ДОКУМЕНТАЦИЯ

по заданию «Крестики–нолики»

Разработано: Куваевым Никитой Владиславовичем

ОГЛАВЛЕНИЕ

ПОСТАНОВКА ЗАДАЧИ

Написать программу для игры Крестики-нолики.

Требования к программе:

1. Язык реализации для frontend: (python + QT) или (WEB + python + js) или C#.

2. Язык для реализации backend: python, mssql.

3. Играют зарегистрированные игроки. Поэтому должна быть регистрация нового игрока, корректировка логина или пароля, удаление учетной записи(не физическое, а только логическое).

4. По каждому игроку ведется кол-во проведенных поединков и их исходы.

5. В окне программы есть:

– классическое рабочее игровое поле;

– присутствует кнопка НАЧАТЬ ИГРУ;

– поле индикакции текущего игрока;

– счетчики для игрока и компьютера проведенных боев и их результатов в разрезе кол-ва побед, проигрышей и ничьих;

– выпадающий список выбора уровня игры противника(бота) типа НОВИЧОК, ЗАЩИТА, НАПАДЕНИЕ, ГУРУ, ИИ;

– радио-клавиша о возможности получения от комьютера подсказка на очередной ход игрока;

– таблица рейтингов 5-ти успешых игроков и текущий рейтинг текущего игрока, для расчета рейтинга принимается результат поединка как: 0-проигрыш, 1-ничья, 2-победа .

6. Бот играет ноликами, игрок - крестиками, во время игры выпадающий элемент списка уровня игры и клавиша подсказки блокируются. Первую игру начинает рандомно бот или игрок, остальные - победитель предыдущего поединка

7. После окончания игры высвечивается ее результат и обновляются счетчики и рейтинги.

8. Цвет проставляемых ноликов или крестиков пусть будет синим, а цвет при подсказке серым на белом фоне игрового поля.

9. Для уровня игры с ИИ продумать алгоритм обучения компьютера на основе успешных для бота или игрока атак

10. В режиме игры с подсказками, бот высвечавает подсказку для игрока в нужной клетке игрового поля, а игрок будет решать сам, воспользоваться этой подсказкой или нет.

11. Удаление игрока из базы возможно только если он не регистрировался в игре более 2-х месяцев назад.12.Исходники разработанной программы можно прислать по почте. А в дальнейшем иметь при себе работающий экземпляр программы (можно на ноуте с развернутой средой разработки и mssql сервером)

ЭТАПЫ ПРОЕКТИРОВАНИЯ

Этап 1: Проектирование и архитектура системы

Определение архитектуры приложения:

– Frontend: WPF (Windows Presentation Foundation) на C#.

– Backend: C#, MSSQL.

Применение принципов SOLID и паттернов проектирования:

– Использование MVVM (Model-View-ViewModel) для разделения представления, логики и данных.

– Применение паттернов Repository для доступа к базе данных, Dependency Injection для управления зависимостями.

– Применение Gitflow Workflow для организации и структуризации процесса разработки.

Этап 2: База данных и модели

Создание схемы базы данных в MSSQL:

– Таблица Players: хранение информации об игроках (логическое удаление с помощью поля IsDeleted).

– Таблица Games: хранение информации о проведенных играх, результатах, датах и времени.

Реализация моделей и репозиториев:

– Класс Player: представляет игрока.

– Класс Game: представляет игру.

Этап 3: Регистрация и управление учетными записями

Создание функциональности для регистрации нового игрока:

– Валидация данных.

– Добавление игрока в базу данных.

Функциональность корректировки логина или пароля:

– Возможность изменения логина и пароля.

Логическое удаление учетной записи:

– Обновление поля IsDeleted для учетной записи.

Этап 4: Логика игры и интерфейс

Создание игрового поля 3x3 в WPF.

Реализация логики игры:

– Определение текущего игрока.

– Обновление игрового поля.

– Проверка победителя.

Разработка UI:

– Кнопка "НАЧАТЬ ИГРУ".

– Поле индикации текущего игрока.

– Счетчики для игрока и компьютера (проведенные бои, победы, проигрыши, ничьи).

– Выпадающий список выбора уровня игры противника (бота).

– Радио-кнопка для включения подсказок от компьютера.

– Таблица рейтингов 5-ти успешных игроков и текущий рейтинг игрока.

Этап 5: Реализация бота и уровней сложности

– Простой бот для начального уровня "НОВИЧОК": Случайный выбор ходов.

– Уровни сложности "ЗАЩИТА", "НАПАДЕНИЕ", "ГУРУ": Реализация стратегий для каждого уровня.

– Реализация ИИ: Алгоритмы обучения на основе успешных атак. В данном случае, Minimax

Этап 6: Дополнительные функции

Реализация подсказок для игрока:

– Логика для генерации подсказок ботом.

– Отображение подсказок в сером цвете.

Обновление счетчиков и рейтингов после окончания игры:

– Обновление данных в базе.

– Отображение результатов на UI.

Этап 7: Тестирование и оптимизация

Тестирование всех функций:

– Регистрация и управление учетными записями.

– Логика игры и взаимодействие с ботом.

– Подсказки и уровни сложности.

Оптимизация производительности и исправление багов.

Этап 8: Развертывание и поддержка

Развертывание приложения.

Мониторинг и поддержка:

– Обновление приложения.

– Внесение изменений на основе обратной связи от пользователей.

КОНКРЕТИЗАЦИЯ ЭТАПОВ

Этап 1: Проектирование и архитектура системы

1.1. Определение архитектуры приложения

Frontend:

– Используем WPF (Windows Presentation Foundation) на C#.

– Применяем паттерн MVVM (Model-View-ViewModel) для разделения логики представления и данных.

Backend:

– Используем C# и MSSQL для базы данных.

– Применяем принцип разделения ответственности (SOLID) и паттерны проектирования для поддержания чистоты и гибкости кода.

1.2. Применение принципов SOLID и паттернов проектирования

Принципы SOLID.

1. Single Responsibility Principle (SRP):

Каждый класс и метод должны выполнять только одну задачу. Например, класс PlayerService будет ответственен только за операции с игроками (регистрация, логин и т.д.).

2. Open/Closed Principle (OCP):

Программные сущности должны быть открыты для расширения, но закрыты для модификации. Например, создание интерфейсов для ботов разного уровня сложности, чтобы можно было легко добавлять новые уровни.

3. Liskov Substitution Principle (LSP):

Объекты должны быть заменяемыми на экземпляры их подтипов без изменения правильности программы. Например, любой бот должен

реализовывать интерфейс IBot, и можно заменить одного бота на другого без изменения логики игры.

4. Interface Segregation Principle (ISP):

Клиенты не должны зависеть от интерфейсов, которые они не используют. Например, создание отдельных интерфейсов для различных сервисов (IPlayerService, IGameService).

5. Dependency Inversion Principle (DIP):

– Модули высокого уровня не должны зависеть от модулей низкого уровня. Оба должны зависеть от абстракций.

– Использование Dependency Injection для внедрения зависимостей.

– Паттерны проектирования:

Repository Pattern:

– Обеспечивает абстракцию доступа к данным и изолирует логику доступа к данным от бизнес-логики.

– Создаем репозитории для игроков и игр (PlayerRepository, GameRepository).

Dependency Injection:

Внедрение зависимостей через конструкторы для повышения тестируемости и гибкости кода.

Strategy Pattern:

Позволяет выбрать алгоритм поведения во время выполнения.

Реализация различных стратегий для уровней сложности бота.

Этап 2: База данных и модели

Создание схемы базы данных в MSSQL:

– Создаем таблицу Players для хранения информации об игроках, включая логическое удаление с помощью поля IsDeleted.

– Создаем таблицу Games для хранения информации о проведенных играх, их результатах, датах и времени.

Реализация моделей и репозиториев:

– Создаем класс Player, представляющий модель игрока, с соответствующими полями (Id, Username, Password, Wins, GamesPlayed и т.д.).

– Создаем класс Game, представляющий модель игры, с полями для хранения информации о результатах игры и дате/времени проведения.

– Реализуем репозитории (PlayerRepository, GameRepository) для доступа к данным, используя паттерн Repository. Эти репозитории предоставляют методы для работы с данными (добавление, обновление, удаление, получение и т.д.) и взаимодействуют с базой данных через ORM или непосредственно через SQL запросы.

Этап 3: Регистрация и управление учетными записями

Создание функциональности для регистрации нового игрока:

– На UI добавляем форму для регистрации, где пользователь вводит свой логин и пароль.

– В PlayerService добавляем метод RegisterPlayer, который будет принимать логин и пароль, проводить их валидацию и добавлять нового игрока в базу данных через PlayerRepository.

Функциональность корректировки логина или пароля:

– На UI добавляем возможность изменения логина и пароля для зарегистрированных пользователей.

– В PlayerService добавляем метод UpdateProfile, который принимает id игрока, новый логин и/или пароль, и обновляет соответствующие данные в базе данных через PlayerRepository.

Логическое удаление учетной записи:

– Добавляем кнопку или функцию для удаления учетной записи на UI.

– В PlayerService добавляем метод LogicalDeletePlayer, который помечает учетную запись как удаленную, обновляя поле IsDeleted в базе данных через PlayerRepository.

Этап 4: Логика игры и интерфейс

Создание игрового поля 3x3 в WPF:

– В XAML добавляем элементы для отображения игрового поля, например, Grid с кнопками, представляющими клетки.

– Каждая кнопка будет иметь обработчик события для выполнения хода игрока при нажатии.

Реализация логики игры:

– В GameService добавляем методы для определения текущего игрока, обновления игрового поля и проверки победителя.

– Логика определения хода бота должна быть реализована в соответствии с выбранным уровнем сложности.

Разработка UI:

– Добавляем кнопку "НАЧАТЬ ИГРУ", которая будет инициировать новую игру.

– Выводим информацию о текущем игроке и его счете (проведенные бои, победы, проигрыши, ничьи).

– Добавляем выпадающий список для выбора уровня игры противника (бота) и радио-кнопку для включения подсказок от компьютера.

– Создаем таблицу рейтингов, отображающую 5 лучших игроков и текущий рейтинг игрока.

Этап 5: Реализация бота и уровней сложности

Создание простого бота для уровня "НОВИЧОК":

Реализуем метод в GameService для случайного выбора ходов бота на пустые клетки.

Реализация уровней сложности "ЗАЩИТА", "НАПАДЕНИЕ", "ГУРУ":

Для каждого уровня сложности разрабатываем стратегии:

– "ЗАЩИТА": бот будет предпочитать блокировать ходы игрока, чтобы не дать ему выиграть.

– "НАПАДЕНИЕ": бот будет стараться выиграть самому, формируя выигрышные комбинации.

– "ГУРУ": бот будет использовать оптимальные стратегии, учитывая текущее состояние игрового поля.

Реализация ИИ:

Разработаем алгоритм обучения бота на основе успешных атак и защиты – minimax.

Этап 6: Дополнительные функции

Реализация подсказок для игрока:

– Разработка логики для генерации подсказок ботом.

– Отображение подсказок на игровом поле в сером цвете.

Обновление счетчиков и рейтингов после окончания игры:

– Обновление данных в базе данных о количестве проведенных боев, побед, проигрышей и ничьих.

– Отображение результатов на пользовательском интерфейсе, чтобы игрок мог видеть свой прогресс и рейтинг.

Этап 7: Тестирование и оптимизация

Тестирование всех функций:

– Проведение модульного тестирования для каждой функции и метода, чтобы проверить их правильность и корректность работы.

– Тестирование интеграции между различными компонентами приложения для обеспечения их взаимодействия без ошибок.

– Проверка на соответствие требованиям и спецификации, выявление и устранение ошибок.

Оптимизация производительности и исправление багов:

– Анализ производительности приложения для выявления узких мест и неэффективных участков кода.

– Оптимизация алгоритмов и структур данных для улучшения скорости работы приложения.

– Исправление обнаруженных ошибок и уязвимостей, обновление программного кода для повышения стабильности и надежности приложения.

Этап 8: Развертывание и поддержка

Развертывание приложения:

– Подготовка приложения к развертыванию на целевой платформе (например, упаковка в установщик или создание дистрибутива).

– Установка и настройка приложения на сервере или компьютере пользователя.

– Проведение начальной конфигурации и настройки, включая подключение к базе данных и другие настройки.

Мониторинг и поддержка:

– Организация мониторинга работы приложения для отслеживания его производительности, доступности и стабильности.

– Поддержка пользователей и ответ на их запросы и обратную связь.

– Регулярное обновление приложения для внесения улучшений, исправления ошибок и добавления новых функций на основе обратной связи от пользователей.

– Проведение регулярных резервных копий данных для обеспечения безопасности и сохранности информации.

ВЫВОД

В ходе проектирования и разработки приложения для игры в крестики-нолики были выполнены следующие этапы:

Проектирование и архитектура системы:

– Определена архитектура приложения, использующая WPF для фронтенда и C# с использованием MSSQL для бэкенда.

– Применены принципы SOLID и паттерны проектирования, такие как MVVM, Repository и Dependency Injection.

База данных и модели:

– Создана схема базы данных с таблицами для хранения информации об игроках и проведенных играх.

– Реализованы модели Player и Game, а также репозитории для доступа к данным.

Регистрация и управление учетными записями:

Реализована функциональность регистрации новых игроков, изменения логина или пароля, а также логического удаления учетных записей.

Логика игры и интерфейс:

– Создано игровое поле 3x3 в WPF.

– Реализована логика игры, включая определение текущего игрока, обновление поля и проверку на победу.

– Разработан пользовательский интерфейс с кнопкой начала игры, индикаторами текущего игрока, счетчиками и выбором уровня сложности бота.

Реализация бота и уровней сложности:

Созданы различные уровни сложности бота с соответствующими стратегиями игры.

Дополнительные функции:

Реализованы подсказки для игрока и обновление счетчиков и рейтингов после игры.

Тестирование и оптимизация:

Проведено тестирование всех функций приложения, оптимизирована производительность и исправлены обнаруженные баги.

Развертывание и поддержка:

Приложение готово к развертыванию на целевой платформе, обеспечивается мониторинг его работы и поддержка пользователей, а также проводится регулярное обновление с целью улучшения и добавления новых функций.

Этапы разработки включают в себя как создание базового функционала, так и его дальнейшее совершенствование, чтобы обеспечить удобство использования и качество работы приложения.

ПЕРЕЧЕНЬ ССЫЛОК

1. Алгоритм Minimax: Simple Explanation of the Minimax Algorithm with Tic-Tac-Toe - YouTube

2. Шаблоны проектирования: Паттерны/шаблоны проектирования (refactoring.guru)

3. Принципы SOLID: Принципы SOLID, о которых должен знать каждый разработчик | by Nikita | WebbDEV | Medium

4. Dependency Injection: Внедрение зависимостей - .NET | Microsoft Learn

5. Реализация: Kuvaev-dev/TicTacToe: Tic-Tac-Toe Game using WPF C#. (github.com)