Dokumentacja projektu zaliczeniowego

Przedmiot: Inżynierski projekt zespołowy 1

Temat: **Anonimowane Algorytmy – aplikacja AlgoLearn**

Autorzy: **Monika Rozmarynowska**

**Jakub Kucharski**

**Krzysztof Bieniek**

**Olaf Maliszewski**

**Krzysztof Kubiś**

**Piotr Wojdalski**

Grupa: 330

Kierunek: informatyka

Rok akademicki: 3

Poziom i semestr: I/5

Tryb studiów: stacjonarne

Spis treści

[2 Odnośniki do innych źródeł 4](#_Toc1976793)

[3 Słownik pojęć 5](#_Toc1976794)

[4 Wprowadzenie 6](#_Toc1976795)

[4.1 Cel dokumentacji 6](#_Toc1976796)

[4.2 Przeznaczenie dokumentacji 6](#_Toc1976797)

[4.3 Opis organizacji lub analiza rynku 6](#_Toc1976798)

[4.4 Analiza SWOT organizacji 6](#_Toc1976799)

[5 Specyfikacja wymagań 7](#_Toc1976800)

[5.1 Charakterystyka ogólna 7](#_Toc1976801)

[5.2 Wymagania funkcjonalne 7](#_Toc1976802)

[5.3 Wymagania niefunkcjonalne 8](#_Toc1976803)

[6 Zarządzanie projektem 9](#_Toc1976804)

[6.1 Zasoby ludzkie 9](#_Toc1976805)

[6.2 Harmonogram prac 9](#_Toc1976806)

[6.3 Etapy/kamienie milowe projektu 9](#_Toc1976807)

[7 Zarządzanie ryzykiem 10](#_Toc1976808)

[7.1 Lista czynników ryzyka 10](#_Toc1976809)

[7.2 Ocena ryzyka 10](#_Toc1976810)

[7.3 Plan reakcji na ryzyko 10](#_Toc1976811)

[8 Zarządzanie jakością 11](#_Toc1976812)

[8.1 Scenariusze i przypadki testowe 11](#_Toc1976813)

[9 Projekt techniczny 12](#_Toc1976814)

[9.1 Opis architektury systemu 12](#_Toc1976815)

[9.2 Technologie implementacji systemu 12](#_Toc1976816)

[9.3 Diagramy UML 12](#_Toc1976817)

[9.4 Charakterystyka zastosowanych wzorców projektowych 12](#_Toc1976818)

[9.5 Projekt bazy danych 12](#_Toc1976819)

[9.6 Projekt interfejsu użytkownika 12](#_Toc1976820)

[9.7 Procedura wdrożenia 13](#_Toc1976821)

[10 Dokumentacja dla użytkownika 14](#_Toc1976822)

[11 Podsumowanie 15](#_Toc1976823)

[11.1 Szczegółowe nakłady projektowe członków zespołu 15](#_Toc1976824)

[12 Inne informacje 16](#_Toc1976825)

# Odnośniki do innych źródeł

* + Zarządzania projektem – [Trello - AlgoLearn](https://trello.com/b/nc3SZWBF/algolearn)
  + Wersjonowanie kodu – [GitHub - AlgoLearn](https://github.com/Rozmarynka19/AlgoLearn)
  + System obsługi defektów – np. Bitbucket, JazzHub

# Słownik pojęć

1. **Użytkownik** – osoba fizyczna korzystająca z aplikacji AlgoLearn.
2. **Aplikacja/System** – rozumie się przez to aplikację AlgoLearn.
3. **Dokumentacja** – niniejszy dokument

# Wprowadzenie

## Cel dokumentacji

Niniejsza Dokumentacja ma za zadanie być podstawą do stworzenia Aplikacji, testowania funkcjonalności oferowanych przez Aplikację oraz późniejszej jej konserwacji.  
W związku z powyższym, Dokumentacja zawiera:

* analizę korzyści dla docelowych odbiorców Aplikacji.
* specyfikację wymagań Aplikacji,
* harmonogram prac nad Aplikacją,
* opis architektury Aplikacji wraz z zastosowanymi technologiami.

## Przeznaczenie dokumentacji

Niniejsza Dokumentacja jest dla:

* Programistów Aplikacji
* Testerów Aplikacji
* Użytkowników Aplikacji

## Opis organizacji lub analiza rynku

Aplikacja będzie przeznaczona dla studentów Wydziału Informatyki Zachodniopomorskiego Uniwersytetu Technologicznego w Szczecinie. Jako twórcy aplikacji oraz jednocześnie jako studenci wiemy, czego najbardziej nam brakowało podczas opanowywania materiału z przedmiotu Algorytmy 2. Dlatego chcemy, aby przyszli studenci mieli łatwiej.

Biorąc pod uwagę, że algorytmy są uważane przez studentów za dość trudny przedmiot podczas studiów, aplikacja będzie nieodzowną pomocą podczas nauki.

Przypuszczamy, że po wdrożeniu aplikacji procent niezdanych egzaminów z przedmiotu Algorytmy 2 na WI ZUT drastycznie się zmniejszy z powodu zwiększenia zrozumienia omawianych algorytmów.

Istnieje możliwość w przyszłości sprzedawania aplikacji innym uczelniom technicznym.

# Specyfikacja wymagań

## Charakterystyka ogólna

### Definicja produktu

System do nauki i ćwiczeń algorytmów, prezentujący zagadnienia teoretyczne oraz praktyczne z dziedziny algorytmiki.

### Podstawowe założenia

System ma na celu ułatwienie oraz przyśpieszenie nauki algorytmów studentom kierunków technicznych, umożliwienie poznania problemu z perspektywy innych studentów, zapewnienie dostępu do zakresu materiału, zgrupowanego w jednym miejscu. System może służyć również jako narzędzie dla dydaktyków w ramach zajęć.

### Cel biznesowy

Usprawnienie nauczania, poprawa wyników oraz efektywności w nauce, zapewnienie dodatkowego źródła wiedzy z danego tematu.

### Użytkownicy

1. Użytkownik

### Korzyści z systemu

1. Użytkownik
   1. Dostęp do materiałów naukowych.
   2. Możliwość rozwiązywania testów w celu nauki.
   3. Dostęp do programistycznej realizacji algorytmów.
   4. Dostęp do wizualizacji problemów algorytmicznych.
   5. Możliwość monitorowania swojego postępu w nauce.

### Ograniczenia projektowe i wdrożeniowe

System do poprawnego działania potrzebuje komputera z systemem operacyjnym z rodziny Windows.

## Wymagania funkcjonalne

### Lista wymagań

* Wyświetlanie informacji na temat algorytmu.
* Wizualizacja algorytmu.
* Testowanie wiedzy na temat algorytmu.
* Monitorowanie postępu w nauce.

### Diagramy przypadków użycia

Tutaj same diagramy – bez specyfikacji, ale każdy diagram z tytułem i na osobnej stronie

### Szczegółowy opis wymagań

|  |  |
| --- | --- |
| #1 | |
| Numer | **1.1** |
| Nazwa | **Wyświetlanie informacji na temat algorytmu korzystając z prawego panelu.** |
| Uzasadnienie biznesowe | Użytkownik powinien mieć dostęp do materiałów szkoleniowych. |
| Użytkownicy | Użytkownik |
| Scenariusz główny | |
| Warunki początkowe | Użytkownik jest w głównym oknie aplikacji. |
| 1. | Użytkownik wybiera z listy po lewej interesujący go algorytm |
| 2. | Aplikacja po prawej stronie wyświetla 3 pod panele. |
| 3. | Użytkownik wybiera pod panel „wprowadzenie”. |
| Efekty | Na prawym oknie zostaje wyświetlona informacja na temat algorytmu. |
| Wymagania niefunkcjonalne | |
| 1. | szczegółowe wobec poszczególnych wymagań funkcjonalnych Wyświetlenie informacji o algorytmie, po wybraniu go z listy, nie powinno trwać dłużej niż 0.5s |
| 2. | System powinien działać na komputerach z wspieranym systemem z rodziny Windows. |
|  | |
| Częstotliwość | 5 |
| Istotność | 5 |

|  |  |
| --- | --- |
| Scenariusz alternatywny 1 | |
| ID | **1.2** |
| Nazwa | **Wyświetlanie informacji na temat algorytmu korzystając z rozwijanej listy po lewej.** |
| Warunki początkowe | Użytkownik jest w głównym oknie aplikacji |
| 1. | Użytkownik rozwija podlistę wybranego algorytmu po lewej stronie aplikacji. |
| 2. | Użytkownik wybiera z podlisty „wprowadzenie” |
| Efekty | Na prawym oknie zostaje wyświetlona informacja na temat algorytmu. |
| Wymagania niefunkcjonalne | |
| 1. | Wyświetlenie informacji o algorytmie, po wybraniu go z listy, nie powinno trwać dłużej niż 0.5s |
| 2. | System powinien działać na komputerach z wspieranym systemem z rodziny Windows. |
|  | |
| Częstotliwość | 5 |
| Istotność | 5 |

|  |  |
| --- | --- |
| #2 | |
| Numer | **2** |
| Nazwa | **Wizualizacja algorytmu.** |
| Uzasadnienie biznesowe | Użytkownik powinien mieć dostęp do wizualizacji algorytmu. |
| Użytkownicy | Użytkownik |
| Scenariusz główny | |
| Warunki początkowe | Użytkownik jest w głównym oknie aplikacji. |
| 1. | Użytkownik wybiera z listy po lewej interesujący go algorytm |
| 2. | Aplikacja po prawej stronie wyświetla 3 pod panele. |
| 3. | Użytkownik wybiera pod panel „wizualizacja”. |
| Efekty | Na prawym oknie zostaje wyświetlona informacja na temat algorytmu. |
| Wymagania niefunkcjonalne | |
| 1. | szczegółowe wobec poszczególnych wymagań funkcjonalnych Wyświetlenie wizualizacji algorytmu, po wybraniu go z listy, nie powinno trwać dłużej niż 0.5s |
| 2. | System powinien działać na komputerach z wspieranym systemem z rodziny Windows. |
|  | |
| Częstotliwość | 5 |
| Istotność | 5 |

|  |  |
| --- | --- |
| #3 | |
| Numer | **3** |
| Nazwa | **Testowanie wiedzy na temat algorytmu.** |
| Uzasadnienie biznesowe | Użytkownik powinien móc przetestować swoją wiedzę na temat przedstawionego algorytmu. |
| Użytkownicy | Użytkownik |
| Scenariusz główny | |
| Warunki początkowe | Użytkownik jest w głównym oknie aplikacji. |
| 1. | Użytkownik wybiera z listy po lewej interesujący go algorytm |
| 2. | Aplikacja po prawej stronie wyświetla 3 pod panele. |
| 3. | Użytkownik wybiera pod panel „test wiedzy”. |
| Efekty | Na prawym oknie zostaje wyświetlona test wielo lub jednokrotnego wyboru. Możliwe też zadanie otwarte. |
| Wymagania niefunkcjonalne | |
| 1. | Podpowiedzi do zadań, po spełnieniu warunków, powinny się wyświetlić po maksymalnie 0.5s |
| 2. | System powinien działać na komputerach z wspieranym systemem z rodziny Windows. |
|  | |
| Częstotliwość | 4 |
| Istotność | 5 |

|  |  |
| --- | --- |
| #4 | |
| Numer | **4** |
| Nazwa | **Monitorowanie postępu w nauce.** |
| Uzasadnienie biznesowe | Użytkownik powinien mieć możliwość monitorowania postępów w nauce. |
| Użytkownicy | Użytkownik |
| Scenariusz główny | |
| Warunki początkowe | Użytkownik jest w głównym oknie aplikacji. |
| 1. | Użytkownik wybiera z głównego okna aplikacji zakładkę monitorowanie postępów. |
| Efekty | Na prawym oknie zostają wyświetlone wyniki postępów rozwiązywania poszczególnych algorytmów. |
| Wymagania niefunkcjonalne | |
| 1. | Pasek postępu w oknie startowym danego tematu, powinien być zaktualizowany od następnego przejścia do okna startowego po wykonaniu/przejściu przez zaliczaną część. |
| 2. | System powinien działać na komputerach z wspieranym systemem z rodziny Windows. |
|  | |
| Częstotliwość | 5 |
| Istotność | 5 |

## Wymagania niefunkcjonalne

1. Wydajność
   1. Wyświetlenie informacji o algorytmie, po wybraniu go z listy, nie powinno trwać dłużej niż 0.5s
   2. Po uruchomieniu wizualizacji algorytmu, animacja powinna się rozpocząć maksymalnie po 0.5s
   3. Pasek postępu w oknie startowym danego tematu, powinien być zaktualizowany od następnego przejścia do okna startowego po wykonaniu/przejściu przez zaliczaną część.
   4. Podpowiedzi do zadań, po spełnieniu warunków, powinny się wyświetlić po maksymalnie 0.5s
2. Inne cechy jakości
   1. System powinien działać na komputerach z wspieranym systemem z rodziny Windows.

# Zarządzanie projektem

## Zasoby ludzkie

1. Programiści (3 osoby)
2. Tester (1 osoba)
3. UI designer (1 osoba)
4. Kierownik projektu (1 osoba)

## Harmonogram prac

## Etapy/kamienie milowe projektu

1. Opracowanie specyfikacji wymagań
   1. wymagania funkcjonalne
   2. wymagania niefunkcjonalne
2. Ocena ryzyka projektu
3. Zaprojektowanie architektury aplikacji
   1. diagramy przypadków użycia
   2. diagramy klas
   3. diagramy komponentów
   4. diagramy pakietów
   5. diagramy maszyny stanowej
   6. diagramy komunikacji
4. Zaprojektowanie GUI
5. Opracowanie testów
   1. opracowanie testów jednostkowych
   2. opracowanie testów integracyjnych
   3. opracowanie testów akceptacyjnych
6. Wprowadzenie poprawek do architektury aplikacji
7. Wytworzenie pakietów
8. Przeprowadzenie testów jednostkowych
9. Integracja pakietów
10. Przeprowadzenie testów integracyjnych
11. Wdrożenie systemu
12. Testowanie systemu w środowisku pracy

# Zarządzanie ryzykiem

## Lista czynników ryzyka, ocena ryzyka oraz środki zaradcze

|  |  |  |  |
| --- | --- | --- | --- |
| Ryzyko | Stopień zagrożenia dla projektu | Szansa na wystąpienie | Środki zaradcze |
| Niski poziom znajomości języka Javy przez zespół. | Wysoki | Duże | Zdecydowano aby wszelkie problemy dotyczące języka były zgłaszane, w grupie szybciej będzie znaleźć błąd i wymienić się potrzebną wiedzą. |
| Nowy zespół, niezintegrowany | Średni | Średni | Wspólna pomoc i cel powinny pomóc w integracji. W razie problemów będą one omawiane w grupie przy obecności lidera. |
| Pierwszy projekt realizowany na większą skalę-brak doświadczenia | Wysoki | Duże | W razie wątpliwości o postępowaniu projektu, jego dalszych planach, po próbie samodzielnego rozwiązania problemu zgłaszamy problem profesorowi El Freyowi jako konsultantowi. Wskazówki doświadczonego programisty powinny pomóc. |
| Brak doświadczenia z wizualizacją i potrzebnymi narzędziami. | Średni | Duże | Wcześniejsze wyznaczenie osób odpowiedzialnych do wizualizacji aby mieli czas podszkolić się z tego zakresu. Pomoc innych w razie konieczności. W razie wątpliwości problem do konsultacji z profesorem El Freyem. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Analzia SWOT cech zespołu oraz okoliczności

|  |  |  |
| --- | --- | --- |
| Cechy | Pozytywne | Negatywne |
| Wewnętrzne  (cechy organizacji) | - duże zainteresowanie i motywacja  - temat wybrany przez wszystkich  - | - małe doświadczenie  - braki w wiedzy  - |
| Zewnętrzne (cechy otoczenia) | - dobry prowadzący  - dobre warunki pracy  - dostęp do Internetu najczęściej jest  - | -brak możliwości obgadania spraw twarzą w twarz  -możliwość utraty członka zespołu -> utrata dostępu Internetu  - |

# Zarządzanie jakością

## Scenariusze i przypadki testowe

**Numer 1**

* Przełączanie między dostępnymi językami programistycznymi
* Rodzaj testów: testy dymne
* Tester – Użytkownik
* Termin – nie wcześniej niż po realizacji Zaprojektowanie GUI (Punkt 6.2)
* Narzędzia wspomagające – brak

|  |  |
| --- | --- |
| **Przebieg działań: Przełączanie między dostępnymi językami programistycznymi** | |
| **Użytkownik** | **System** |
|  | System czeka na polecenie od Użytkownika |
| Użytkownik przy pomocy kursora myszy najeżdża na przycisk, który wybiera dostępny język |  |
| Użytkownik zatwierdza wybór poprzez kliknięcie myszą. |  |
|  | System odbiera wydane polecenie |
|  | System podmienia treść w oknie na wybrany język programistyczny |

* Założenia – Użytkownikowi wyświetli się poprawnie treść kodu w wybranym języku
* Środowisko – aplikacja desktopowa
* Warunki wstępne – Użytkownik musi mieć uruchomioną aplikację, wybrane w głównym menu “Opis algorytmu” oraz wybrany algorytm “Drzewo AVL” z listy algorytmów.
* **Zestaw danych testowych**

|  |  |
| --- | --- |
| **Przebieg działań: Przełączanie między dostępnymi językami programistycznymi** | |
| **Użytkownik** | **System** |
|  | System czeka na polecenie od Użytkownika |
| Użytkownik przy pomocy kursora myszy najeżdża na przycisk z podpisem “Java” |  |
| Użytkownik zatwierdza wybór poprzez kliknięcie myszą. |  |
|  | System odbiera wydane polecenie |
|  | System podmienia treść w oknie na kod napisany w języku Java |

|  |  |
| --- | --- |
| **Przebieg działań: Przełączanie między dostępnymi językami programistycznymi (brak dostępnego języka)** | |
| **Użytkownik** | **System** |
|  | System czeka na polecenie od Użytkownika |
| Użytkownik przy pomocy kursora myszy najeżdża na przycisk z podpisem “Java” |  |
| Użytkownik zatwierdza wybór poprzez kliknięcie myszą. |  |
|  | System odbiera wydane polecenie |
|  | System pozostaje w stanie bezczynności |

**Numer 2**

* Dodanie nowej wartości do węzła w wizualizacji
* Rodzaj testów: testy dymne
* Tester – Użytkownik
* Termin – nie wcześniej niż po realizacji Zaprojektowanie GUI (Punkt 6.2)
* Narzędzia wspomagające – brak

|  |  |
| --- | --- |
| **Przebieg działań: Dodanie nowej wartości do węzła w wizualizacji** | |
| **Użytkownik** | **System** |
|  | System czeka na polecenie od Użytkownika |
| Użytkownik przy pomocy bocznego menu w polu “Dodaj węzeł” wprowadza z klawiatury wartość. |  |
| Użytkownik zatwierdza wybór poprzez kliknięcie przycisku z symbolem strzałki |  |
|  | System odbiera wydane polecenie |
|  | System dodaje do wizualizacji wybraną wartość |

* Założenia – Użytkownikowi wyświetli się poprawnie wybrana przez niego wartość na wizualizacji
* Środowisko – aplikacja desktopowa
* Warunki wstępne – Użytkownik musi mieć uruchomioną aplikację, wybrane w głównym menu “Wizualizacja” oraz wybrany algorytm “Drzewo Czerwono-Czarne” z listy algorytmów
* **Zestaw danych testowych**

|  |  |
| --- | --- |
| **Przebieg działań: Dodanie nowej wartości do węzła w wizualizacji** | |
| **Użytkownik** | **System** |
|  | System czeka na polecenie od Użytkownika |
| Użytkownik przy pomocy bocznego menu w polu “Dodaj węzeł” wprowadza z klawiatury wartość=12. |  |
| Użytkownik zatwierdza wybór poprzez kliknięcie przycisku z symbolem strzałki |  |
|  | System odbiera wydane polecenie |
|  | System dodaje do wizualizacji wartość=12 |

|  |  |
| --- | --- |
| **Przebieg działań: Dodanie nowej wartości do węzła w wizualizacji (błędna wartość)** | |
| **Użytkownik** | **System** |
|  | System czeka na polecenie od Użytkownika |
| Użytkownik przy pomocy bocznego menu w polu “Dodaj węzeł” wprowadza z klawiatury wartość=a. |  |
| Użytkownik zatwierdza wybór poprzez kliknięcie przycisku z symbolem strzałki |  |
|  | System odbiera wydane polecenie |
|  | System pozostaje w stanie bezczynności |

|  |  |
| --- | --- |
| **Przebieg działań: Dodanie nowej wartości do węzła w wizualizacji (brak danych)** | |
| **Użytkownik** | **System** |
|  | System czeka na polecenie od Użytkownika |
| Użytkownik przy pomocy bocznego menu w polu “Dodaj węzeł” wprowadza z klawiatury wartość=’ ‘ (pusty znak). |  |
| Użytkownik zatwierdza wybór poprzez kliknięcie przycisku z symbolem strzałki |  |
|  | System odbiera wydane polecenie |
|  | System pozostaje w stanie bezczynności |

**Numer 3**

* Usunięcie węzła w wizualizacji
* Rodzaj testów: testy dymne
* Tester – Użytkownik
* Termin – nie wcześniej niż po realizacji Zaprojektowanie GUI (Punkt 6.2) i zrealizowaniu przypadku testowego oznaczonego numerem 2 (8.1)
* Narzędzia wspomagające – brak

|  |  |
| --- | --- |
| **Przebieg działań: Usunięcie węzła w wizualizacji** | |
| **Użytkownik** | **System** |
|  | System czeka na polecenie od Użytkownika |
| Użytkownik przy pomocy bocznego menu w polu “Usuń węzeł” wprowadza z klawiatury wartość. |  |
| Użytkownik zatwierdza wybór poprzez kliknięcie przycisku z symbolem strzałki |  |
|  | System odbiera wydane polecenie |
|  | System usuwa z wizualizacji wybraną wartość |

* Założenia – Użytkownikowi wyświetli się poprawnie wizualizacja po wprowadzeniu zmian
* Środowisko – aplikacja desktopowa
* Warunki wstępne – Użytkownik musi mieć uruchomioną aplikację, wybrane w głównym menu “Wizualizacja” oraz wybrany algorytm “Drzewo Czerwono-Czarne” z listy algorytmów.
* **Zestaw danych testowych**

|  |  |
| --- | --- |
| **Przebieg działań: Usunięcie węzła w wizualizacji** | |
| **Użytkownik** | **System** |
|  | System czeka na polecenie od Użytkownika |
| Użytkownik przy pomocy bocznego menu w polu “Usuń węzeł” wprowadza z klawiatury wartość=12. |  |
| Użytkownik zatwierdza wybór poprzez kliknięcie przycisku z symbolem strzałki |  |
|  | System odbiera wydane polecenie |
|  | System usuwa z wizualizacji wartość=12 |

|  |  |
| --- | --- |
| **Przebieg działań: Dodanie nowej wartości do węzła w wizualizacji (błędna wartość)** | |
| **Użytkownik** | **System** |
|  | System czeka na polecenie od Użytkownika |
| Użytkownik przy pomocy bocznego menu w polu “Usuń węzeł” wprowadza z klawiatury wartość=a. |  |
| Użytkownik zatwierdza wybór poprzez kliknięcie przycisku z symbolem strzałki |  |
|  | System odbiera wydane polecenie |
|  | System pozostaje w stanie bezczynności |

|  |  |
| --- | --- |
| **Przebieg działań: Dodanie nowej wartości do węzła w wizualizacji (brak węzłów do usunięcia)** | |
| **Użytkownik** | **System** |
|  | System czeka na polecenie od Użytkownika |
| Użytkownik przy pomocy bocznego menu w polu “Usuń węzeł” wprowadza z klawiatury wartość=12 |  |
| Użytkownik zatwierdza wybór poprzez kliknięcie przycisku z symbolem strzałki |  |
|  | System odbiera wydane polecenie |
|  | System usuwa z wizualizacji wartość=12 |
| Użytkownik przy pomocy bocznego menu w polu “Usuń węzeł” wprowadza z klawiatury wartość=12 |  |
| Użytkownik zatwierdza wybór poprzez kliknięcie przycisku z symbolem strzałki |  |
|  | System odbiera wydane polecenie |
|  | System pozostaje w stanie bezczynności |

# Projekt techniczny

## Opis architektury systemu

z ew. rysunkami pomocniczymi

## Technologie implementacji systemu

*tabela z listą wykorzystanych technologii, każda z uzasadnieniem*

Java -> core

JavaFX/Swing -> biblioteka do tworzenia GUI

TODO: X -> biblioteka do tworzenia wizualizacji

## Diagramy UML

każdy diagram ma mieć tytuł oraz ma być na osobnej stronie

diagramy przypadków użycia umieszczone w punkcie 5.2.2, a nie tutaj.

### Diagram(-y) klas

### Diagram(-y) czynności

### Diagramy sekwencji

co najmniej 5, w tym co najmniej 1 przypadek użycia zilustrowany kilkoma diagramami sekwencji

### Inne diagramy

co najmniej trzy – komponentów, rozmieszczenia, maszyny stanowej itp.

## Charakterystyka zastosowanych wzorców projektowych

informacja opisowa wspomagana diagramami (odsyłaczami do diagramów UML); jeśli wykorzystano wzorce projektowe, to należy wykazać dwa z nich

## Projekt bazy danych

### Schemat

w trzeciej formie normalnej; jeśli w innej to umieć uzasadnić wybór

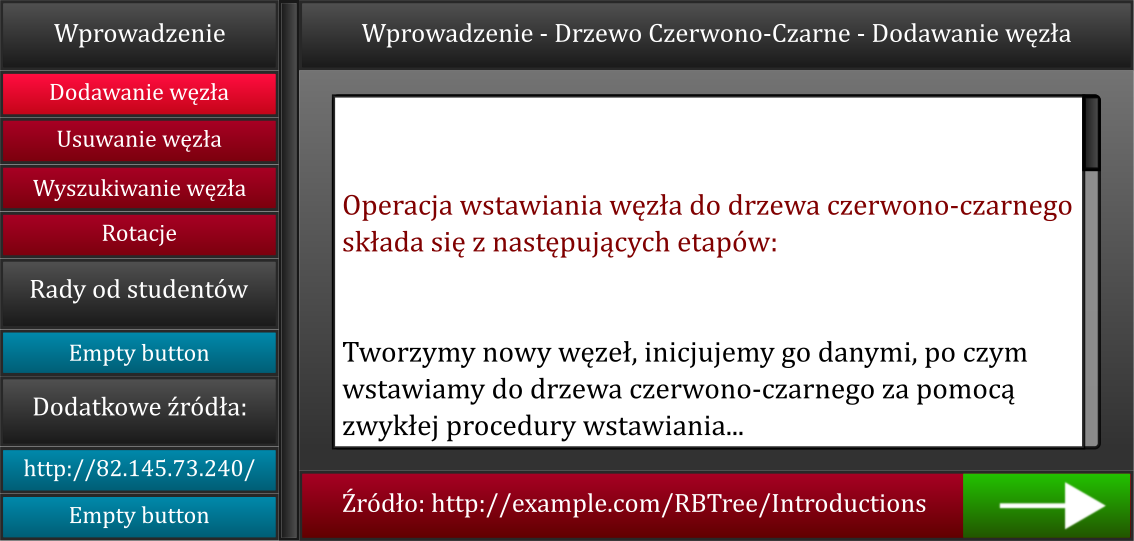
### Projekty szczegółowe tabel

## Projekt interfejsu użytkownika

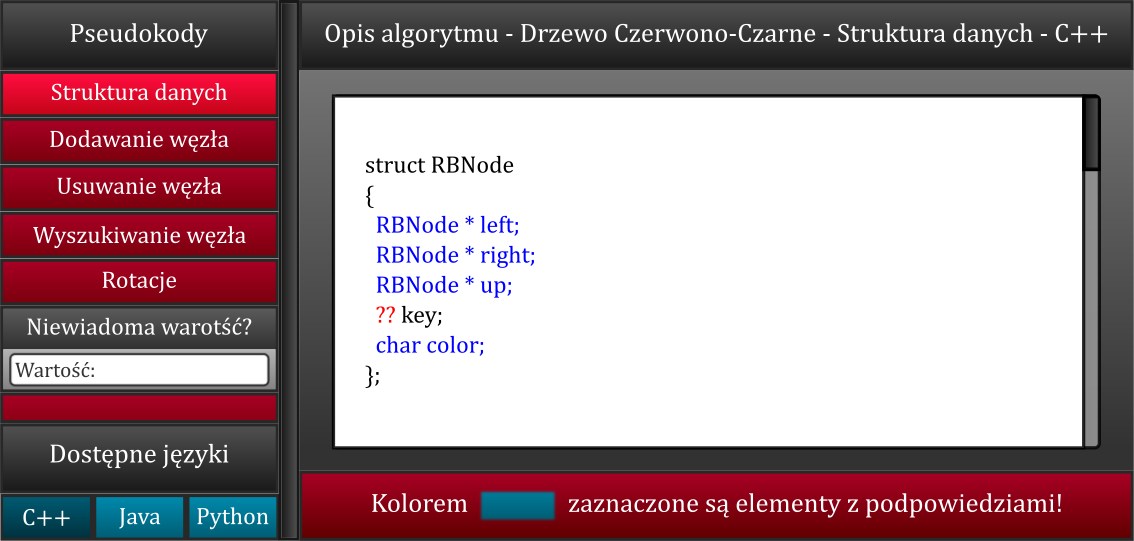
* **Okno główne apliakcji z możliwym wyborem algorytmu oraz rożnymi etapami edukacji po których zakończeniu można przystąpić to „Egzaminu”. Egzamin jest niczym innym jak podsumowaniem wiedzy zdobytej w trzech kategoriach.**



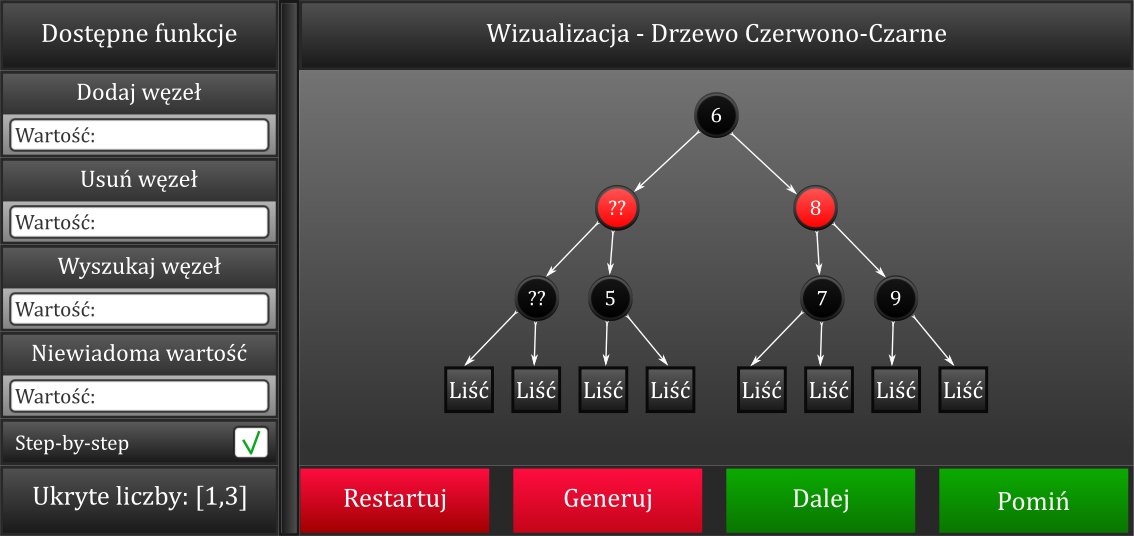
* **Wprowadzenie – część aplikacji przedstawiająca teorie i zasadę działania danego algorytmu.**



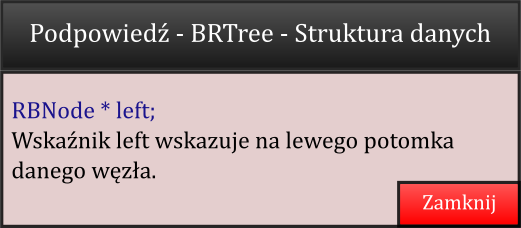
* **Opis algorytmu – kurs odpowiedzialny za przedstawienie problematyki algorytmu w postaci pseudokodu. Posiada on pola „??” w którch użytkownik sam musi podać zawartość oraz niebieskie fragemnty które są podpowiedziami. Sam pseudokod znajduje się w trzech wariantach językowych takich jak Java, Python oraz C++.**



* **Wizualizacja – graficzne przedstawienie działania danego algorytmu.**



* **Okno podpowiedzi – występuje podczas wyświetlania wskazówek dotyczących pseudokodu i nie tylko.**



### Lista głównych elementów interfejsu

okien, stron, aktywności (Android)

### Przejścia między głównymi elementami

### Projekty szczegółowe poszczególnych elementów

każdy element od nowej strony z następującą minimalną zawartością:

* numer – ID elementu
* nazwa – np. formularz danych produktu
* projekt graficzny – wystarczy schemat w narzędziu graficznym lub zrzut ekranu – z przykładowymi informacjami (nie pusty!!!)
* opcjonalnie:
* opis – dodatkowe opcjonalne informacje o przeznaczeniu, obsłudze – jeśli nazwa nie będzie wystarczająco czytelna
* wykorzystane dane – jakie dane z bazy danych są wykorzystywane
* opis działania – tabela pokazująca m.in. co się dzieje po kliknięciu przycisku, wybraniu opcji z menu itp.

## Procedura wdrożenia

jeśli informacje w harmonogramie nie są wystarczające (a zapewne nie są)

# Zasady pracy przy projekcie

## Język

Piszemy kod, komentarze, nazwy zmiennych, klas, commity itd. w języku angielskim.

## Opracowania algorytmów

Root folder nazywa się "opracowaniaAlgo".

Wewnątrz dajemy folder z nazwą algorytmu, nad jakim pracujemy (może być skrót, np. "BST" - przede wszystkim ma być wiadomo na 1. rzut oka, o jaki algorytm chodzi).

Dopiero w tym folderze wrzucamy pliki będące opracowaniem algo - obrazki, notatki, pliki Worda .docx.

## Używane notacje

Nazwy klas, plików .fxml – PascalCase, np. MainWindow.java.

Nazwy zmiennych i metod w klasie – cammelCase, np. anchorPaneRoot.

## Tworzenie nazw

Klasy będące kontrolerami koniecznie muszą mieć końcówkę „Controller”, np. MainController.java.

Pliki .fxml będące projektem okna muszą mieć końcówkę „Window”, np. MenuWindow.fxml.

Nazwy mają być samoopisujące się! Jeśli na 1. Rzut oka nie wiadomo, co dana metoda robi/do czego służy zmienna – koniecznie dodajcie komentarz nad deklaracją/definicją!

## Repozytorium

Na czas pracy nad jakimś elementem projektu, tworzymy brancha i na nim pracujemy.

Nie commitujemy bezpośrednio na mastera! Dopuszczamy tylko merge. Ma to zabezpieczyć przed dziwnymi sytuacjami na repozytorium.

Dbamy o to, aby każdy commit zawierał działającą aplikację – czyli nie comitujemy rozgrzebanych aplikacji. Jeśli musimy przełączyć brancha, to odkładamy na stos aktualne zmiany (git stash), przełączamy się, sprawdzamy coś, wracamy na naszego brancha i przywracamy naszą prace (git stash pop).

Po zrealizowaniu większej części projektu, tagujemy commita-merga na masterze nadając odpowiedni numer wersji – decyzję o tagowaniu danego etapu projektu podejmujemy na spotkaniu.

## Trello

Jeśli zauważymy problem do rozwiązania, dodajemy go do listy „do zrobienia” i oznaczamy odpowiednią etykietę:

* Pomarańczowy – zadanie związane z kodem,
* Niebieski – zadanie związane z dokumentacją,
* Żółty – zadanie związane z opracowaniem algorytmów.

Po samodzielnym wybraniu zadania do wykonania, oznaczamy siebie na tym zadaniu i przenosimy go do listy „w trakcie”. Zadania możemy wybierać w dowolnym czasie.

Po ukończonym zadaniu, przenosimy zadanie do „zrobione”.

## Spotkania

Podczas cotygodniowych spotkań, omawiamy:

* Co zrobiliśmy
* Nad czym pracujemy
* Co jest jeszcze do zrobienia

Lider zespołu ma obowiązek przygotować się do spotkania:

* zbierając informację o stanie projektu min. 1 dzień wcześniej,
* analizując stan kanbana („do zrobienia”, „w trakcie”),
* przygotowując informację zwrotną dla zespołu.

Podczas spotkań zastanawiamy się nad rozwiązywaniem aktualnych problemów oraz wybieramy dla siebie zadania do wykonania.

# Dokumentacja dla użytkownika

Opcjonalnie – dla chętnych

Na podstawie projektu docelowej aplikacji, a nie zaimplementowanego prototypu architektury

4-6 stron z obrazkami (np. zrzuty ekranowe, polecenia do wpisania na konsoli, itp.)

* pisana językiem odpowiednim do grupy odbiorców – czyli najczęściej nie do informatyków
* może to być przebieg krok po kroku obsługi jednej głównej funkcji systemu, kilku mniejszych, instrukcja instalacji lub innej pomocniczej czynności.

# Podsumowanie

## Szczegółowe nakłady projektowe członków zespołu

tabela (kolumny to osoby, wiersze to działania) pokazująca, kto ile czasu poświęcił na projekt oraz procentowy udział każdej osoby w danym zadaniu oraz wiersz podsumowania – udział każdej osoby w skali całego projektu

# Inne informacje

przydatne informacje, które nie zostały ujęte we wcześniejszych punktach