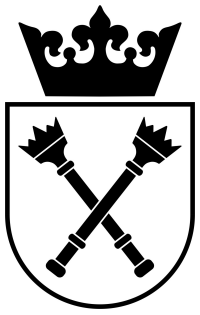
Uniwerystet Jagielloński

Projekt zaliczeniowy

z kursu Inżynieria Oprogramowania

Sudoku Solver

Daniel Dobrowolski
Małgorzata Dymek
Tomasz Janik
Łukasz Kosmaty
Nikodem Kwaśniak
Dawid Szczerba



Semestr letni 2018/2019

Spis treści

1 Dokumentacja

Na dokumentację projektu składają się następujące pliki:

- Sudoku_Solver.pdf
- IEEE830.pdf
- Testowanie.pdf
- Instrukcja_kompilacji.pdf
- Instrukcja_obslugi.pdf
- folder Diagramy_UML zawierający:
  - diagram przypadków użycia,
  - dwa diagramy czynności.
- folder Organizacja_pracy zawierający pierwotne i aktualne wersje:
  - diagramu sieciowego,
  - diagramu WBS,
  - wykresu Gannta.

2 Opis projektu.

2.1 Wstępne założenia projektu.

Projekt ma na celu stworzenie aplikacji desktopowej umożliwiającej klientowi wpisanie, lub wczytanie ze zdjęcia zagadki logicznej typu Sudoku, a następnie otrzymanie jej rozwiązania.

2.2 Wymagania.

[$W1$] **Rozwiązywanie sudoku na planszy 9x9.**

[$W2$] **Możliwość wpisania własnego sudoku.**

- Możliwość wpisania cyfr w ramce 9x9.

[$W3$] **Odczyt sudoku z zdjęcia.**

- Obsługiwany formaty: jpg, png.
- Zdjęcie musi zawierać czarno-białe sudoku.
- Rozpoznawanie cyfr na poziomie poprawności 70%.

[$W4$] **Pół-brutalny algorytm rozwiązywania sudoku.**

- Rozwiązuje sudoku do 7 sekund.
- Daje tylko jedno (pierwsze znalezione) rozwiązanie.

[$W5$] **Interfejs graficzny.**

- System skinów (motywów): skin jasny, skin ciemny.

[$W6$] **Wyświetlanie wyników w interfejsie graficznym.**

[$W7$] **Obróbka zdjęcia po stronie klienta.**

- Możliwość przycięcia zdjęcia przez Użytkownika.

[$W8$] **Parametryzacja obrazów.**

- Zmiana za pomocą sliderów (suwaków).
- Dostępne parametry:
  - Minimalna grubość linii, nazwany “Grubość linii”
  - Minimalny odstęp między ramką a liczbą, nazwany “Odstęp”
  - Poziom progowania, nazwany “Progowanie”
  - Zmiana natężenia rozmycia Gaussa, nazwany “Rozmycie Gaussa”

[$W9$] **Wysyłka obrazu do serwera.**

- Zdjęcie wysłane metodą POST.

- Wysłane odpowiednie parametry.
- Wysłane zdjęcie po obróbce.

[$W10$] **Działający serwer.**

- Możliwość obsłużenia wielu klientów na raz.
- Sprawna odpowiedź - do 15 sekund.
- Uptime na poziomie 90%.

[$W11$] **Odczyt wyniku z serwera.**

- Odczyt z formatu JSON.

[$W12$] **System Errorów.**

- [$E001$] Nie wykryto odpowiedniej wersji JAVY
- [$E002$] Nieznany format pliku (tylko .jpg)
- [$E003$] Niewykryte poprawne sudoku na zdjęciu
- [$E004$] Sudoku nie posiada rozwiązania
- [$E005$] Brak połączenia z siecią
- [$E006$] Serwer nieaktywny
- [$E007$] Plik nie istnieje

2.3 Scenariusze przypadków użycia.

[$PU1$] **Wczytywanie obrazu.**
1. Aplikacja wyświetla formularz główny.
2. Użytkownik rozpoczyna proces wczytywania zdjęcia poprzez kliknięcie przycisku "Wczytaj".
3. Aplikacja wyświetla formularz wyboru (domyślny systemowy).
4. Użytkownik wybiera plik jpg/png (ma możliwość wyboru tylko jednego).
5. Aplikacja wyświetla formularz ze zdjęciem wybranym przez użytkownika (podgląd).
6. Użytkownik zatwierdza wybór zdjęcia poprzez kliknięcie przycisku "Zatwierdź".

7. Aplikacja wysyła zdjęcie metodą POST na serwer w celu przeczytania sudoku.
8. Serwer zwraca tablice przeczytanych cyfr.
9. Aplikacja przechodzi do formularza głównego z wyświetlonym wybranym zdjęciem oraz uzupełnioną siatką.

[*PU*2] alternatywny do PU1: **Wczytanie nieobsługiwanego formatu pliku lub ścieżki do pliku nieistniejącego.**
1-4. jak w PU1.
5.Aplikacja wyświetla błąd E002/E003.
6. Powrót do punktu 2 PU1.

[*PU*3] rozszerzający do PU1: **Edycja zdjęcia.**
1-4. jak w PU1. 5. Użytkownika naciska przycisk "Zaawansowane".
6. Aplikacja wyświetla formularz opcji zaawansowanych.
7. Użytkownik używa suwaków do parametryzacji wybranego wcześniej zdjęcia.
8. Użytkownik kończy edycję poprzez naciśnięcie przycisku "Ok".
9. Powrót do punktu 6 PU1.

[*PU*4] alternatywny do PU3: **Przywrócenie ustawień domyślnych.**
1-3. jak w PU3. 4. Użytkownik przywraca ustawienia domyślne poprzez naciśnięcie przycisku "Ustawienia domyślne".
5. Aplikacja ustawia parametry obrazu na domyślne - powrót suwaków na ustawienie neutralne.
6. Powrót do punktu 4 PU3.

[*PU*5] alternatywny do PU1: **Zmiana wybranego zdjęcia.**
1-5 jak w PU1.
6. Użytkownik klika na wcześniej wczytane zdjęcie.
7. Powrót do punktu 3 PU1.

[*PU*6] alternatywny do PU1: **Anulowanie wczytywania zdjęcia.**

1-3. jak w PU1.
4. Użytkownik anuluje wybór na jeden z dwóch sposobów:

- Użytkownik zamyka systemowy formularz wyboru pliku poprzez naciśnięcie przycisku “Anuluj” lub ikony zamykania (krzyżyk).
- Użytkownik może też zatwierdzić wybór pliku ale zamknąć formularz zatwierdzenia zdjęcia (krzyżyk).

5. Powrót do punktu 1 PU1.

[*PU*7] rozszerzający do PU1: **Przycięcie zdjęcia.**
1-4. jak w PU1. 5. Użytkownik zaznacza obszar na zdjęciu.
6. Użytkownik klika przycisk “Przytnij”.
7. Aplikacja przycina według zaznaczonego obszaru.
8. Aplikacja zwraca error w przypadku braku zaznaczonego obszaru.
9. Użytkownik kończy edycję poprzez naciśnięcie przycisku “Zatwierdź”.
10. Powrót do punktu 6 PU1 (gdzie wyświetla się wersja zdjęcia zedytowana przez użytkownika).

[*PU*8] **Wpisanie Sudoku przez użytkownika.**
1. Aplikacja wyświetla formularz główny.
2. Użytkownik klika w pole (na siatce sudoku), w którym chce dokonać zmian.
3. System podświetla kliknięte pole (czerwona obramówka).
4. Użytkownik za pomocą klawiatury wprowadza poprawną cyfrę (tz. taką która może znajdować się w wybranym miejscu).
5. W wskazanym miejscu pojawia się wprowadzona cyfra.

[*PU*9] alternatywny do PU8: **Wprowadzenie błędnych danych.**
1-3. jak w PU9.
4. Użytkownik podaje niepoprawną cyfrę lub literę.
5. Wskazane miejsce pozostaje bez zmian.

[*PU*10] **Rozwiązanie poprawnego Sudoku przez program.**

Zawiera PU1 lub PU9 (wczytanie sudoku)
1. Użytkownik klika przycisk “Rozwiąż”.
2. Aplikacja wysyła zapytanie POST do serwera tablicą cyfr.
3. Sudoku zostaje rozwiązane po stronie serwera.
4. Aplikacja otrzymuje odpowiedź od serwera.
5. Aplikacja wyświetla uzupełnioną siatkę w formularzu głównym, gdzie na zielono zaznaczone są uzupełnione przez algorytm cyfry.

[$PU11$] alternatywny do PU10 lub do PU1: **Błędne Sudoku.** 1. 1-3 jak w PU9 lub 1-7 jak w PU1.
2. Serwer nie jest w stanie rozwiązać Sudoku (okazuje się błędne).
3. Aplikacja otrzymuje odpowiedź z informacją o błędzie E004.
4. Aplikacja wyświetla błąd.
5. Powrót do punktu 2 PU1.

[$PU12$] alternatywny do PU10 lub do PU1: **Nie rozpoznano Sudoku.**
1. 1-3 jak w PU9 lub 1-7 jak w PU1.
2. Aplikacja na serwerze nie jest w stanie znaleźć Sudoku na zdjęciu.
3. Aplikacja otrzymuje odpowiedź z informacją o błędzie E003.
4. Aplikacja wyświetla błąd.
5. Powrót do punktu 2 PU1.

[$PU13$] alternatywny do PU10 lub do PU1: **Serwer nieaktywny.**
1. 1-3 jak w PU9 lub 1-7 jak w PU1.
2. Brak możliwości połączenia z serwerem - serwer nieaktywny.
3. Aplikacja wyświetla błąd E006.
4. Powrót do punktu 2 PU1.

[$PU14$] alternatywny do PU10 lub do PU1: **Brak połączenia z siecią.**
1. 1-3 jak w PU9 lub 1-7 jak w PU1.
2. Brak możliwości połączenia z serwerem - brak połączenia z siecią.
3. Aplikacja wyświetla błąd E005.

4. Powrót do punktu 2 PU1.

[$PU15$] **Zmiana motywu.**
1. Aplikacja wyświetla formularz główny.
2. Użytkownik klika myszką przycisk “MOTYW”.
3. Użytkownik wybiera z rozwijanej listy motyw.
4. Aplikacja zmienia szatę graficzną.

[$PU16$] **Wyczyszczenie sudoku.**
1. Aplikacja wyświetla formularz główny.
2. Użytkownik klika przycisk “Wyczyść”.
3. Aplikacja czyści siatkę z sudoku.

[$PU17$] alternatywa do PU1: **Plik nie istnieje lub wystąpił problem podczas otwierania.**
1-5. jak w PU1.
6.Aplikacja nie jest wstanie otworzyć wybranego pliku.
7.Aplikacja wyświetla informację o błędzie E007.
8. Powrót do 1 PU1.

[$PU18$] **Uruchomienie aplikacji.**
1. Użytkownik klika ikon aplikacji.
2. Aplikacja sprawdza wersje javy na komputerze użytkownika.
3. Aplikacja wyświetla formularz główny.

[$PU19$] alternatyw do PU18: **Nieobsługiwana wersja JAVY.**
1-2. jak w PU18. 3. Aplikacja zwraca błąd E001.

2.4 Funkcjonalności podstawowe.

2.4.1 Dostępność aplikacji klienckiej.

Użytkownik ma dostęp do aplikacji desktopowej poprawnie działającej w środowisku określonym w dokumencie IEEE830.

2.4.2 Możliwość wczytania obrazu .jpg przez Użytkownika.

Użytkownik ma możliwość wczytania do aplikacji desktopowej obraz w formiacie .jpg.

2.4.3 Możliwość podstawowej edycji zdjęcia.

Użytkownik ma możliwość przycięcia i/lub parametryzacji wczytanego zdjęcia zgodnie z zakresem podanym w Wymaganiach.

2.4.4 Możliwość ręcznego wpisania cyfr Sudoku.

Aplikacja umożliwia Użytkownikowi ręczne wpisanie cyfr na siatkę Sudoku.

2.4.5 Rozpoznanie Sudoku ze zdjęcia przez aplikację serwera.

2.4.6 Uzyskanie rozwiązania Sudoku na działajacym serwerze.

Użytkownik ma możliwość uzyskania rozwiązania wprowadzonego poprawnego Sudoku. Aplikacja wysyła dane na serwer, gdzie Sudoku zostaje, przy założeniu jego poprawności, rozwiązane, a następnie wynik zostaje zwrócony użytkownikowi.

2.5 Funkcjonalności dodatkowe.

2.5.1 Możliwość wczytania obrazu .png przez Użytkownika.

Użytkownik ma możliwość wczytania do aplikacji desktopowej obraz w formiacie .png.

2.5.2 Możliwość rozbudowanej edycji zdjęcia.

Rozszerzenie możliwość parametryzacji zgodnie z opisem w sekcji Implementacja/Parametry.

3 Zastosowane rozwiązania, realizacja wymagań.

3.1 Aplikacja.

Realizacja wymagań [W2], [W5], [W6], [W7], [W9], [W11].

3.1.1 Użyte technologie.

Do stworzenia aplikacji do Sudoku Solvera użyliśmy Java 8 oraz JavaFX 8. Do testowania używaliśmy JUnit w wersji 5.4.2.

Zdecydowaliśmy się na wybór JavaFX w wersji 8 z kilku powodów, głównymi z nich są:

- wspiera CSS, użyteczny do stylizowania obiektów GUI
- wspiera wyrażenia lambda, które wykorzystaliśmy w kodzie, aby był on bardziej czytelny i przejrzysty
- istnieje możliwość łatwego i szybkiego przeniesnia aplikacji na system Android lub iOS, jeśli w przyszłości klient wyraźiłby taką chęć
- jest nowsza i częściej używana niż Swing

Zdecydowaliśmy się na wybór JUnit w tej wersji, ponieważ:

- zapewnia wsteczną kompatybilność z JUnit 4
- wspiera wyrażenia lambda, w przeciwieństwie do swojego poprzednika JUnit 4
- jest powszechnie używany w środowiku informatycznym

3.1.2 Użyte wzorce projektowe.

- **Singleton**
  Potrzebowaliśmy globalnie dostępnej zmiennej informującej nas o tym, czy w danym momencie oczekujemy na odpowiedź z serwara, żeby wiedzieć czy pozwolić użytkownikowi na wysłanie kolejnego zapytania na serwer, dlatego użyliśmy w kodzie wzorca singletona, który pozwolił nam na uzyskanie pożadanego efektu.

- **Observer**
  W aplikacji mamy kilka przypadków, kiedy po zmienie stanu w jednej klasie musimy poinformować o tym inne klasy i dokonać w ich zmian w odpowiedzi na tą pierwszą zmiane, więc wzorzec obserwatora jest tutaj idealny. Obserwator pozwala na przesyłanie informacji między obiektami bez żadnych zmian w klasie obserwatora i klasie ovserwowanej.

3.2 Serwer

Realizacja wymagań [W1], [W3], [W4], [W10].

3.2.1 Użyte technologie.

Do budowania oraz zarządzania projektem korzystamy z maven wraz z kompilatorem javy w wersji 8. Projekt serwera został podzielony na 3 moduły.

Technologie użyte przy ich tworzeniu:

- **Solver** – czysta java 8.
- **RecofnizerLib**
  - OpenCV 4.0.1 – biblioteka do rozpoznawania obrazu oraz do nauczania maszynowego.
  - Google guava 16.0.1 – zbiór pomocniczych funkcji i pojemników.
  - Tess4j 3.5.3 – wrapper do zewnętrznego ocra.
- **Server**
  - Spring MVC 5.1.6 – framework do tworzenia aplikacji webowych w javie.
  - Commons-fileupload 1.4 – dodatek do springa umożliwiający przetwarzanie zapytania http składającego się z wielu części (np. do odebrania zdjęcie).
  - Gson 2.8.5 – bilblioteka do parsowania jsona.

Serwer został przeniesiony w wirtualne środowisko przy pomocy dockera. Dzięki temu możemy w łatwy sposób instalować go oraz przenosić na różne platformy.

3.2.2 Użyte algorytmy.

- **Półbrutalny algorytm rozwiązywania Sudoku.**
  Na początku tworzona jest stuktura danych, która trzyma co można wpisać w daną komórke wraz z iteratorem do aktualnie wybranej wartości (na początku nie wskazuje on na żaden element). Dzięki temu ograniczamy liczbę sprawdzanych kombinacji oraz pozwala szybciej odpowiedzieć na pytanie czy dane sudoku da się rozwiązać. Następnie algorytm działa podobnie do klasycznego algotyrmu z „backtrackingiem". **Średni czas działania: 55 ms.**

- **Algorytm znajdowania Sudoku i wyodrębniania cyfr.**
  - Skalujemy zdjęcie do wymiarów 1000px szerokości do szybszego przeprowadzania algorytmów usuwania szumu.
  - Konwertujemy zdjęcie na czarno białe i stosujemy filtr usuwający szum do łatwiejszego znajdowania osobnych obiektów na zdjęciu.
  - Konwertujemy zdjęcie na gammę binarną (tylko kolor czarny[0,0,0] i biały[255,255,255]) z uwzględnieniem różnicy wartości na zbiorze pikseli.
  - Pobieramy kontur z każdego osobnego zbioru pikseli i szukamy z największym polem.
  - Kontur (który jest zbiorem punktów) obrabiamy przez obliczenie otoczki wypukłej do pobrania dokładniejszej pozycji sudoku.
  - Wycinamy wszystko wewnątrz naszej otoczki lecz z oryginalnego zdjęcia.
  - Zmniejszamy zdjęcie o 1px by pozbyć się pozostałości po otoczce
  - Otrzymane sudoku skalujemy do wymiaru 600x600 px w celu zachowania mniej więcej stałych proporcji kolejnych kratek w sudoku (w stosunku do innych zdjęć)
  - Pracując na oryginalnym zdjęciu stosujemy tym razem medianBlur do pozbycia się szumu i znów konwertujemy na obraz binarny tym razem z większym naciskiem na różnice kolorów pikseli (pozostawienie tylko ewentualnych części ramek i cyfr).
  - Szukamy linii dłuższych niż 65px (zapewnienie by cyfra nie została rozpozna jako linia) i usuwamy je z zdjęcia.
  - Wycinamy metodą prymitywną (okienka wielkości 60x60) i sprawdzamy czy znajduje się tam obiekt według parametrów:

    - ratio znalezionego obiektu musi być z przedziału (0.3; 4.0).
    - pole które zajmuje prostokąt opisany na tym obiekcie musi być większe niż 50.
  - Wycinamy obiekt.
  - Stosujemy jeden z algorytmów rozpoznawania cyfr na danym obiekcie.

3.2.3 Użyte wzorce projektowe.

- **Singleton**
  Tworzenie nowych obiektów ocrów jest bardzo kosztowane i czasochłonne temu też zdecydowaliśmy się opakowac je w ten wzorzec projektowy. Umożliwiło nam to swobodny dostęp do tych obiektów.
- **Builder**
  Podczas każdego zapytania o ekstrakcjie sudoku tworzona jest nowa instancja BaseSudokuExtractora, który zarządza całym procesem ekstrakcji danych ze zdjęcia. Obiekt ten musi zostać stworzony na nowo, ponieważ korzysta on z danych przesłanych wraz z zapytaniem. Niestety konstrukcja tego obiektu jest skomplikowana dlatego też postanowiliśmy zastosować ten wzorze, aby ułatwić konstruowanie tak skomplikowanego obiektu oraz móc łatwiej zarządzać procesem tworzenia.
- **Stategy**
  Najważniejsze składowe BaseSudokuExtractora są reprezentowane przez odpowiednie klasy. Dzięki takiemu rozwiązaniu możemy w łatwy sposób rozszerzać naszą RecognizerLib oraz w przyjemny sposób testować nasze oprogramowanie.

3.3 Parametryzacja,

Realizacja wymagania [W8].

Proces obróbki obrazu jest bardzo skomplikowany i wpływa na niego wiele czynników takich jak jakoś zdjęcia, nasycenie kolorów, kontrast. Dlatego też nie jesteśmy w stanie dobrać takich parametrów, aby nasz algorytm znajdowania sudoku zawsze dawał najlepsze wyniki. Aby rozwiązać ten problem postanowiliśmy dać możliwość użytkownikowi zmiany niektórych (najbardziej istotnych) parametrów.

Parametry:

- Line threshold – odpowiada za to z jakim prawdopodobieństwem coś zostanie zakwalifikowane jako linia.
- Line gap – odległość, która musi być zachowana po między liniami.
- Minimum line size – minimalna długość, którą musi mieć odcinek aby został zakwalifikowany jako linia
- Blur size – jak duże kwadraty mają być brane pod uwagę podczas rozmycia.
- Blur block size – jak duży kwadrat ma być brany pod uwagę podczas usuwania zbędnych pixeli.
- Blur c – stała dodawana podczas liczenia rozmycia.
- Scalling
  - Fixed width scaling – obraz skalowany jest do szerokości
  - Max axis resize – obraz skalowany jest do dłuższego wymiaru
  - None – obraz nie jest skalowany
- Recognition – używany model do rozpoznawania cyfr. Dostępne:
  - SVM
  - Tesseract
  - ANN
- Strict mode – czy algorytmy mają działać w sposób rygorystyczny.

3.4 System Errorów.

Realizacja wymagania [W12].

Proces obróbki zdjęcia wiąże się z ryzykiem wystąpienia różnych rodzai błędów. Dlatego też przygotowaliśmy nasz serwer oraz aplikacje w taki sposób, aby mógł reagować w takich sytuacjach i powiadomić o tym użytkownika. W przypadku wystąpienia jakiegokolwiek błędu serwer zwraca odpowiedź z krótką wiadomością opisującą problem, która następnie wyświetlana jest użytnikowi aplikcji.

3.4.1 Rodzaje błędów zwracanych przez serwer.

- $[ID : 000]$ Unknown – gdy serwer nie jest w stanie określić z jakiego powodu wystąpiło przerwanie. Może również wystąpić, gdy podane parametry rozpoznawania spowodują wyrzucenie wyjątku w bibliotece openCV.
- $[ID : 003]$ SudokuNotFound – gdy na zdjęciu nie udało się znaleźć sudoku lub było one zbyt małe aby poddać je dalszej obróbce.
- $[ID : 004]$ SolverFailed – gdy nie uda się rozwiązać sudoku
- $[ID : 007]$ FileIsCorrupted – gdy plik jest uszkodzony
- $[ID : 013]$ PageNotFound – gdy podane url jest nie poprawne.
- $[ID : 014]$ MissingParameter – gdy brakuje parametru.
- $[ID : 015]$ InvalidParameter – gdy podany parametr jest niepoprawny (np. nieprawidłowy json).
- $[ID : 016]$ CellsExtractionFailed – gdy wycinanie komórek nie powiedzie się (np. przez błędne parametry).

Wszystkie te błędy posiadają własny kod oraz informacja o nich jest przekazana jako json. Przykładowa błędna odpowiedź serwera:

{"errorCode":1,"errorMessage":"Strona nie znaleziona"}

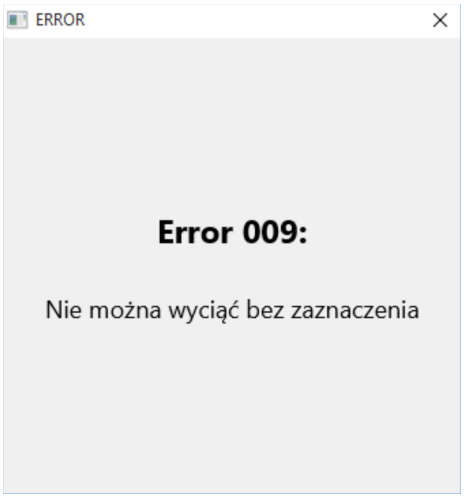
3.4.2 Rodzaje błedów wyświetlanych przez aplikacje.

- $[ID : 001]$ - Nie wykryto odpowiedniej wersji JAVY. – gdy urządzenie na którym uruchamiana jest aplikacja nie posiada Javy w wersji 8
- $[ID : 002]$ - Nieznany format pliku (tylko .jpg, .png). – gdy użytkownik próbuje otworzyć w aplikacji plik o rozszerzeniu, którego nie wspieamy
- $[ID : 004]$ – Sudoku nie posiada rozwiązania – gdy nie uda się rozwiązać sudoku
- $[ID : 005]$ – Brak połączenia z siecią. – gdy użytkownik próbuje uzyskać odpowiedź od serwara, ale nie jesteśmy w stanie się z nim połaczyć
- $[ID : 006]$ – Nie udało się połączyć z serwerem – gdy użytkownik chce uzyskać odpowiedź z serwera jednak ten nie jest w stanie nam jej dostarczyć
- $[ID : 007]$ - Plik nie istnieje. – gdy użytkownik próbuje otworzyć w aplikacji plik, który nie istnieje

- [ID : 008] - Nie można wpisać tej wartości w to pole. – gdy użytkownik próbuje wpisać do sudoku liczbę, która zgodnie z zasadami sudoku nie może znajdować się w tym miejscu (np. dwie takie same wartości w tym samym rzędzie/kolumnie)
- [ID : 009] - Nie można wyciąć bez zaznaczenia. – gdy użytkownik użyje przycisku 'Wytnij' bez wcześniejszego zaznacznia części zdjęcia do wycięcia
- [ID : 010] - Serwer nie odpowiedział wystarczająco szybko. – gdy użytkownik czeka na odpowiedź z serwera dłużej niż jest to ustalone
- [ID : 011] - Plik nie jest poprawny. – gdy użytkownik próbuje otworzyć w aplikacji plik, który ma poprawny format, ale jest uszkodzony
- [ID : 012] - Nie udało się otworzyć zdjęcia. – gdy użytkownik próbuje otworzyć w aplikacji plik, który został usunięty

Oprócz powyższych błedów aplikacja wyświetla błedy otrzymane od serwera oznaczone odpowiednim ID

Przykładowy błąd widoczny dla użytkownika:



4 Zmiany wprowadzone w projekcie.

4.1 Zmiany w wymaganiach.

- [$W3$] Dodanie obsługi formatu .png.

4.2 Zmiany w scenariuszach przypadków użycia.

- [$PU1$] Wczytanie obrazu.
  Dodanie obsługi formatu .png.
- [$PU7$] Przycięcie zdjęcia.
  Zostało zmienione, gdyż aktualne rozwiązanie jest bardziej intuicyjne.
- [$PU16$] Wyczyszczenie sudoku.
  Zostało zmienione ze względu na zmianę wymagań (dodanie możliwości wpisania ręcznie sudoku).

4.3 Instrukcja obsługi.

Instrukcja obsługi jako osobny plik, gdyż pole "O Programie" jest zbyt małe, by wyczerpująco opisać w nim działanie aplikacji,

5 Zmiany w organizacji pracy.

W trakcie trwania projektu wystąpiły opóźnienia widoczne w diagramach w folderze "Organizacja Pracy". Poniżej klasyfikacja czynników, które wpłynęły na plan pracy zespołu.

5.1 Opóźnienia po stronie serwera.

[$S001$] : *2 dni opóźnienia*
Opóźnienie spowodowane problemem z czyszczeniem szumu na zdjęciach.

[$S002$] : *0 dni opóźnienia*
Czas wykonania zadania został niezmieniony. Opóźnienie wywołane wcześniejszym zadaniem (Kod opóźnienia S001). Mimo opóźnienia wszystko zmieściło się w deadlinie założonym w tym bloku. Brak opóźnienia produkcyjnego.

[$S003$] : *7 dni opóźnienia*
Uszkodzenie mechaniczne laptopa jednego z programistów. Opóźnienie nie wpłynęło na dalszy przebieg produkcji aplikacji, przez opóźnienia wykonania testów narzucone przez model waterfall.

[$ST001$] : *7 dni opóźnienia*
Opóźnienie spowodowane dostosowanie się do wykładu i do modelu waterfall.

[$ST002$] : *1 dzień opóźnienia*
Opóźnienie spowodowane dostosowanie się do wykładu i do modelu waterfall. Jest konsekwencją opóźnienia ST001.

[$ST003$] : *6 dni opóźnienia*
Opóźnienie spowodowane dostosowanie się do wykładu i do modelu waterfall. Opóźnienie nie wpłynęło na proces produkcyjny. Potrzeba oczekiwania na skończenie 3.1.3 (Testy odczytu sudoku), by zacząć 3.4 (Testy serwera)

[$ST004/ST005$] : *0 dni opóźnienia*
Byliśmy zmuszeni wykonać testy manualne wcześniej niż zakładał waterfall. Bez przetestowania tych funkcjonalności nie mogliśmy ruszyć dalej projektu. Błąd wpisany ze względu na odejście od modelu obranego przez projekt.

5.2 Opóźnienia po stronie klienta.

[$A001$] : *6 dni opóźnienia*
Opóźnienie spowodowane problemem rysowania canvasa. Założony zbyt mały okres czasowy. Opóźnienie nie spowodowało opóźnień produkcyjnych przed testowaniem.

[$A002$] : *0 dni opóźnienia*
Opóźnienie spowodowane złym założeniem kolejności wykonywania działań. Konieczność wykonywania wysłania po narysowaniu interfejsu.

[$A003$] : *2 dni opóźnienia*
Opóźnienie spowodowane złym założeniem kolejności wykonywania działań.

[$AT001$] : *14 dni opóźnienia*
Opóźnienie spowodowane dostosowanie się do wykładu i do modelu waterfall.

[$AT002$] : *14 dni opóźnienia*
Opóźnienie spowodowane dostosowanie się do wykładu i do modelu waterfall.

5.3 Opóźnienia ogólne.

[$T001$] : *10 dni opóźnienia*
Opóźnienie spowodowane dostawcą serwerów dedykowanych. Problem występował z wirtualnym portfelem dostawcy i systemem przelewów. Ostatecznie serwer został wykupiony u innego dostawcy po zwróceniu środków.

[$T002$] : *10 dni opóźnienia*
Opóźnienia spowodowane opóźnieniem T001.

[$T003$] : *10 dni opóźnienia*
Opóźnienia spowodowane opóźnieniem T001.

[$T004$] : *18 dni opóźnienia*
Opóźnienia spowodowane opóźnieniem T001. Dodatkowo poświęciliśmy więcej czasu na testy systemowe. Całość zakończona przed deadlinem.

5.4 Wnioski.

Podsumowując pracę naszego zespołu można zaobserwować błąd popełniony przez nas w fazie planowania, mianowicie przeznaczenie zbyt małej ilości czasu na testy.