



**WYŻSZA SZKOŁA  
INFORMATYKI i ZARZĄDZANIA**  
z siedzibą w Rzeszowie

## **Dokumentacja projektu**

Przedmiot: **Programowanie**

Tytuł projektu: Aplikacja rozwiązująca zagadkę  
Sudoku.

Prowadzący:

*Dr Marek Jaszuk*

Wykonawca:

*Krzysztof Bigos W60156*

*Konrad Haduch W61513*

*Dawid Hamerla W60174*

Semestr i symbol kierunku: 4IIZ

Grupa: GP01, GP02

Grupa na platformie BB:

GP01/15

Rzeszów, 2020

1.	Opis założeń projektu(wymagania biznesowe).....	3
2.	Specyfikacja wymagań .....	4
2.1	Wymagania funkcjonalne.....	4
2.2	Wymagania нефункционалне .....	4
3.	Diagram przypadków użycia .....	5
4.	Harmonogram prac nad projektem .....	6
5.	Opis techniczny projektu .....	8
6.	Prezentacja warstwy użytkowej projektu.....	9
7.	Raport z testów jednostkowych .....	10
8.	Materiały źródłowe .....	11

## **1. Opis założeń projektu(wymagania biznesowe)**

Aplikacja w założeniu ma pomagać w rozwiązywaniu zagadki logicznej Sudoku. Użytkownik otrzyma możliwość skorzystania z pomocy przy rozwiązywaniu łamigłówki, po uprzednim wprowadzeniu przez niego stanu faktycznego, aplikacja sprawdzi czy wprowadzone dane są poprawne a następnie, jeśli nie napotka błędu w otrzymanych danych rozwiąże problem. Wyżej wymieniony program, może być wykorzystany do uczenia się z jego pomocą rozwiązywania Sudoku o różnej skali trudności. Może być również wykorzystany w miejscach gdzie, wymagana jest szybka analiza wyników, np. różnego rodzaju turnieje, w których sędziowie przygotowując zagadki, potrzebują szybko poprawnych wyników, by następnie móc porównać je z wynikami uzyskanymi przez zawodników. Za pomocą aplikacji można też testować czy wymyślony przez nas układ będzie w ogóle możliwy do rozwiązania.

## **2. Specyfikacja wymagań**

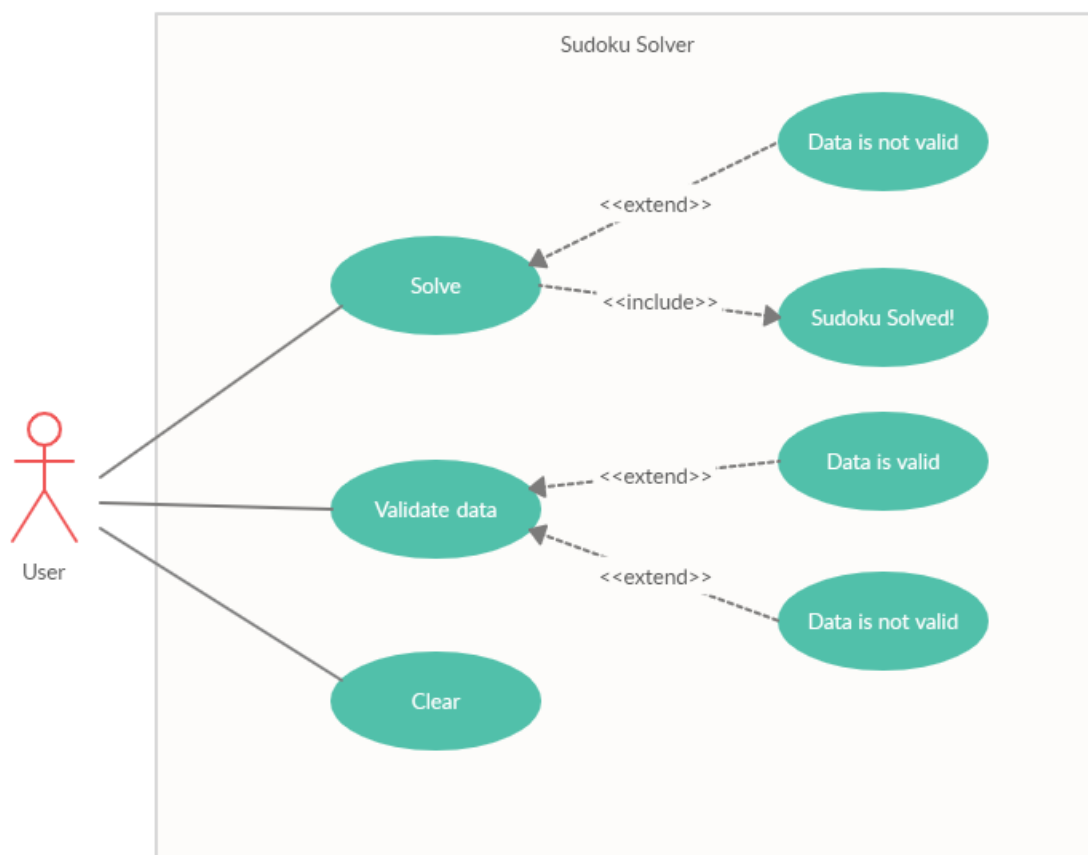
### **2.1 Wymagania funkcjonalne**

- 2.1.1 Aplikacja posiada graficzny interfejs pozwalający zawierający planszę do gry.
- 2.1.2 Plansza podzielona jest na dziewięć kwadratów, które z kolei zawierają po dziewięć kwadratów w których umieszczane są cyfry.
- 2.1.3 W prawym górnym rogu znajduje się przycisk „Validate data” służący do sprawdzenia poprawności danych wpisanych przez użytkownika.
- 2.1.4 Poniżej przycisku „Validate data” znajduje się przycisk „Solve”. Po jego kliknięciu zagadka powinna zostać rozwiązana.
- 2.1.5 Pod przyciskiem „Solve” znajduje się pole tekstowe służące do wyświetlania komunikatów z aplikacji. Jeżeli klikniemy przycisk „Solve” to wyświetli się zielony napis „Sudoku solved!”, Jeżeli klikniemy przycisk „Validate data” to wyświetli się komunikat „Data is valid” jeśli dane są poprawne lub „Data is not valid” jeśli dane są niepoprawne.
- 2.1.6 Na dole ekranu po prawej stronie znajduje się przycisk „Clear” służący do czyszczenia planszy.

### **2.2 Wymagania нефunkcjonalne**

- 2.2.1 Każde pole ma ograniczoną możliwość wprowadzania danych do jednego znaku na pole tekstowe,
- 2.2.2 Każde dane wprowadzane do aplikacji są typu integer
- 2.2.3 Aplikacja działa bez opóźnień
- 2.2.4 Wymiary poszczególnych pól danych powinny mieć wymiar 20px na 20px.

### 3. Diagram przypadków użycia



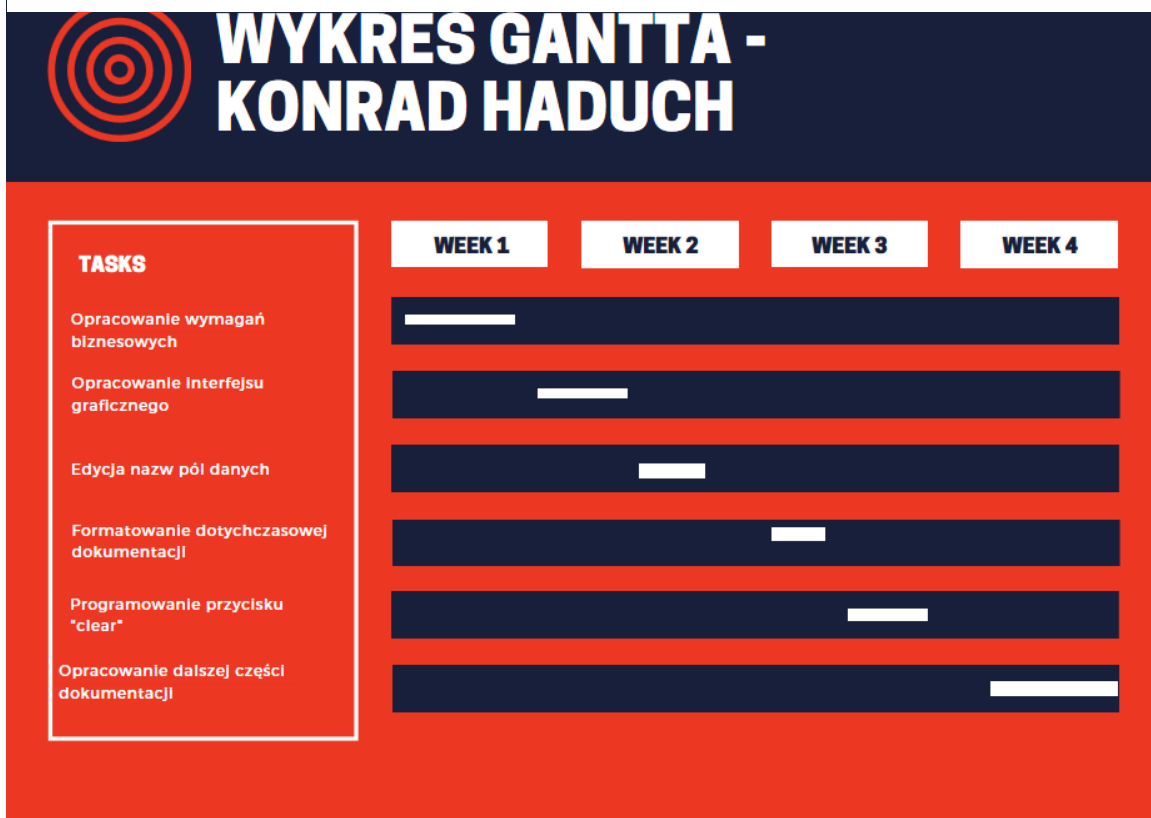
Rys. 1 Diagram przypadków użycia

W powyższym diagramie można zaobserwować przypadki użycia wszystkich założeń funkcjonalnych, których z poziomu aplikacji może dokonać użytkownik. Za pomocą „Solve” rozwiązuje sudoku z wcześniej wprowadzonymi danymi. Przy użyciu „Validate data” sprawdza czy dane przez niego wprowadzone są poprawne. Natomiast za wykorzystaniem „Clear” może wyczyścić wprowadzone własnoręcznie dane lub rozwiązana przez program łamigłówkę.

#### 4. Harmonogram prac nad projektem



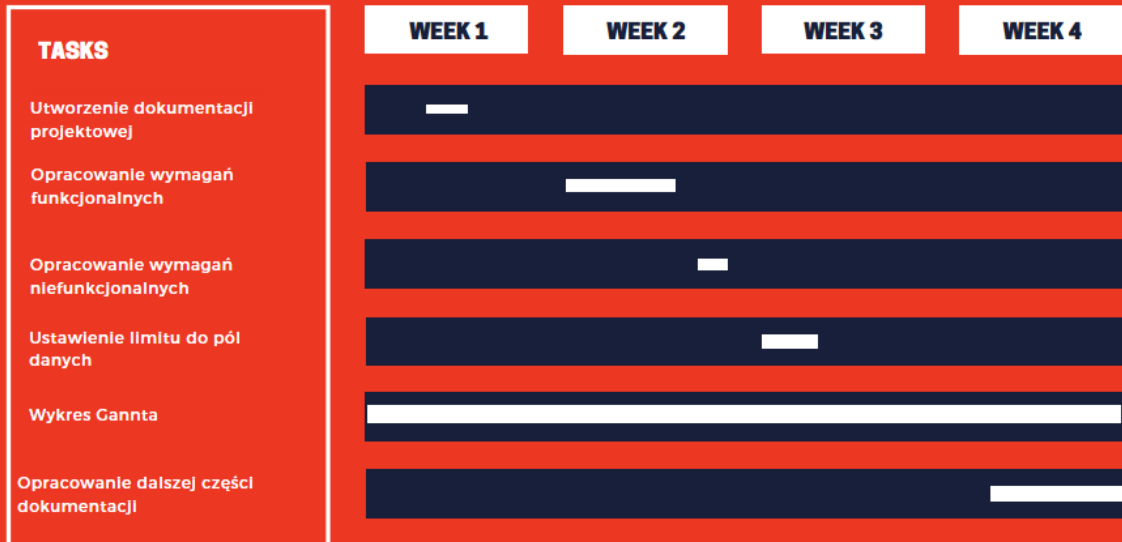
Rys. 2 Wykres Gantt - Krzysztof Bigos



Rys. 3 Wykres Gantt - Konrad Haduch



# WYKRES GANTTA - DAWID HAMERLA



Rys. 4 Wykres Gantta - Dawid Hamerla

## 5. Opis techniczny projektu

Projekt wykonany został przy użyciu następujących aplikacji:

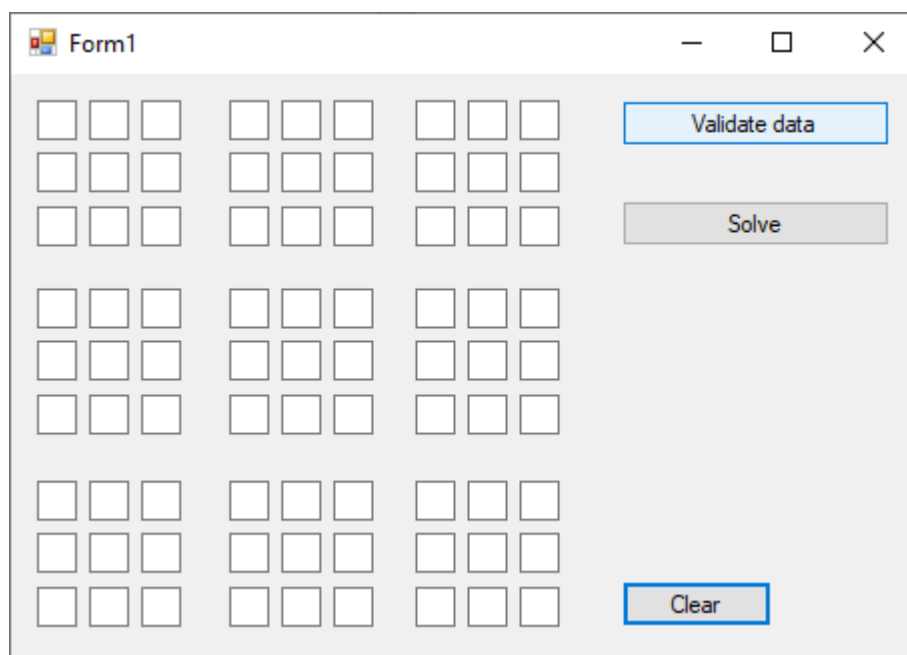
- Visual Studio (w języku C#, używając szablonu aplikacji okienkowej WinForms),
- <https://creately.com/> (diagram przypadków użycia),
- <https://www.canva.com/> (diagram Gantt),
- Doxygen 1.8.18 (komentarze dokumentujące).

Główną klasą jest Form1.cs gdzie znajdują się metody:

- Solve – metoda, która jest wykonywana kiedy użytkownik naciśnie przycisk „Solve”, sprawdza czy wprowadzone dane są poprawne, jeśli tak rozwiązuje zagadkę Sudoku, w przeciwnym przypadku poinformuje o błędnie wprowadzonych danych
- IsAvailable – metoda sprawdzająca czy w dane pole, może zostać wstawiona podana przez użytkownika liczba, pobiera parametry z bazy już wprowadzonych w poszczególne pola danych, następnie dokonuje sprawdzenia czy w zadanej kolumnie, wierszu nie ma już podanej liczby.
- IsValid – metoda, która jest wykonywana po wciśnięciu przez użytkownika przycisku „Validate Data”, sprawdza ona czy wprowadzone przez użytkownika dane są poprawne i zwraca „Data is valid” w przypadku kiedy dane są poprawne lub „Data is not valid” w przypadku kiedy dane nie są poprawne.



## 6. Prezentacja warstwy użytkowej projektu

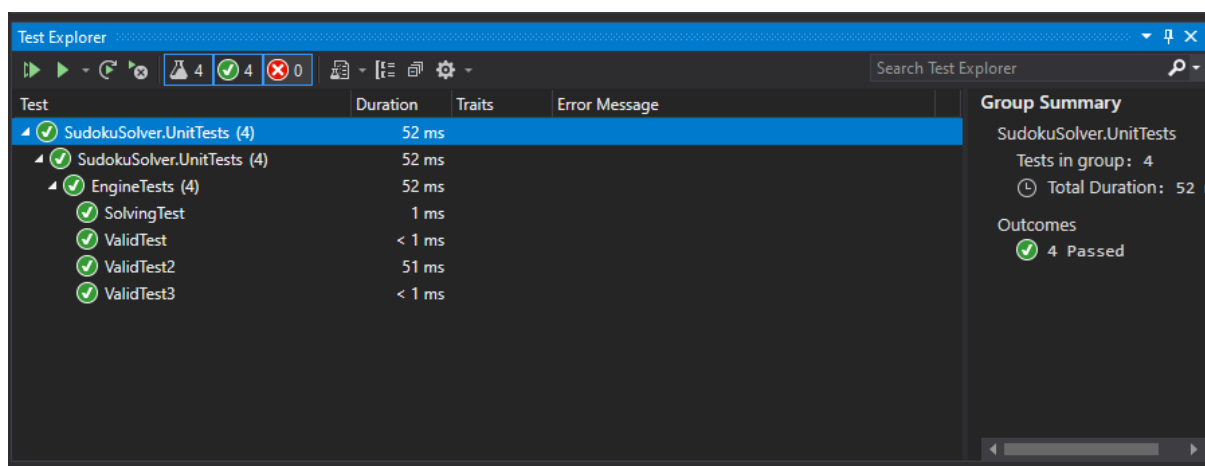


Rys. 5 Sudoku solver

Program w swojej formie, dla prostszej nawigacji przypominać ma wyglądem łamigłówkę Sudoku, pozwoli to dosyć intuicyjnie wprowadzać dane do odpowiednich okienek, jak widać wiodącym motywem w okienku jest „tablica sudoku” gdzie użytkownik wprowadza dane, po stronie lewej są przyciski które, pozwalają wykonywać operacje na wprowadzanych wcześniej danych. Na samej górze znajduje się przycisk „Validate data” został on umieszczony jako pierwszy, ponieważ poprawność danych jest kluczem do poprawnego rozwiązania zagadki. Na samym dole umieszczony został przycisk „Clear” za pomocą którego można wyczyścić uprzednio wprowadzone dane, lub rozwiązane Sudoku i przejść do wprowadzania innych, nowych danych.

## 7. Raport z testów jednostkowych

Testowaniu zostały poddane dwie funkcjonalności. Przetestowano funkcję walidacji danych oraz funkcję rozwiązywania zagadki. Obie funkcje przeszły testy pozytywnie bez poprawiania kodu źródłowego. Funkcję walidacji danych przetestowano na 3 różne sposoby i każdy dał wynik pozytywny. *ValidTest* testując metodę *IsDataValid* zawartą w klasie *Engine*, która jest silnikiem aplikacji. *ValidTest* tworzy tablicę poprawnych danych które następnie implementuje do metody *IsDataValid*.. Test zwrócił poprawny wynik zatem test uznano za pozytywny. Następnie *ValidTest2* posiada jednowymiarową tablicę z niepoprawnymi danymi. Test miał na celu sprawdzenie czy niepoprawne dane zostaną wykryte przez metodę i czy zostanie zwrócona wartość „false”. Po przeprowadzeniu testu wynik spełnił oczekiwania i została zwrócona wartość „false” zatem test drugi również zakończono pozytywnie. *ValidTest3* również posiada niepoprawne dane. Test pokazał zwrócił poprawny wynik zatem *ValidTest3* przeszedł również pozytywnie. Czasy wykonywania się testów przedstawiono na rysunku poniżej. *SolvingTest* zawiera dwa zbiory danych. Pierwszy zawiera dane przygotowane do rozwiązywania zagadki. Test miał na celu sprawdzenie czy metoda uzupełni poprawnie brakujące dane. Wynik był zadowalający i uznano test za pozytywny. Drugi zbiór danych to gotowe rozwiązanie zagadki. Test miał za zadanie sprawdzić czy dane oczekiwane będą się pokrywały z danymi wyświetlonymi przez metodę. Test wykonany dał wynik pozytywny.



Test	Duration	Traits	Error Message
✓ SudokuSolver.UnitTests (4)	52 ms		
✓ SudokuSolver.UnitTests (4)	52 ms		
✓ EngineTests (4)	52 ms		
✓ SolvingTest	1 ms		
✓ ValidTest	< 1 ms		
✓ ValidTest2	51 ms		
✓ ValidTest3	< 1 ms		

**Group Summary**  
SudokuSolver.UnitTests  
Tests in group: 4  
⌚ Total Duration: 52 ms  
**Outcomes**  
✓ 4 Passed

Rys. 6 Testy jednostkowe

## 8. Materiały źródłowe

1. [https://pl.wikipedia.org/wiki/Diagram\\_Gantt](https://pl.wikipedia.org/wiki/Diagram_Gantt)
2. <https://pl.wikipedia.org/wiki/Sudoku>
3. [A Pencil-and-Paper Algorithm for Solving Sudoku Puzzles](#)
4. [Sudoku in C#](#)
5. <https://docs.microsoft.com/pl-pl/dotnet/csharp/tour-of-csharp/>
6. [Use Case Diagram Tutorial](#)