints = HitPointDice.Type + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 2:

HitPointDice.Count = 2;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 3:

HitPointDice.Count = 3;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 4:

HitPointDice.Count = 4;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 5:

HitPointDice.Count = 5;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 6:

HitPointDice.Count = 6;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 7:

HitPointDice.Count = 7;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 8:

HitPointDice.Count = 8;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 9:

HitPointDice.Count = 9;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 10:

HitPointDice.Count = 10;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 11:

HitPointDice.Count = 11;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 12:

HitPointDice.Count = 12;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 13:

HitPointDice.Count = 13;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 14:

HitPointDice.Count = 14;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 15:

HitPointDice.Count = 15;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 16:

HitPointDice.Count = 16;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 17:

HitPointDice.Count = 17;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 18:

HitPointDice.Count = 18;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 19:

HitPointDice.Count = 19;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

case 20:

HitPointDice.Count = 20;

HitPoints = HitPointDice.Cast() + Constitution.Modifier;

CurrentHitPoints = HitPoints;

CurrentHitPointDice = HitPointDice.Count;

break;

}

OnPropertyChanged(nameof(Level));

UpdateMasteryBonus();

}

}

private int hitPoints;

public new int HitPoints

{

get { return hitPoints; }

set

{

hitPoints = value;

OnPropertyChanged(nameof(HitPoints));

}

}

private int currentHitPointDice;

public int CurrentHitPointDice

{

get { return currentHitPointDice; }

set

{

currentHitPointDice = value;

OnPropertyChanged(nameof(CurrentHitPointDice));

}

}

private string classCharacter;

public string ClassCharacter

{

get { return classCharacter; }

set

{

classCharacter = value;

OnPropertyChanged(nameof(ClassCharacter));

UpdateMasteryBonus();

}

}

private ObservableCollection<string> abilities;

public ObservableCollection<string> Abilities

{

get { return abilities; }

set

{

abilities = value;

OnPropertyChanged(nameof(Abilities));

}

}

private Race race;

public new Race Race

{

get { return race; }

set

{

race = value;

OnPropertyChanged(nameof(Race));

}

}

public Character() : base()

{

Exhaustion = 0;

}

}

}

## Приложение З: Dice. cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DndHelper.Model

{

public class Dice : PropPropertyChanged

{

private int count;

private int type;

public int Count

{

get { return count; }

set

{

count = value;

OnPropertyChanged(nameof(Count));

}

}

public int Type

{

get { return type; }

set

{

type = value;

OnPropertyChanged(nameof(Type));

}

}

public int Cast()

{

Random random = new Random();

int result = 0;

for (int i = 0; i < Count; i++)

{

// Генерируем случайное число от 1 до Type и добавляем Modifier

int roll = random.Next(1, Type + 1);

// Добавляем результат к общему суммарному значению

result += roll;

}

return result;

}

}

}

## Приложение И: Skill. cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Controls.Primitives;

using System.Windows.Media;

using System.Windows.Threading;

using DndHelper.Model.Stats;

using DndHelper.ViewModel;

namespace DndHelper.Model

{

public class Skill : PropPropertyChanged

{

private string name;

private bool proficient;

private int modifier;

private int value;

private Stat stat;

private Dice dice;

private RelayCommand skillCastCommand;

public RelayCommand SkillCastCommand

{

get

{

return skillCastCommand ?? (skillCastCommand = new RelayCommand(obj => {

// Создание объекта Popup

Popup messagePopup = new Popup();

messagePopup.StaysOpen = false;

// Текст сообщения

messagePopup.PlacementTarget = Application.Current.MainWindow;

// Создание объекта Border для стилизации внешнего вида Popup

Border popupBorder = new Border();

popupBorder.Background = Brushes.White; // Цвет фона

popupBorder.BorderBrush = Brushes.Black; // Цвет границы

popupBorder.BorderThickness = new Thickness(1); // Толщина границы

// Текст сообщения

TextBlock messageText = new TextBlock();

messageText.Text = Cast();

ApplicationViewModel.Log.Add($"{Stat.Character.Name}: {messageText.Text}");

messageText.Padding = new Thickness(5, 10, 5, 10);

// Добавление элементов в Border

popupBorder.Child = messageText;

// Установка Border в качестве содержимого Popup

messagePopup.Child = popupBorder;

messagePopup.Placement = PlacementMode.Custom;

// Установка пользовательского расположения

messagePopup.CustomPopupPlacementCallback = (popupSize, targetSize, offset) =>

{

// Расположение в правом нижнем углу

double x = targetSize.Width - popupSize.Width\*1.5;

double y = targetSize.Height - popupSize.Height\*3;

return new[] { new CustomPopupPlacement(new Point(x, y), PopupPrimaryAxis.None) };

};

messagePopup.IsOpen = true;

// Создайте таймер

DispatcherTimer timer = new DispatcherTimer();

// Установите интервал таймера в 10 секунд

timer.Interval = TimeSpan.FromSeconds(3);

// Обработчик события таймера

timer.Tick += (sender, e) =>

{

// Закрыть Popup после истечения времени

messagePopup.IsOpen = false;

// Остановить таймер

timer.Stop();

};

// Запустите таймер

timer.Start();

}));

}

}

private SolidColorBrush skillBackground = Brushes.Aquamarine; // Используйте начальный цвет фона

public SolidColorBrush SkillBackground

{

get { return skillBackground; }

set

{

if (skillBackground != value)

{

skillBackground = value;

OnPropertyChanged(nameof(SkillBackground));

}

}

}

public string Name

{

get => name;

set

{

if (name != value)

{

name = value;

OnPropertyChanged(nameof(Name));

}

}

}

public bool Proficient

{

get { return proficient; }

set

{

if (proficient != value)

{

proficient = value;

OnPropertyChanged(nameof(Modifier));

OnPropertyChanged(nameof(Proficient));

}

}

}

public int Modifier

{

get { if (Proficient == true) { return modifier = (Value-10)/2 + MasteryBonus; } else { return modifier = (Value-10)/2; } }

set

{

if (this.modifier != value)

{

this.modifier = value;

OnPropertyChanged(nameof(Modifier));

}

}

}

public int Value

{

get { return value; }

set

{

if (this.value != value)

{

this.value = value;

OnPropertyChanged(nameof(Value));

OnPropertyChanged(nameof(Modifier)); // При изменении значения обновляем и модификатор

}

}

}

public Stat Stat

{

get { return stat; }

set

{

stat = value;

OnPropertyChanged(nameof(Stat));

// Уведомляем об изменении MasteryBonus при изменении связанной Stat

OnPropertyChanged(nameof(MasteryBonus));

}

}

public Dice Dice

{

get { return dice; }

set

{

dice = value;

OnPropertyChanged(nameof(Dice));

OnPropertyChanged(nameof(Dice.Count));

}

}

public int MasteryBonus

{

get { return Stat?.MasteryBonus ?? 0; }

}

public Skill(string name, Stat stat)

{

Name = name;

Stat = stat;

Dice = new Dice() { Count = 1, Type = 20 };

}

public string Cast()

{

int cast = Dice.Cast();

string result = $"{Name}: Результат броска {cast + Modifier}: {cast} + {Modifier}";

return result;

}

}

}

## Приложение К: Stat. cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Controls.Primitives;

using System.Windows.Controls;

using System.Windows;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.TextFormatting;

using System.Windows.Threading;

using System.Xml.Linq;

using DndHelper.ViewModel;

namespace DndHelper.Model.Stats

{

public class Stat : PropPropertyChanged

{

private string name;

private int value;

private int modifier;

private Skill savingThrows;

private Dice dice;

private Template character;

private RelayCommand statCastCommand;

public RelayCommand StatCastCommand

{

get

{

return statCastCommand ?? (statCastCommand = new RelayCommand(obj =>

{

// Создание объекта Popup

Popup messagePopup = new Popup();

messagePopup.StaysOpen = false;

// Текст сообщения

messagePopup.PlacementTarget = Application.Current.MainWindow;

// Создание объекта Border для стилизации внешнего вида Popup

Border popupBorder = new Border();

popupBorder.Background = Brushes.White; // Цвет фона

popupBorder.BorderBrush = Brushes.Black; // Цвет границы

popupBorder.BorderThickness = new Thickness(1); // Толщина границы

// Текст сообщения

TextBlock messageText = new TextBlock();

messageText.Text = Cast();

ApplicationViewModel.Log.Add($"{Character.Name}: {messageText.Text}");

messageText.Padding = new Thickness(5, 10, 5, 10);

// Добавление элементов в Border

popupBorder.Child = messageText;

// Установка Border в качестве содержимого Popup

messagePopup.Child = popupBorder;

messagePopup.Placement = PlacementMode.Custom;

// Установка пользовательского расположения

messagePopup.CustomPopupPlacementCallback = (popupSize, targetSize, offset) =>

{

// Расположение в правом нижнем углу

double x = targetSize.Width - popupSize.Width \* 1.5;

double y = targetSize.Height - popupSize.Height \* 3;

return new[] { new CustomPopupPlacement(new Point(x, y), PopupPrimaryAxis.None) };

};

messagePopup.IsOpen = true;

// Создайте таймер

DispatcherTimer timer = new DispatcherTimer();

// Установите интервал таймера в 10 секунд

timer.Interval = TimeSpan.FromSeconds(3);

// Обработчик события таймера

timer.Tick += (sender, e) =>

{

// Закрыть Popup после истечения времени

messagePopup.IsOpen = false;

// Остановить таймер

timer.Stop();

};

// Запустите таймер

timer.Start();

}));

}

}

public string Name

{

get => name;

set

{

if (name != value)

{

name = value;

OnPropertyChanged(nameof(Name));

}

}

}

public int Value

{

get => value;

set

{

if (this.value != value)

{

this.value = value;

OnPropertyChanged(nameof(Value));

OnPropertyChanged(nameof(Modifier)); // При изменении значения обновляем и модификатор

}

}

}

public int MasteryBonus

{

get { return Character.MasteryBonus; }

}

public int Modifier

{

get { return modifier = (Value - 10) / 2; }

}

public Skill SavingThrows

{

get => savingThrows;

set

{

if (savingThrows != value)

{

savingThrows = value;

OnPropertyChanged(nameof(SavingThrows));

}

}

}

public Dice Dice

{

get { return dice; }

set

{

dice = value;

OnPropertyChanged(nameof(Dice));

OnPropertyChanged(nameof(Dice.Count));

}

}

public Template Character

{

get { return character; }

set

{

character = value;

OnPropertyChanged(nameof(Character));

// Уведомляем об изменении MasteryBonus при изменении связанного Character

OnPropertyChanged(nameof(MasteryBonus));

}

}

public Stat()

{

Dice = new Dice() { Count = 1, Type = 20 };

}

public string Cast()

{

int cast = Dice.Cast();

string result = $"{Name}: Результат броска {cast + Modifier}: {cast} + {Modifier}";

return result;

}

}

}

Приложение Л: Strength. cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DndHelper.Model.Stats

{

public class Strength : Stat

{

private Skill athletics;

private int value;

private int modifier;

public Skill Athletics

{

get { return athletics; }

set

{

athletics = value;

OnPropertyChanged(nameof(Athletics));

}

}

public new int Value

{

get => value;

set

{

if (this.value != value)

{

this.value = value;

Athletics.Value = value;

SavingThrows.Value = value;

OnPropertyChanged(nameof(Value));

OnPropertyChanged(nameof(Modifier)); // При изменении значения обновляем и модификатор

}

}

}

public int Modifier

{

get { return modifier = (Value - 10) / 2; }

}

public Strength()

{

Name = "Strength";

SavingThrows = new("Strength", this);

Athletics = new("Athletics", this);

}

}

}

## Приложение М: Dexterity. cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Media.Media3D;

namespace DndHelper.Model.Stats

{

public class Dexterity : Stat

{

private Skill acrobatics;

private Skill sleightOfHand;

private Skill stealth;

private int value;

private int modifier;

public Skill Acrobatics

{

get { return acrobatics; }

set

{

acrobatics = value;

OnPropertyChanged(nameof(Acrobatics));

}

}

public Skill SleightOfHand

{

get { return sleightOfHand; }

set

{

sleightOfHand = value;

OnPropertyChanged(nameof(SleightOfHand));

}

}

public Skill Stealth

{

get { return stealth; }

set

{

stealth = value;

OnPropertyChanged(nameof(Stealth));

}

}

public new int Value

{

get => value;

set

{

if (this.value != value)

{

this.value = value;

Acrobatics.Value = value;

SleightOfHand.Value = value;

Stealth.Value = value;

SavingThrows.Value = value;

OnPropertyChanged(nameof(Value));

OnPropertyChanged(nameof(Modifier)); // При изменении значения обновляем и модификатор

}

}

}

public int Modifier

{

get { return modifier = (Value - 10) / 2; }

}

public Dexterity()

{

Name = "Dexterity";

SavingThrows = new("Dexterity", this);

Acrobatics = new("Acrobatics", this);

SleightOfHand = new("Sleight of Hand", this);

Stealth = new("Stealth", this);

}

}

}

## Приложение Н: Constitution. cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DndHelper.Model.Stats

{

public class Constitution : Stat

{

private int value;

private int modifier;

public new int Value

{

get => value;

set

{

if (this.value != value)

{

this.value = value;

SavingThrows.Value = value;

OnPropertyChanged(nameof(Value));

OnPropertyChanged(nameof(Modifier)); // При изменении значения обновляем и модификатор

}

}

}

public int Modifier

{

get { return modifier = (Value - 10) / 2; }

}

public Constitution()

{

Name = "Constitution";

SavingThrows = new("Constitution", this);

}

}

}

## Приложение О: Intelligence. cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using DndHelper.Model.Stats;

namespace DndHelper.Model.Stats

{

public class Intelligence : Stat

{

private Skill arcana;

private Skill history;

private Skill investigation;

private Skill nature;

private Skill religion;

private int value;

private int modifier;

public Skill Arcana

{

get { return arcana; }

set

{

arcana = value;

OnPropertyChanged(nameof(Arcana));

}

}

public Skill History

{

get { return history; }

set

{

history = value;

OnPropertyChanged(nameof(History));

}

}

public Skill Investigation

{

get { return investigation; }

set

{

investigation = value;

OnPropertyChanged(nameof(Investigation));

}

}

public Skill Nature

{

get { return nature; }

set

{

nature = value;

OnPropertyChanged(nameof(Nature));

}

}

public Skill Religion

{

get { return religion; }

set

{

religion = value;

OnPropertyChanged(nameof(Religion));

}

}

public new int Value

{

get => value;

set

{

if (this.value != value)

{

this.value = value;

SavingThrows.Value = value;

Arcana.Value = value;

History.Value = value;

Investigation.Value = value;

Nature.Value = value;

Religion.Value = value;

OnPropertyChanged(nameof(Value));

OnPropertyChanged(nameof(Modifier)); // При изменении значения обновляем и модификатор

}

}

}

public int Modifier

{

get { return modifier = (Value - 10) / 2; }

}

public Intelligence()

{

Name = "Intelligence";

SavingThrows = new("Intelligence", this);

Arcana = new("Arcana", this);

History = new("History", this);

Investigation = new("Investigation", this);

Nature = new("Nature", this);

Religion = new("Religion", this);

}

}

}

## Приложение П: Wisdom. cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DndHelper.Model.Stats

{

public class Wisdom : Stat

{

private Skill animalHandling;

private Skill insight;

private Skill medicine;

private Skill perception;

private Skill survival;

private int value;

private int modifier;

public Skill AnimalHandling

{

get { return animalHandling; }

set

{

animalHandling = value;

OnPropertyChanged(nameof(AnimalHandling));

}

}

public Skill Insight

{

get { return insight; }

set

{

insight = value;

OnPropertyChanged(nameof(Insight));

}

}

public Skill Medicine

{

get { return medicine; }

set

{

medicine = value;

OnPropertyChanged(nameof(Medicine));

}

}

public Skill Perception

{

get { return perception; }

set

{

perception = value;

OnPropertyChanged(nameof(Perception));

}

}

public Skill Survival

{

get { return survival; }

set

{

survival = value;

OnPropertyChanged(nameof(Survival));

}

}

public new int Value

{

get => value;

set

{

if (this.value != value)

{

this.value = value;

SavingThrows.Value = value;

AnimalHandling.Value = value;

Insight.Value = value;

Medicine.Value = value;

Perception.Value = value;

Survival.Value = value;

OnPropertyChanged(nameof(Value));

OnPropertyChanged(nameof(Modifier)); // При изменении значения обновляем и модификатор

}

}

}

public int Modifier

{

get { return modifier = (Value - 10) / 2; }

}

public Wisdom()

{

Name = "Wisdom";

SavingThrows = new("Wisdom", this);

AnimalHandling = new("Animal Handling", this);

Insight = new("Insight", this);

Medicine = new("Medicine", this);

Perception = new("Perception", this);

Survival = new("Survival", this);

}

}

}

## Приложение Р: Charisma. cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DndHelper.Model.Stats

{

public class Charisma : Stat

{

private Skill deception;

private Skill intimidation;

private Skill performance;

private Skill persuasion;

private int value;

private int modifier;

public Skill Deception

{

get { return deception; }

set

{

deception = value;

OnPropertyChanged(nameof(Deception));

}

}

public Skill Intimidation

{

get { return intimidation; }

set

{

intimidation = value;

OnPropertyChanged(nameof(Intimidation));

}

}

public Skill Performance

{

get { return performance; }

set

{

performance = value;

OnPropertyChanged(nameof(Performance));

}

}

public Skill Persuasion

{

get { return persuasion; }

set

{

persuasion = value;

OnPropertyChanged(nameof(Persuasion));

}

}

public new int Value

{

get => value;

set

{

if (this.value != value)

{

this.value = value;

SavingThrows.Value = value;

Deception.Value = value;

Intimidation.Value = value;

Performance.Value = value;

Persuasion.Value = value;

OnPropertyChanged(nameof(Value));

OnPropertyChanged(nameof(Modifier)); // При изменении значения обновляем и модификатор

}

}

}

public int Modifier

{

get { return modifier = (Value - 10) / 2; }

}

public Charisma()

{

Name = "Charisma";

SavingThrows = new("Charisma", this);

Deception = new("Deception", this);

Intimidation = new("Intimidation", this);

Performance = new("Performance", this);

Persuasion = new("Persuasion", this);

}

}

}

## Приложение С: Elf. cs

using DndHelper.Model.Stats;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DndHelper.Model.Races

{

public class Elf : Race

{

public Strength Strength

{

get { return Character.Strength; }

}

public Dexterity Dexterity

{

get { return Character.Dexterity; }

}

public Constitution Constitution

{

get { return Character.Constitution; }

}

public Intelligence Intelligence

{

get { return Character.Intelligence; }

}

public Wisdom Wisdom

{

get { return Character.Wisdom; }

}

public Charisma Charisma

{

get { return Character.Charisma; }

}

public Elf()

{

Name = "Elf";

StrengthBonus = 0;

DexterityBonus = 2;

ConstitutionBonus = 0;

IntelligenceBonus = 0;

WisdomBonus = 0;

CharismaBonus = 0;

Speed = 30;

}

}

}

## Приложение Т: Goblin. cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DndHelper.Model.Races

{

public class Goblin : Race

{

public Goblin()

{

Name = "Goblin";

StrengthBonus = 0;

DexterityBonus = 2;

ConstitutionBonus = 1;

IntelligenceBonus = 0;

WisdomBonus = 0;

CharismaBonus = 0;

Speed = 30;

}

}

}

## Приложение У: HalfOrc. cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Media.TextFormatting;

namespace DndHelper.Model.Races

{

public class HalfOrc : Race

{

public HalfOrc()

{

Name = "Half-Orc";

StrengthBonus = 2;

DexterityBonus = 0;

ConstitutionBonus = 1;

IntelligenceBonus = 0;

WisdomBonus = 0;

CharismaBonus = 0;

Speed = 30;

}

}

}

## Приложение Ф: HighElf

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DndHelper.Model.Races

{

public class HighElf : Elf

{

public HighElf() : base()

{

Name = "High Elf";

IntelligenceBonus = 1;

}

}

}

## Приложение Х: WoodElf

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Media.Effects;

namespace DndHelper.Model.Races

{

public class WoodElf : Elf

{

public WoodElf() : base()

{

Name = "Wood Elf";

WisdomBonus = 1;

Speed = 35;

}

}

}