Studencka Pracownia Inżynierii Oprogramowania

Instytut Informatyki Uniwersytetu Wrocławskiego

Krystian Jasionek, Jakub Skalski

Koncepcja wykonania systemu

Wrocław, 15 grudnia 2020

Wersja 0.6

Historia zmian dokonanych w dokumencie

Data	Numer wersji	Opis	Autor
2020-11-28	0.1	Utworzenie dokumentu	Krystian Jasionek
2020-11-29	0.2	Korekta dokumentu	Jakub Skalski
2020-12-01	0.3	Korekta dokumentu	Krystian Jasionek
2020-12-02	0.4	Korekta dokumentu	Jakub Skalski
2020-12-13	0.5	Korekta dokumentu	Krystian Jasionek
2020-12-15	0.6	Korekta dokumentu	Krystian Jasionek

Spis treści

1. Wprowadzenie

1.1. Cel dokumentu

Niniejszy dokument ma na celu przedstawienie koncepcji wykonania systemu. Praca stanowi pierwsze z listopadowej listy zadań w ramach pracowni z inżynierii oprogramowania.

2. Scenariusze przypadków użycia

2.1. Utworzenie własnego algorytmu

Aktor: Zarejestrowany użytkownik.
Warunki początkowe: Użytkownik zalogowany do systemu.
Warunki końcowe: Pomyślnie utworzony algorytm, dodany do biblioteki użytkownika.
Scenariusz główny:

1. Użytkownik przechodzi do panelu algorytm.
2. Użytkownik wybiera opcję „Dodaj nowy algorytm".
3. Użytkownik wybiera opcję „Napisz nowy algorytm".
4. Otwiera się edytor kodu wewnątrz aplikacji.
5. Użytkownik zapisuje kod algorytmu.
6. Użytkownik wybiera opcję „Zapisz jako".
7. Użytkownik wprowadza nazwę algorytmu.
8. Algorytm zostaje dodany do biblioteki algorytmów użytkownika.

Rozszerzenia:

3. (a) Użytkownik wybiera opcję „Wybierz plik z algorytmem".
   - System otwiera przeglądarkę plików na komputerze użytkownika.
   - Użytkownik wybiera plik z algorytmem.
   - Przejście do kroku 8.

5. (a) Użytkownik wprowadza kod z błędem w składni
   - System podkreśla błąd w edytorze kodu i wyświetla wiadomość o jego prawdopodobnej przyczynie.
   - Użytkownik poprawia błąd.

   (b) Użytkownik opuszcza edytor, nie zapisując algorytmu
   - System wyświetla ostrzeżenie o możliwości utraty postępów i pyta użytkownika o potwierdzenie opuszczenia edytora.
   - Użytkownik udaje się do kroku 6 lub potwierdza opuszczenie edytora.

2.2. Zakup algorytmu w sklepie

Aktor: Zarejestrowany użytkownik.
Warunki początkowe: Użytkownik zalogowany do systemu.
Warunki końcowe: Dodanie algorytmu do biblioteki algorytmów użytkownika.
Scenariusz główny:

1. Użytkownik przechodzi do panelu sklepu.
2. Użytkownik wybiera kategorię „Algorytmy".
3. System wyświetla listę dostępnych algorytmów.
4. Użytkownik wybiera algorytm.
5. System otwiera podstronę opisującą algorytm.
6. Użytkownik wybiera opcję „Dodaj do koszyka".
7. Użytkownik przechodzi do panelu „Mój koszyk".
8. Użytkownik wybiera opcję „Przejdź do płatności".
9. System wyświetla podsumowanie zamówienia i pyta o potwierdzenie kontynuacji.
10. Użytkownik potwierdza kontynuację.
11. System wyświetla formularz zakupu.
12. Użytkownik uzupełnia:
    - Imię (maksymalnie 100 znaków alfanumerycznych)
    - Nazwisko (maksymalnie 100 znaków alfanumerycznych)
    - Numer karty (dokładnie 16 cyfr)
    - Data ważności (4 cyfry w formacie mm-rr)
    - CVV (dokładnie 3 cyfry)
13. Użytkownik wybiera opcję „Zapłać".
14. System wyświetla komunikat o poprawnym zakończeniu transakcji.
15. Algorytm zostaje dodany do biblioteki algorytmów użytkownika.

Rozszerzenia:

12. (a) Użytkownik wprowadza błędne dane.
    - System powiadamia, które dane zostały wprowadzone nieprawidłowo.
    - Przejście do kroku 10.
13. (a) Transakcja została odrzucona przez serwis obsługujący płatność.
    - System wyświetla komunikat „Nie można zrealizować transakcji".
    - Użytkownik zostaje przeniesiony do panelu „Mój koszyk".

2.3. Eksport gotowych wzorów

Aktor: Zarejestrowany użytkownik.
Warunki początkowe: Użytkownik zalogowany do systemu.
Warunki końcowe: Wyeksportowanie wzoru w wybranym przez użytkownika formacie i zapisanie go na komputerze użytkownika.
Scenariusz główny:

1. Użytkownik przechodzi do panelu kreatora wzorów.
2. Użytkownik przygotowuje wzór.
3. Użytkownik wybiera opcję „Wyeksportuj wzór".
4. System wyświetla listę dostępnych formatów graficznych.
5. Użytkownik wybiera format pliku.
6. System wyświetla przeglądarkę plików na komputerze użytkownika.
7. Użytkownik wybiera docelową nazwę pliku.
8. System eksportuje wzór do wskazanego formatu i przesyła go na komputer użytkownika.
9. System powiadamia użytkownika o zakończonej pomyślnie operacji.

Rozszerzenia:

7. (a) Użytkownik podał nieprawidłową nazwę pliku.
   - System powiadamia, że nazwa pliku jest nieprawidłowa.
   - Przejście do kroku 6.

2.4. Generowanie wzorów

Aktor: Zarejestrowany użytkownik.
Warunki początkowe: Użytkownik zalogowany do systemu.
Warunki końcowe: Wygenerowanie wzoru przez użytkownika.
Scenariusz główny:

1. Użytkownik przechodzi do panelu kreatora wzorów.
2. Użytkownik wybiera opcję „Wybierz algorytm".
3. System wyświetla listę dostępnych algorytmów z biblioteki użytkownika.
4. Użytkownik dokonuje wyboru algorytmu.
5. System wyświetla podgląd gotowego wzoru oraz panel edycji.
6. Użytkownik dopasowuje parametry algorytmu w panelu edycji.

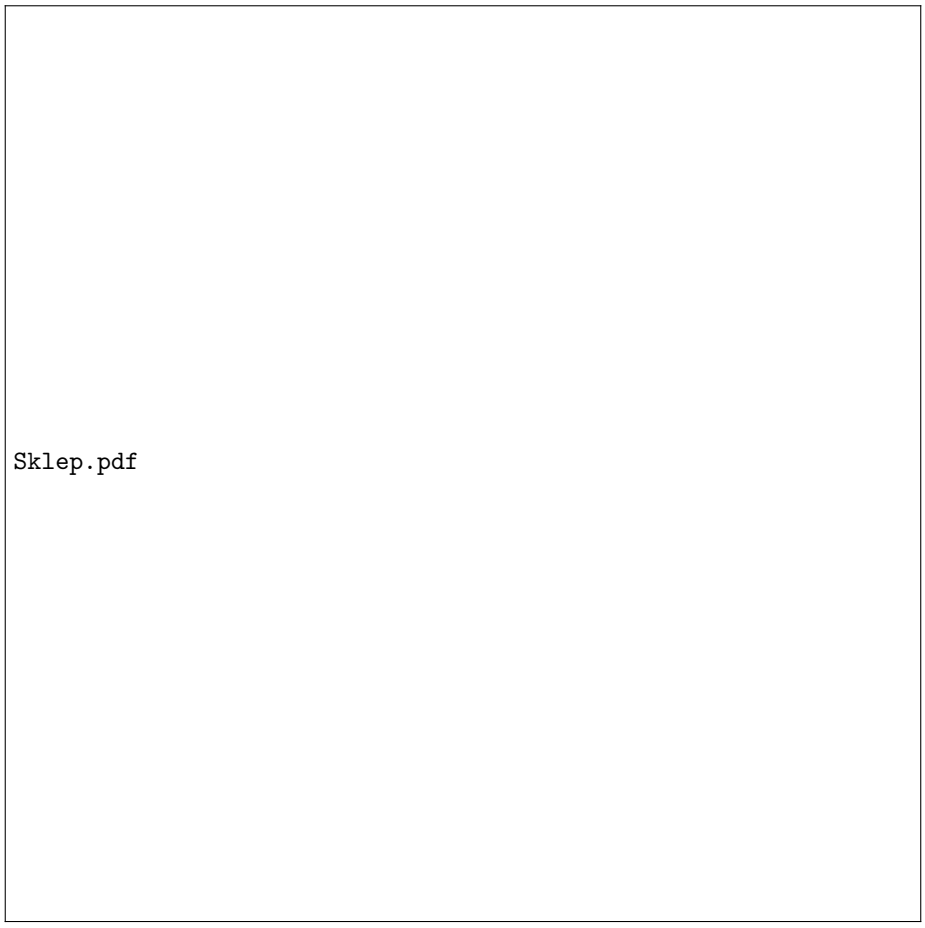
7. Użytkownik wybiera opcję „Generuj wzór“.
8. System generuje wzórna podstawie wprowadzonych danych.
9. System wyświetla komunikat o zakończonym procesie generowania wzoru.

Rozszerzenia:

6. (a) Użytkownik wprowadził nieprawidłowe dane.
   - System powiadamia, które z wprowadzonych danych są nieprawidłowe.
   - Użytkownik przechodzi do kroku 6.

3. Dialog z komputerem – projekt ekranów

Rysunek 1. Strona sklepu



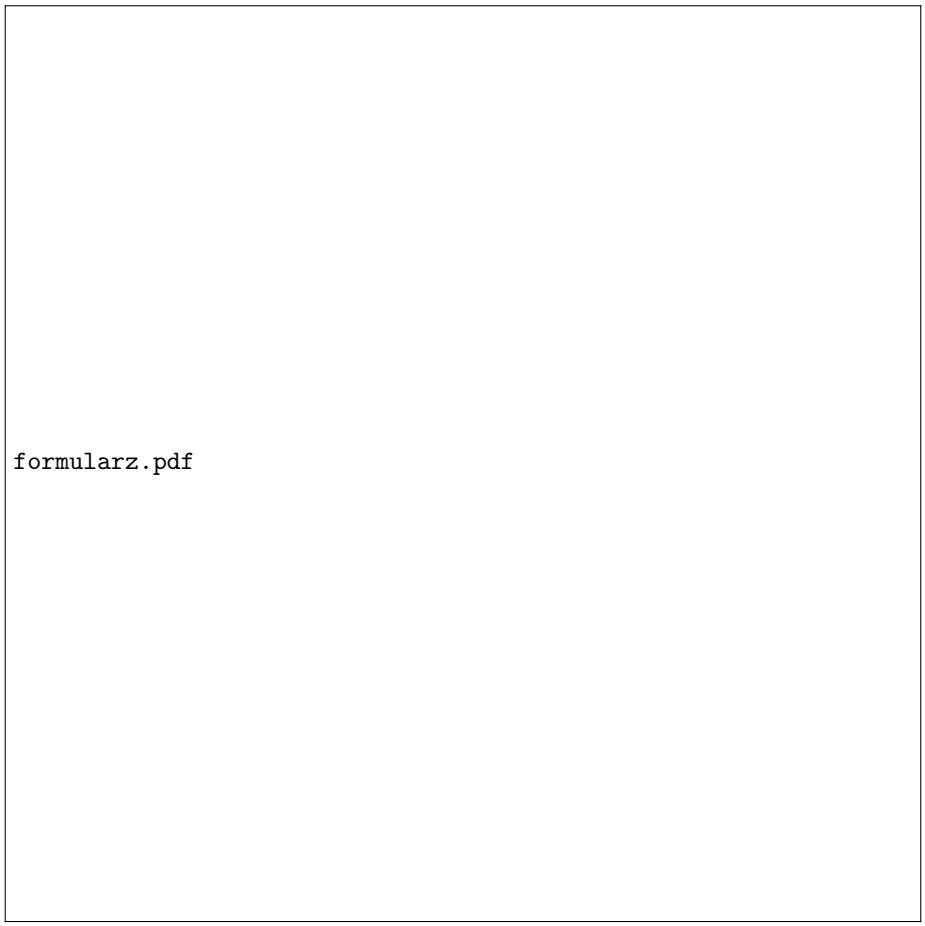
4. Projekt architektury

Do zarządzania projektem będziemy korzystać z edytora kodu Visual Studio Code, systemu kontroli wersji Git w serwisie GitHub. Uruchamianie systemu na serwerze lokalnym przeprowadzimy za pomocą narzędzia Docker i kontenerów. Bazy danych oraz funkcjonującą stronę internetową będziemy przechowywać na zakupionych wcześniej serwerach. Na osobnym serwerze będziemy przechowywać stale aktualizowaną kopię zapasową bazy danych.

4.1. Warstwa zewnętrzna

Interfejs użytkownika zostanie przygotowany w języku JavaScript, HTML i CSS za pomocą technologii jQuery, Apache Tapestry 5/JSF2 J2EE i Spring MVC.

Rysunek 2. Formularz płatności



Projekt wyglądu strony oraz graficzne elementy interfejsu wykonamy w programie Adobe Illustrator.

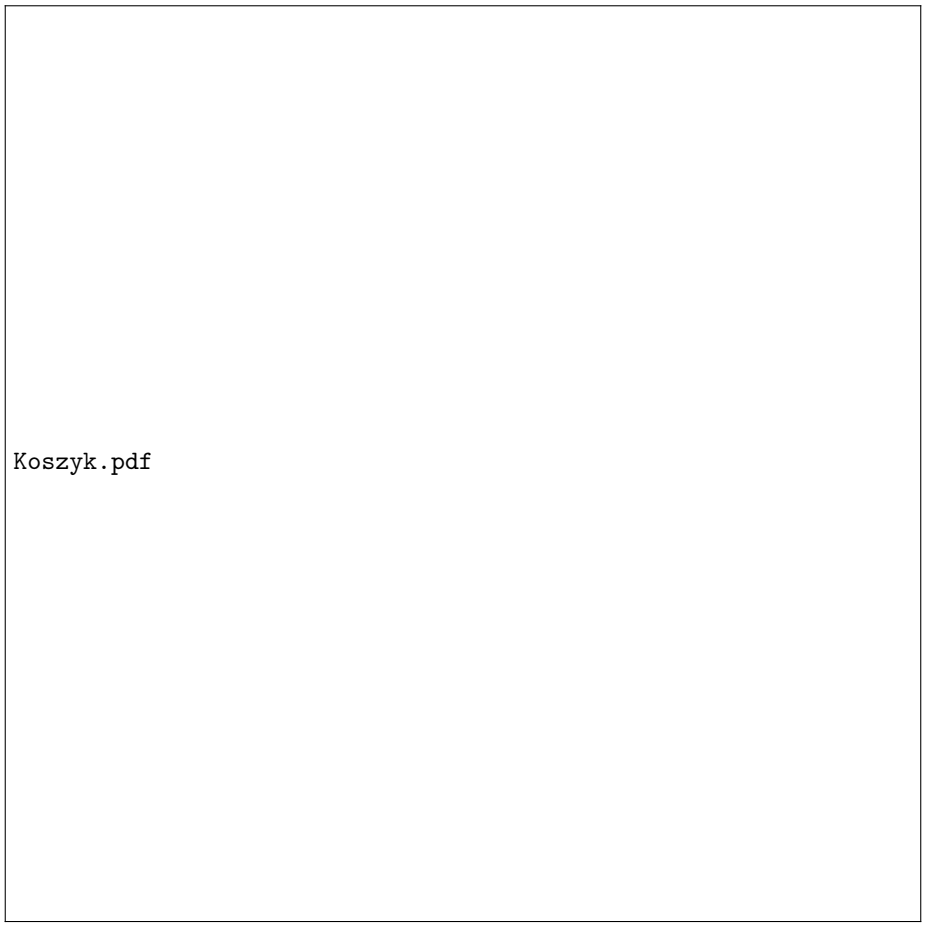
4.2. Warstwa wewnętrzna

Zostanie napisana w języku Python z wykorzystaniem dodatkowych bibliotek oraz rusztowań (ang. *framework*). Zarządzanie bezpieczeństwem systemu będzie działać na podstawie technologię Django. Do obsługi chmury i połączenia interfejsu z serwerami wykorzystamy Dockera oraz FastAPI.

4.3. Baza danych

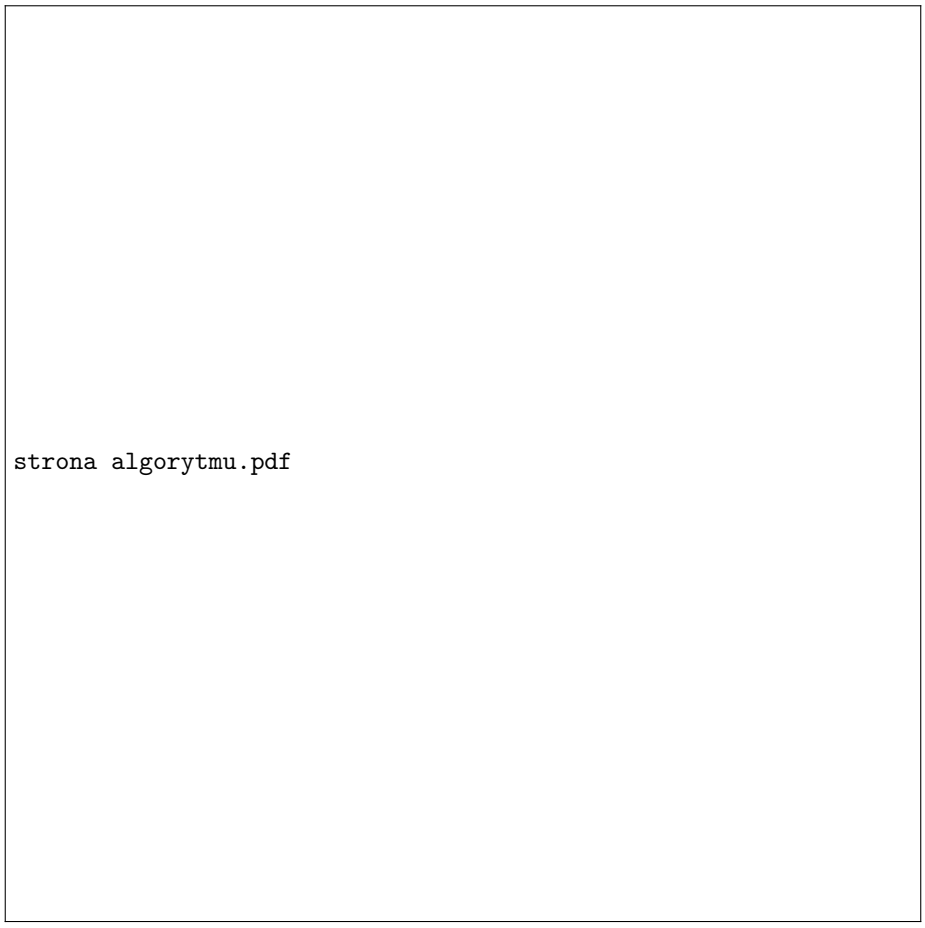
Wykorzystamy otwartoźródłowy, relacyjny system zarządzania bazami danych MySQL. Przechowywane będą na nich dane o użytkownikach, biblioteki algoryt-

Rysunek 3. Podgląd koszyka



mów, zawartość sklepu społeczności. Ze względów bezpieczeństwa damy pełny dostęp do bazy danych jedynie zespołowi zarządzającego jej bezpieczeństwem.

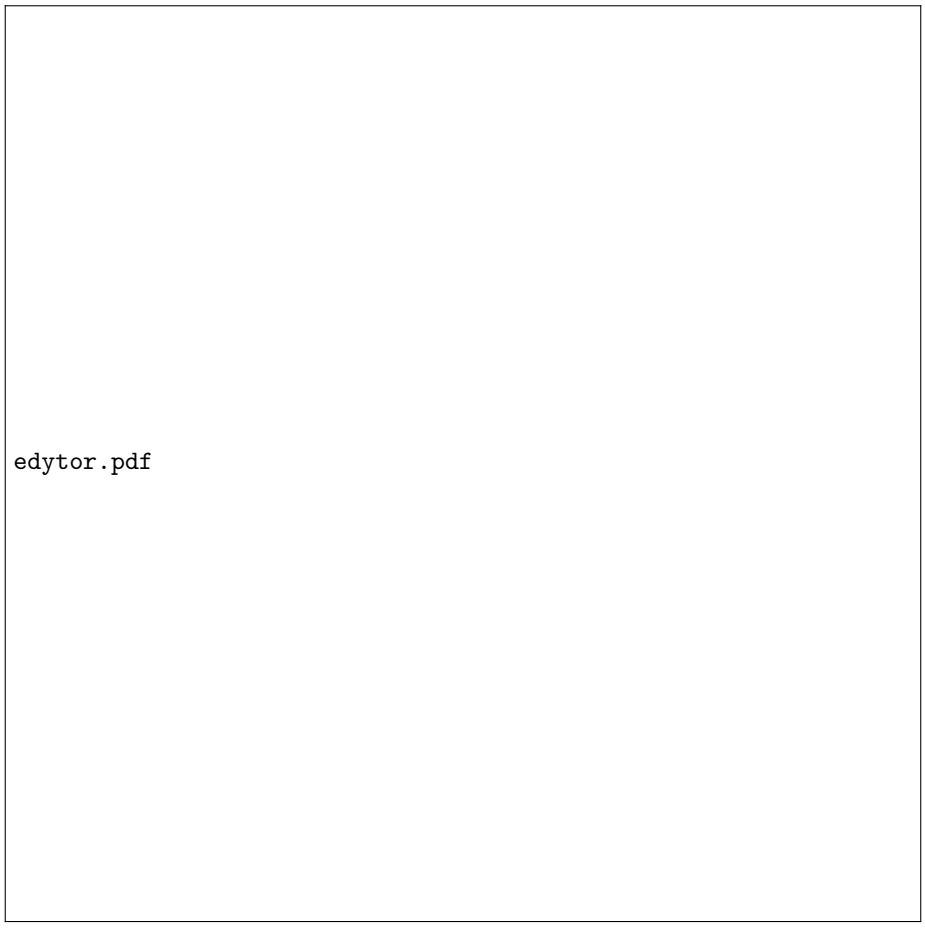
Rysunek 4. Karta wybranego algorytmu



4.4. Schemat bazy danych

4.5. Model konceptualny

Rysunek 5. Edytor algorytmów

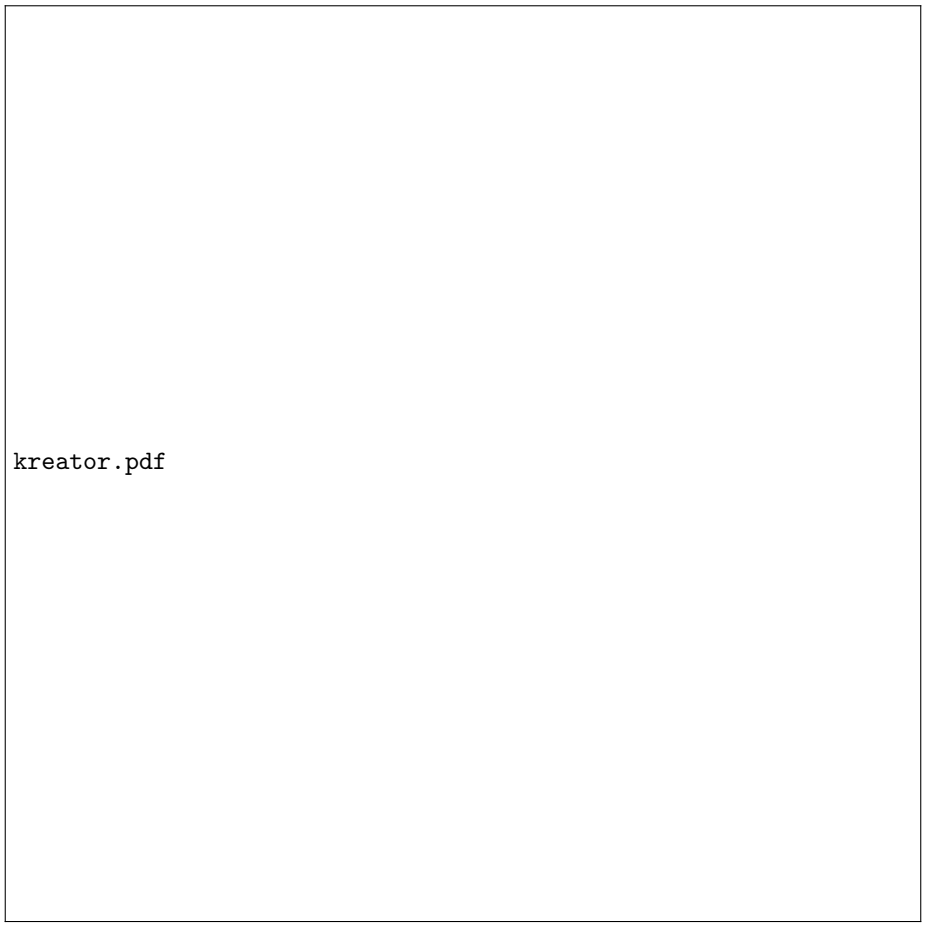


5. Główne zasady kodowania

Warstwa wewnętrzna naszej aplikacji zostanie napisana w języku Python, interfejs użytkownika w JavaScript . Programiści będą przestrzegać zasad opisanych w Google Python Style Guide oraz Google JavaScript Style Guide. Zastosujemy także następującą konwencję:

- każda funkcja powinna być co najwyżej 3-argumentowa,
- każda funkcja powinna zawierać co najwyżej 25 linii kodu,
- między kolejnymi definicjami i deklaracjami powinny się znajdować dokładnie 2 puste wiersze,
- łączna długość wiersza nie powinna przekraczać 100 znaków,

Rysunek 6. Kreator wzorów



- łączna długość kodu w pliku nie powinna przekraczać 100 linii.

Projekt jest przechowywany w repozytorium w serwisie GitHub, zawiera gałąź główną *master* oraz gałąź rozwojową *develop*.

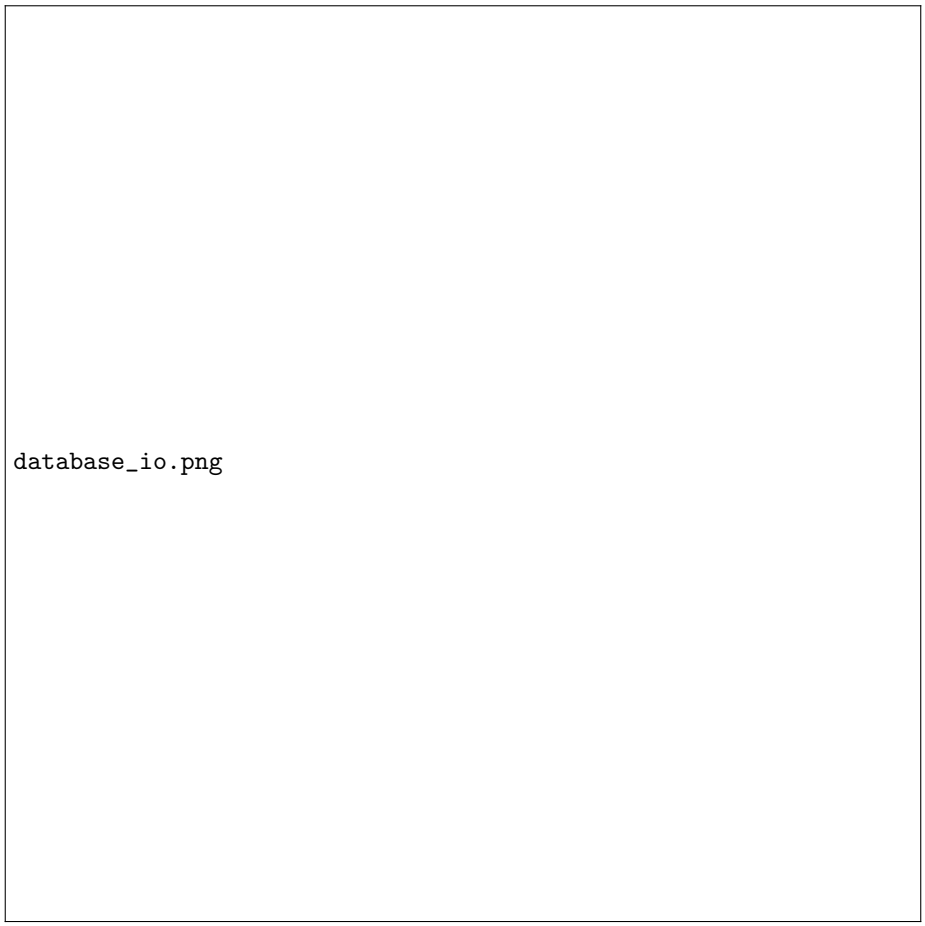
6. Identyfikacja i zasady zarządzania ryzykiem

6.1. Problemy z optymalizacją działania aplikacji

Komplikacja algorytmów o dużej złożoności obliczeniowej może skutkować znacznym spadkiem wydajności aplikacji i długim czasem oczekiwania na wyświetlenie zmian w oknie podglądowym kreatora.

Jest to poważny problem, mogący irytować użytkowników. Rozwiązaniem tego problemu jest wprowadzenie dodatkowego, działającego w tle algorytmu,

Rysunek 7. Schemat bazy danych



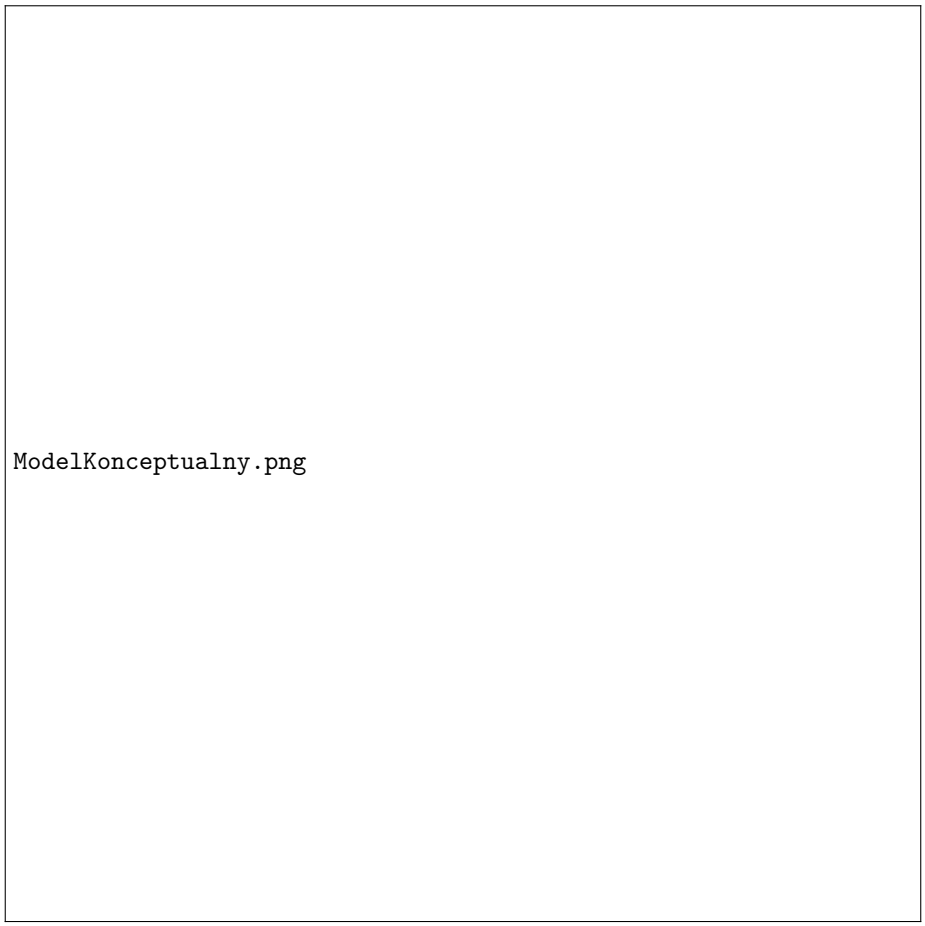
sprawdzającego kod użytkownika pod kątem ewentualnych wymagających obliczeniowo poleceń. W razie wystąpienia takiej sytuacji algorytm wyświetlałby ostrzeżenie o możliwym długim czasie oczekiwania lub, np. w przypadku wykrycia nieskończonej pętli, przerywałby działanie algorytmu, informując o problemie.

6.2. Problemy z bezpieczeństwem bazy danych

W czasie funkcjonowania aplikacji może pojawić się ryzyko ataku na jej serwery oraz próby wykradzenia danych.

Jest to poważny problem, którego wystąpienie należy ograniczyć przez wynajęcie zespołu nadzorującego bezpieczeństwo serwerów i baz danych oraz kompleksowe testowanie ich w czasie produkcji.

Rysunek 8. Model konceptualny



7. Ocena zgodności prac

Koncepcja wykonania naszego systemu jest wierna opisowi zawartemu w tablicy koncepcyjnej oraz dokumencie ze specyfikacją wymagań. Obecna analiza projektu nie wskazuje na konieczność wprowadzania zmian w dokumentacji.