SYSTEM WSPOMAGAJĄCY PROJEKTOWANIE WNĘTRZ – OPTYMALIZUJĄCY ROZKŁAD PROSTOKĄTÓW NA PŁASZCZYŹNIE

Dokumentacja (raport z projektu)

Piotr Paturej, Miłosz Kasak, Jakub Polak, Kamil Szydłowski

Spis treści

Specyfikacja wymagań

Niniejsza specyfikacja wymagań została sporządzona w konwencji MoSCoW, tzn. ilekroć zawiera frazę:

1. ***Oprogramowanie musi*** – oznacza to, że dane wymaganie zostanie zawarte w finalnym rozwiązaniu pod groźbą uznania projektu za porażkę,
2. ***Oprogramowanie powinno*** – oznacza to, że dana pozycja ma wysoki priorytet i będzie zawarta w finalnym rozwiązaniu o ile, to możliwe,
3. ***Oprogramowanie może*** – oznacza to, że dane wymaganie ma średni priorytet i zostanie dostarczone tylko i wyłącznie wtedy, gdy pozwolą na to czas i zasoby,
4. ***Oprogramowanie nie będzie*** – oznacza to, że dana pozycja nie zostanie zaimplementowana w danej wersji oprogramowania, ale możliwe będzie dodanie jej w przyszłości – kwestia jej dołączenia pozostaje otwarta.

Wymagania biznesowe - WB

Oprogramowanie musi:

1. Umożliwiać automatyczne projektowanie wnętrz.
2. Ograniczać czynności w procesie projektowania wnętrz dla dużej liczby różnych ścian.

Oprogramowanie powinno:

3. Ułatwiać projektowanie dużej liczby ścian o takich samych wymiarach i cechach wspólnych.

Oprogramowanie może:

4. Dostarczać narzędzia używane na zajęciach technicznych w szkole do przedstawienia idei procesu projektowania wnętrz.

Wymagania użytkownika - WU

Oprogramowanie musi:

1. Zapewnić intuicyjny i prosty interfejs do projektowania wnętrz – WB1, WB2.
2. Zawierać mechanizmy do automatycznego projektowania wnętrz – WB1.
3. Zapewnić mechanizmy ułatwiające proces projektowania wnętrz dla dużej liczby różnych ścian – WB2.

Oprogramowanie powinno:

4. Generować:
   a) taki sam rozkład fotografii dla każdej ściany lub
   b) dla każdej ściany rozkład fotografii, który posiada cechę wspólną z innymi, lub
   c) dla każdej ściany całkowicie inny rozkład fotografii

   w przypadku projektowania dużej liczby ścian o takich samych wymiarach i cechach wspólnych – WB3.

Oprogramowanie może:

5. Zawierać instrukcje i podpowiedzi ułatwiające korzystanie z systemu – WB1, WB2, WB3, WB4.

Oprogramowanie nie będzie:

6. Zapewniać możliwości zdalnego utrwalania postępów pracy.
7. Zawierać mechanizmów do jednoczesnej pracy nad projektem przez kilka osób.

Wymagania systemowe - WS

Wymagania funkcjonalne - WSF

Oprogramowanie musi:

1. Zawierać zbiór przykładowych wielokątów (reprezentujących ściany) – WU1, WU3.
2. Umożliwiać użytkownikowi wybór wielokąta (reprezentującego ścianę) spośród zbioru wbudowanych – WU1, WU3.
3. Pozwalać użytkownikowi wprowadzać cechy reprezentującego fotografię prostokąta (szerokość, wysokość, położenie) na wielokącie (ścianie) – WU1.
4. Rozmieszczać prostokąty (fotografie) na wielokącie (ścianie) w sposób optymalny z wykorzystaniem algorytmu ewolucyjnego (metoda ewolucji różnicowej) z często inicjowanym startem, uwzględniając zadane ograniczenia – WU2, WU3.
5. Umożliwiać pobranie projektu na dysk lokalny użytkownika.
6. Pozwalać użytkownikowi odtworzyć stan projektu poprzez załadowanie pliku z projektem z dysku lokalnego.

Oprogramowanie powinno:

7. Umożliwiać użytkownikowi zakreślanie wewnątrz wielokąta zamkniętego kształtu, w którym prostokąty zostaną umieszczone w pierwszej kolejności – WU1, WU4.
8. Zapewniać mechanizmy do określenia preferowanej odległości między sąsiadującymi prostokątami - WU1, WU4.
9. Pozwalać użytkownikowi (z wykorzystaniem myszki) określić stałą pozycję wybranych przez siebie prostokątów, które nie podlegają automatycznej optymalizacji - WU1, WU4.

Oprogramowanie może:

10. Umożliwiać dodanie rzeczywistej fotografii do prostokąta – WU1.
11. Pozwalać na zmianę koloru prostokąta – WU1.
12. Wyświetlać podpowiedzi i objaśnienia dotyczące funkcjonalności danego elementu po najechaniu na niego myszką – WU5.
13. Zawierać kartę Pomoc z podręcznikiem użytkownika – WU5.

Oprogramowanie nie będzie:

14. Umożliwiać użytkownikowi logowania – WU6, WU7.
15. Pozwalać na jednoczesną pracę zdalną w zespole nad projektem – WU7.
16. Zapewniać mechanizmów do utrwalania projektów w zdalnym repozytorium – WU6.
17. Umożliwiać współdzielenia projektów – WU6, WU7.

Wymagania pozafunkcjonalne - WSPF

Oprogramowanie musi:

1. Zostać zrealizowane w formie systemu zawierającego aplikację internetową.
2. Być zintegrowane z repozytorium kodu.
3. Być hostowane na zewnętrznym serwerze w celu umożliwienia dostępu do niego z dowolnego komputera.
4. Zostać zaprojektowane zgodnie ze znanymi wzorcami architektonicznymi, umożliwiającymi łatwą rozbudowę systemu.
5. Być możliwe do uruchomienia na dowolnym stosie (ograniczonym jedynie technologią bazową).

Oprogramowanie powinno:

6. Zawierać testy jednostkowe

Oprogramowanie może:

7. Umożliwiać szyfrowanie połączenia z wykorzystaniem TLS/SSL.

Oprogramowanie nie będzie:

8. Zawierać bazy danych.

Specyfikacja przypadków użycia

Aktorzy

1. **Projektant wnętrz**

Biznesowe przypadki użycia - PB

1. **Projektowanie wnętrza**
   a) **scenariusz główny:**
      1) Projektant wybiera kształt ściany spośród zbioru dostarczonych.
      2) Projektant dodaje reprezentujące fotografie prostokąty.
      3) Projektant edytuje właściwości (m.in. rozmiar) dodanych prostokątów.
      4) Projektant rozmieszcza samodzielnie wybrane prostokąty na ścianie.
      5) Projektant zakreśla obszar, w którym prostokąty mają zostać rozmieszczone w pierwszej kolejności.
      6) System rozmieszcza w sposób optymalny i uwzględniający zadane ograniczenia prostokąty na płaszczyźnie.
      7) Projektant odczytuje wizualnie rozmieszczenie prostokątów na płaszczyźnie.
      8) Projektant zapisuje stan pracy do pliku i pobiera go do lokalnego repozytorium.

b) scenariusz alternatywny – praca z zapisanym wcześniej projektem:

1) System wczytuje stan pracy z lokalnie ładowanego pliku.
2) Projektant zmienia kształt ściany.
3) Projektant dodaje nowe prostokąty.
4) Projektant zmienia przy użyciu myszki rozmieszczenie wybranych prostokątów na ścianie.
5) Projektant zmienia kształt obszaru, w którym prostokąty mają być zakreślone w pierwszej kolejności.
6) System rozmieszcza w sposób optymalny i uwzględniający zadane ograniczenia prostokąty na płaszczyźnie.
7) Projektant odczytuje wizualnie rozmieszczenie prostokątów na płaszczyźnie.
8) Projektant zapisuje stan pracy do pliku i pobiera go do lokalnego repozytorium.

Systemowe przypadki użycia – FU

1. Projektowanie wnętrza

Wspiera procedurę: PB1

Korzysta z: FU4, FU5, FU8, FU9

Aktorzy: Projektant wnętrz

a) scenariusz główny:

1) Wybór ściany za pomocą funkcji FU4.
2) Dodanie prostokątów za pomocą funkcji FU5.
3) Zakreślenie obszaru do rozmieszczenia prostokątów w pierwszej kolejności za pomocą funkcji FU8.
4) Rozmieszczenie prostokątów na płaszczyźnie za pomocą funkcji FU9.

2. Zapis projektu

Wspiera procedurę: PB1

Rozszerza funkcję: FU1 po kroku 4.

Aktorzy: Projektant wnętrz

a) scenariusz główny:

1) Projektant klika na przycisk ,,Zapisz projekt”
2) System uruchomia automatyczne pobieranie projektu.

3. Odczyt projektu

Wspiera procedurę: PB1

Rozszerza funkcję: FU1 przed krokiem 1.

Aktorzy: Projektant wnętrz

a) scenariusz główny:

1) Projektant klika na przycisk ,,Wczytaj projekt”
2) System wyświetla osobne okno dialogowe do wybrania pliku.
3) Projektant wskazuje plik z projektem.
4) System zamyka wyświetlone okno dialogowe i wyświetla stan projektu w polu roboczym

4. Wybór ściany

Wspiera procedurę: PB1

Aktorzy: Projektant wnętrz

a) **scenariusz główny**:
   1) Projektant klika na przycisk ,,Ściany”
   2) System wyświetla dostępne kształty ścian w osobnym oknie.
   3) Projektant wybiera jeden kształt.
   4) System zamyka wyświetlone okno do wyboru ściany.
   5) System wyświetla w polu roboczym wybrany kształt.

5. Dodanie prostokąta

Wspiera procedurę: PB1

Aktorzy: Projektant wnętrz

a) scenariusz główny:
   1) Projektant klika na przycisk ,,Prostokąty”
   2) System wyświetla osobne okno z domyślnymi parametrami prostokąta.
   3) Projektant zatwierdza utworzenie nowego prostokąta.
   4) System dodaje prostokąt do zbioru prostokątów na dolnym pasku roboczym.

6. Umieszczenie prostokąta na ścianie

Wspiera procedurę: PB1

Rozszerza funkcję: FU5 po kroku 4.

Aktorzy: Projektant wnętrz

a) scenariusz główny:
   1) Projektant przy pomocy myszki przesuwa prostokąt z dolnego paska do obszaru roboczego na wybraną pozycję.
   2) System wyświetla dany prostokąt w odpowiednim miejscu bez modyfikacji pozostałych elementów.

7. Zakreślenie obszaru do rozmieszczenia prostokątów w pierwszej kolejności.

Wspiera procedurę: PB1

Korzysta z: FU4

Aktorzy: Projektant wnętrz

a) scenariusz główny
   1) Projektant klika na przycisk ,,Zakreśl obszar”
   2) Projektant przy użyciu myszki zakreśla wielokąt wypukły jako obszar.
   3) System wyświetla w polu roboczym zakreślony obszar.

b) scenariusz alternatywny – brak wybranej ściany:
   1) Wybór ściany za pomocą funkcji FU4
   2) Kroki 1-3 ze scenariusza głównego.

8. Rozmieszczenie prostokątów na płaszczyźnie

Wspiera procedurę: PB1

Korzysta z: FU5 i FU8

Aktorzy: Projektant wnętrz

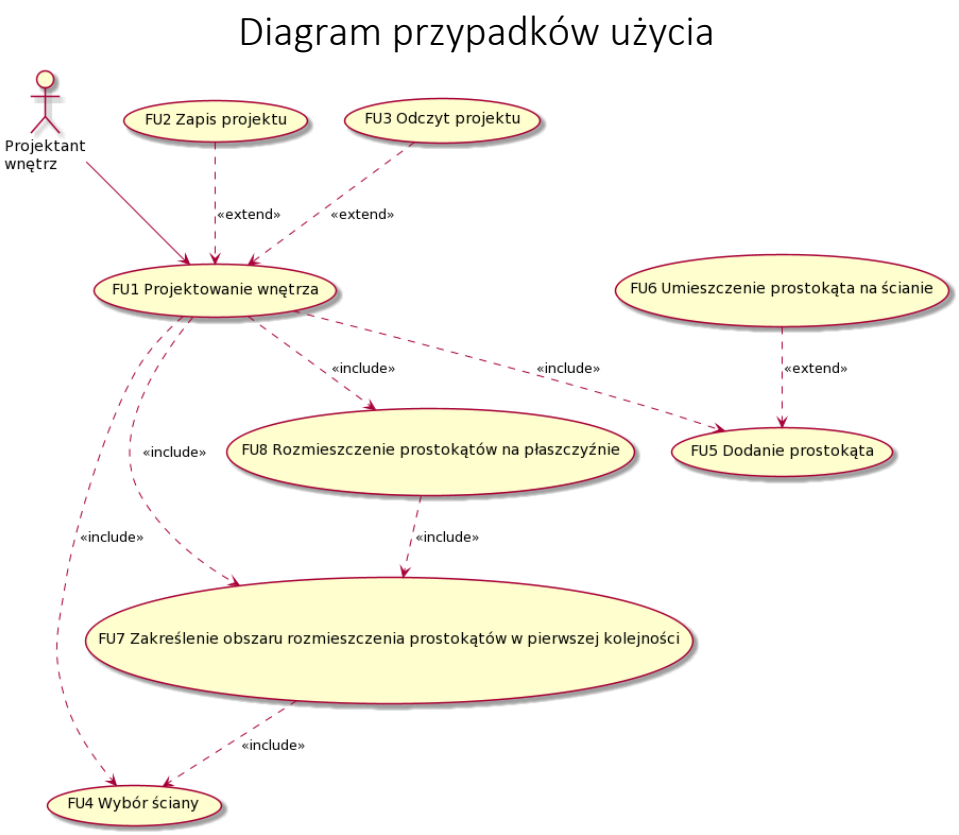
a) scenariusz główny:
   1) Projektant klika na przycisk ,,Rozmieść”
   2) System oblicza optymalne rozmieszczenie prostokątów na ścianie, uwzględniając zadane ograniczenia
   3) System wyświetla wizualizację rozmieszczenia prostokątów na ścianie.

b) scenariusz alternatywny – brak prostokątów

1) Wybierz prostokąty za pomocą funkcji FU5.
2) Kroki 1-3 ze scenariusza głównego

c) scenariusz alternatywny – brak zakreślonego obszaru

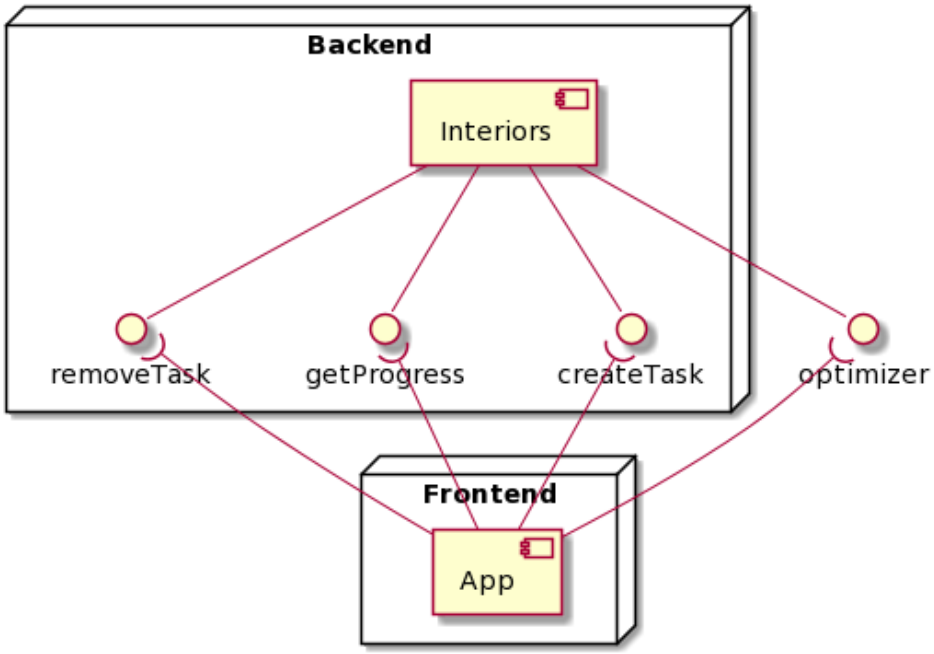
1) Zakreśl obszar za pomocą funkcji FU8.
2) Kroki 1-3 ze scenariusza głównego



Rys. 1 Diagram systemowych przypadków użycia

Definicja architektury

Plan struktury systemu – model komponentów



Rys. 2 Diagram komponentów systemu

Zastosowane szablony architektoniczne

1. Backend – MVC
2. Interiors – monolit
3. Frontend – monolit
4. App – monolit
5. Backend i Frontend – mikroserwis

Kluczowe elementy struktury i ich interfejsy

1. *Backend* – serwer napisany w Pythonie z użyciem Django. Zawiera implementację algorytmu ewolucji różnicowej służącego do optymalizacji rozkładu prostokątów na płaszczyźnie, który udostępnia poprzez interfejs */optimizer/*
2. *Frontend* – serwer plików statycznych napisany w Java Script z użyciem React.js. Stanowi webową aplikację kliencką
3. */createTask/* - interfejs backendu do stworzenia zadania optymalizacji w celu monitorowania jego statusu
4. */optimizer/* - interfejs backendu do optymalizacji rozkładu prostokątów na płaszczyźnie
5. */getProgress/* - interfejs backendu do przesyłania informacji o obecnym statusie danego zadania optymalizacji
6. */removeTask/* - interfejs backendu do usunięcia danego zadania optymalizacji

Interakcje pomiędzy elementami

Aplikacja webowa *Frontend* korzysta z powyżej opisanych interfejsów dostarczanego przez *Backend.*

Wyjaśnienie istoty przyjętych rozwiązań, odniesienie do wymagań

System został zaprojektowany w formie dwóch komponentów: (1) klienckiej aplikacji webowej oraz (2) serwera backendowego udostępniającego interfejs umożliwiający optymalizację rozkładu prostokątów na płaszczyźnie, co wynika z:

1. Wymagania, aby oprogramowanie zostało zrealizowane w formie systemu;
2. Oddzielenia warstwy prezentacji od warstwy logiki biznesowej;
3. Umożliwienia łatwego tworzenia aplikacji klienckich w innych technologiach bez konieczności reimplementacji warstwy logiki (*Backend*)

Określenie podstawowych mechanizmów technicznych

Serwer aplikacyjny

Azure App Service – usługa typu Platform-as-a-Service (PaaS).

Mechanizmy zarządzania

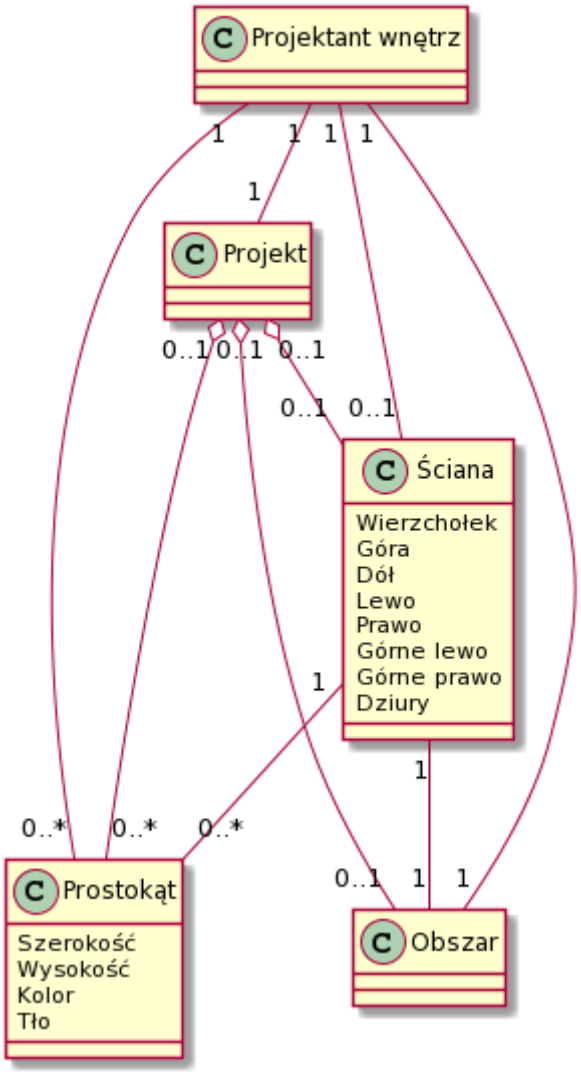
Azure App Service

Mechanizmy bezpieczeństwa

Szyfrowanie połączenia SSL (protokół https).

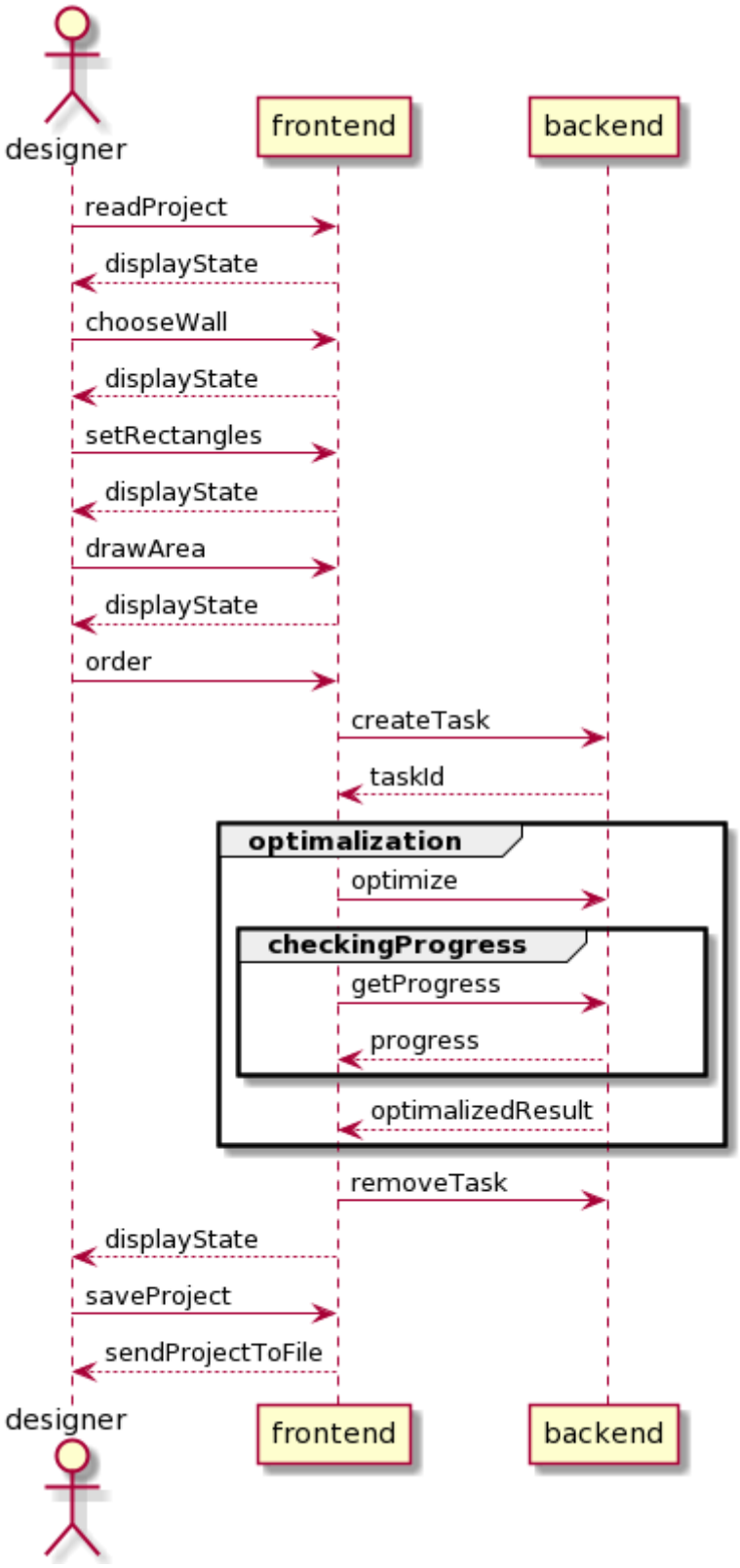
Specyfikacja analityczna

Model dziedziny



Rys. 2 Model dziedziny

Specyfikacja realizacji przypadków użycia – diagramy sekwencji



Rys. 4 Diagram sekwencji

Specyfikacja projektowa

Metody realizacji

Języki programowania

1. Backend – Python 3
2. Frontend – JavaScript

Frameworki

1. Backend – Django
2. Frontend – React.js

Środowisko programowania, uruchamiania, testowania, wdrażania

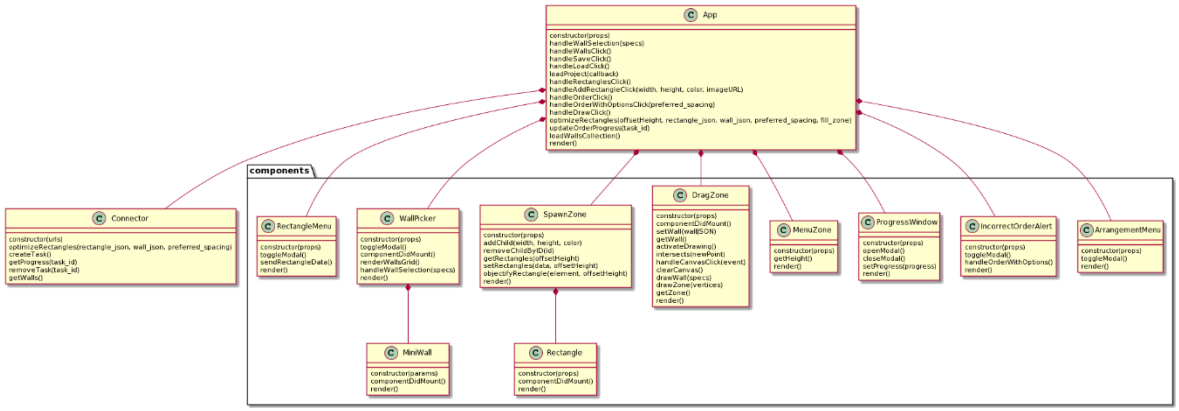
Visual Studio Code, Docker, Python, NPM, Azure Container Registry

Środowisko ciągłej integracji

Gitlab CI/CD z wykorzystaniem Gitlab Runner uruchomionego na maszynie wirtualnej w usłudze Azurze.

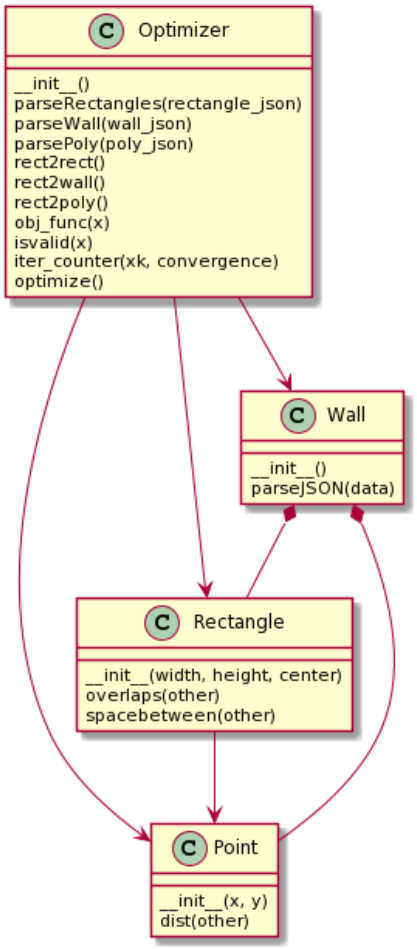
Model obiektowy systemu

Frontend



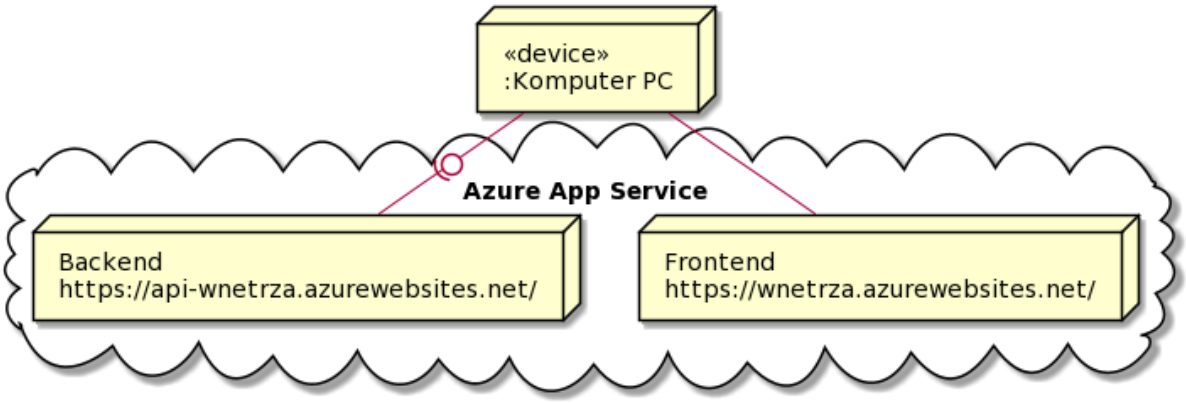
Rys. 5 Diagram obiektowy aplikacji klienckiej (Frontend)

Backend



Rys. 6. Diagram obiektowy optymalizatora rozkładu prostokątów na płaszczyźnie (Backend)

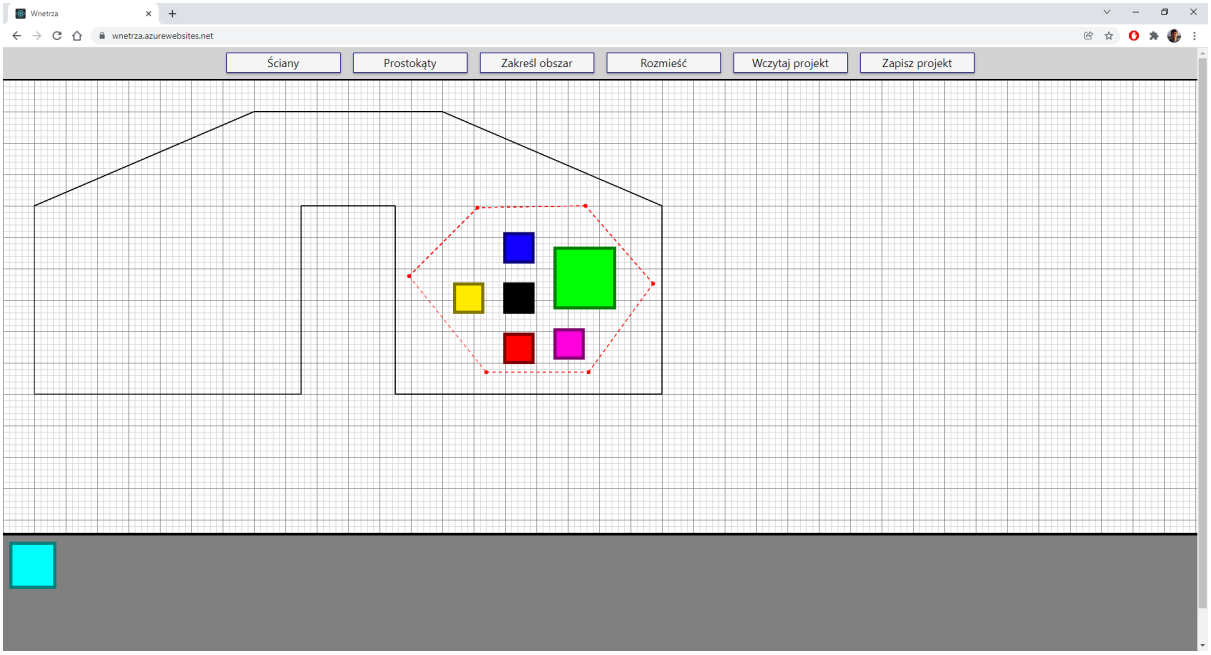
Model struktury systemu (diagram wdrożenia)



Rys. 7 Diagram wdrożenia systemu *Wnętrza*

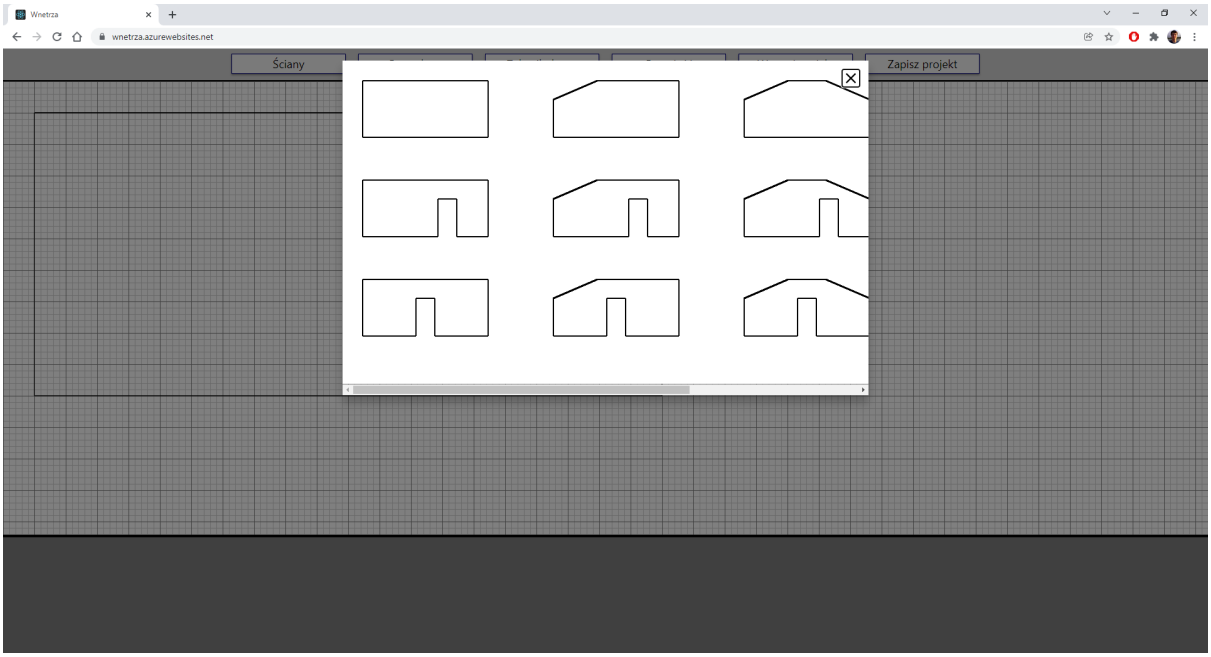
Interfejs użytkownika

Ekran główny



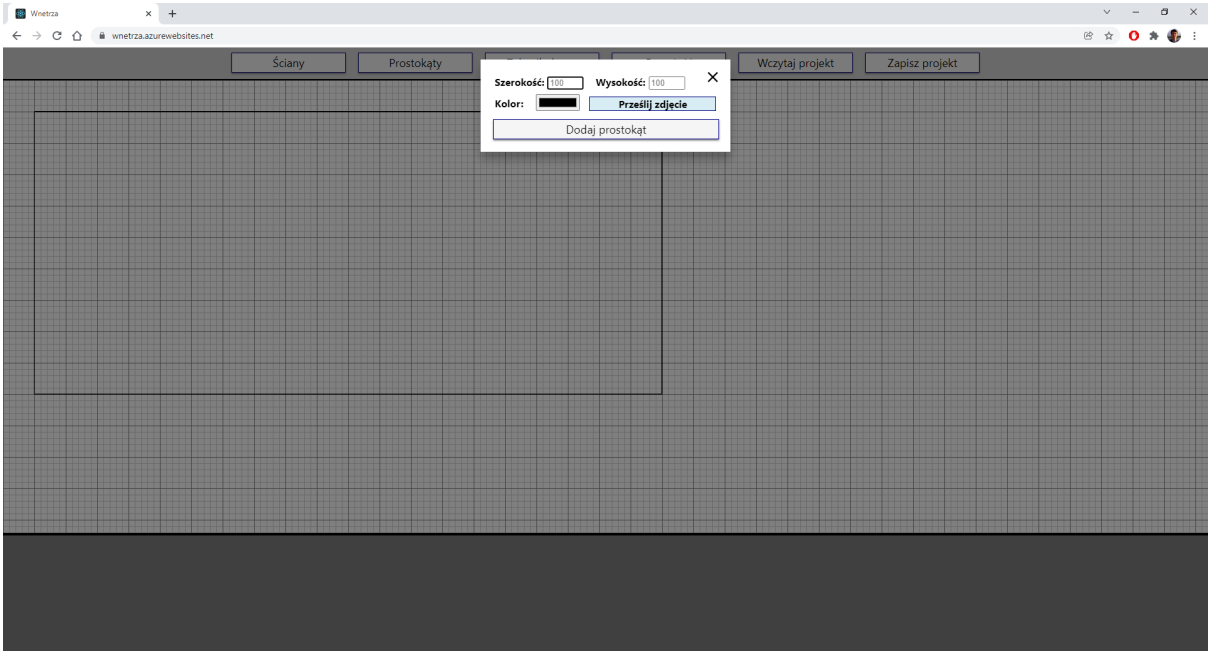
Rys. 8 Ekran główny aplikacji

Ekran wyboru ścian



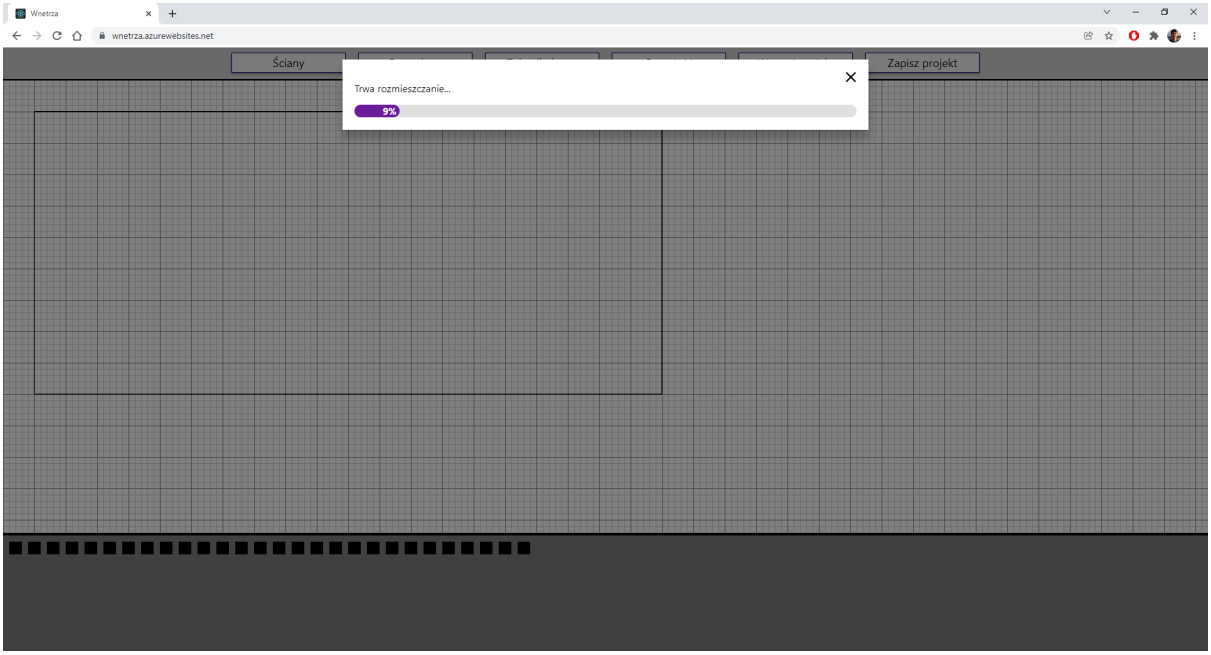
Rys. 9 Ekran wyboru ścian

Ekran umożliwiający dodanie nowego prostokąta z zadanymi parametrami



Rys. 9 Ekran umożliwiający dodanie nowego prostokąta z zadanymi parametrami

Ekran z prezentacją stanu optymalizacji rozkładu prostokątów na płaszczyźnie



Rys. 10 Ekran z prezentacją stanu optymalizacji rozkładu prostokątów na płaszczyźnie

Podręcznik użytkownika

Instrukcja użycia funkcjonalności systemu

Dodanie ściany

Należy kliknąć na przycisk “Ściany” i wybrać jedną z dostępnych ścian poprzez kliknięcie obrazka, na którym jest przedstawiona. Menu ścian można przesuwać w prawo i w lewo.

Dodanie prostokątów

Należy kliknąć przycisk “Prostokąty”, ustawić pożądany rozmiar i kolor prostokąta lub zdjęcie, które będzie wyświetlane w tle prostokąta. Nieustawienie tych danych spowoduje wybranie danych domyślnych. Następnie należy wcisnąć przycisk “Dodaj prostokąt”. Prostokąt o podanych parametrach pojawi się na pasku na dole serwisu.

Zakreślanie obszaru

Należy kliknąć przycisk “Zakreśl obszar” i narysować na ścianie wielokąt, poprzez klikanie na punkty leżące na ścianie, które będą wierzchołkami wielokąta. By zakończyć rysowanie, trzeba utworzyć połączenie między ostatnim punktem, a punktem początkowym.

Rozmieszczanie prostokątów

Prostokąty można rozmieścić ręcznie poprzez przeciągnięcie ich w preferowane miejsce na ścianie. Prostokąty, które mają zostać rozmieszczone automatycznie powinny znajdować się na pasku na dole strony. Po zakreśleniu obszaru należy wcisnąć przycisk “Rozmieść” i wpisać (lub zostawić wartość domyślną) preferowaną odległość między prostokątami, a następnie kliknąć “Rozmieść” w oknie menu. Pokaże się okno z paskiem postępu i algorytm zacznie rozmieszczanie prostokątów.

Zapisywanie i wczytywanie projektu

Żeby zapisać projekt, należy wcisnąć przycisk “Zapisz projekt” i wybrać nazwę pliku i lokalizację zapisu. Aby wczytać projekt, należy kliknąć “Wczytaj projekt” i wybrać lokalizację pliku z projektem.

Podręcznik administratora

Instrukcja budowy systemu z kodu źródłowego

Ze względu na to, że system jest w całości zdockeryzowany, wystarczy w katalogu głównym projektu użyć komendy:

```
docker-compose build
```

W przypadku gdyby zaszła konieczność zbudowania systemu poza środowiskiem docker:

1. Backend: nie wymaga budowy
2. Frontend:

   ```
   npm install
   npm run build
   ```

   Komenda ta spowoduje utworzenie plików statycznych serwera w podkatalogu *build*.

Instrukcja instalacji i konfiguracji systemu

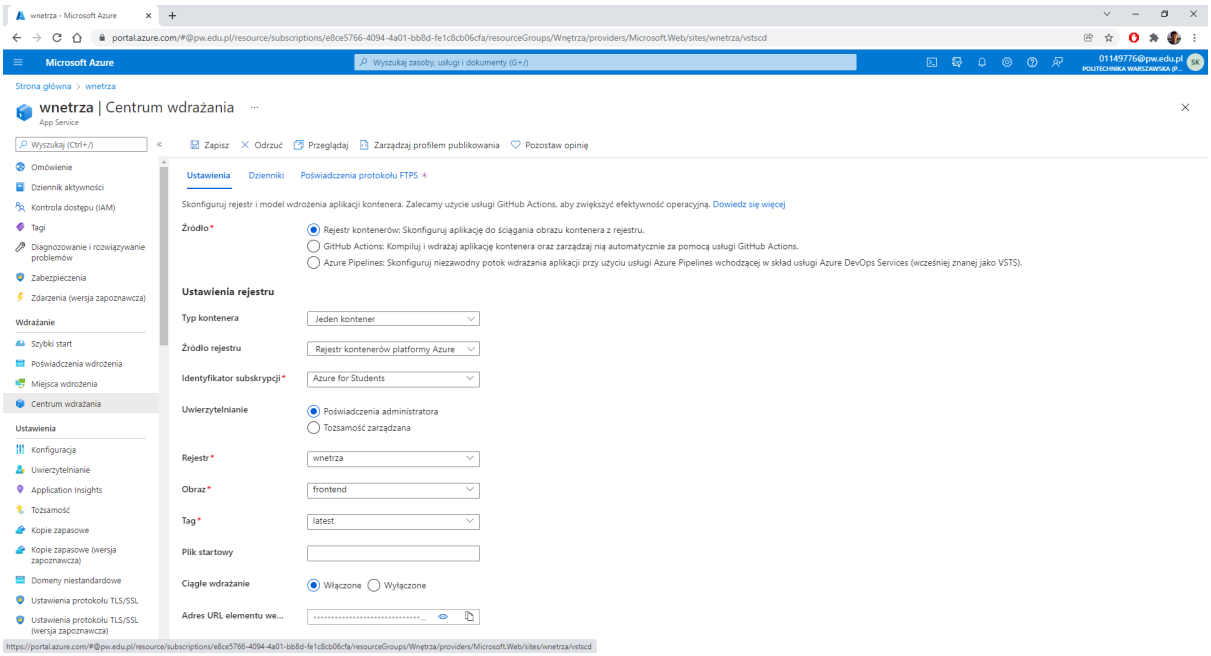
W celu instalacji systemu w usłudze *Azure App Service* najpierw należy tak przygotowane kontenery umieścić w repozytorium artefaktów, np. Azure Container Registry:

```
docker login <registry-name> --username <your-login> --password <your-password>

docker-compose push
```

Następnie należy utworzyć dwie nowe aplikacje w usłudze Azure App Service zgodnie z dokumentacją (https://docs.microsoft.com/en-us/azure/app-service/). Jedną aplikacją będzie część frontendowa, a drugą backendowa.

W dalszej kolejności należy skonfigurować ustawienia wdrażania w obydwu aplikacjach. Na rysunku pokazane są ustawienia dla aplikacji frontendowej, dla backendowej – analogicznie.



Rys. Konfiguracja wdrażania aplikacji frontend w Azure App Service

Tak skonfigurowane aplikacje będą gotowe do pracy.

Instrukcja uruchamiania komponentów systemu

Gdyby zaszła konieczność ręcznego uruchomienia aplikacji poza środowiskiem Azure App Service, można to zrobić wykorzystując program Docker:

```
docker-compose up -d
```

Gdyby zaszła konieczność uruchomienia systemu bez środowiska docker:

1. backend:

   W katalogu backend:

   ```
   pip install -r requirements.txt
   ```

```
python manage.py runserver {adres do uruchomienia}:{port}
```

Adresem do uruchomienia będzie najczęściej loopback (127.0.0.1) albo 0.0.0.0.

2. frontend:

   W katalogu frontend:

```
npm start
```

albo

```
serve -s build -l {port}
```

Instrukcja aktualizacji oprogramowania

Kontenery załadowane do repozytorium w powyższy sposób sprawią, że docelowe aplikacje Azure App Service zaktualizują się automatycznie.