PERT diagram creator

Pawel Blaszczyk

Programowanie Aplikacji Webowych

Projekt Zaliczeniowy

2023/2024

Streszczenie

Dokument opisuje projekt aplikacj webowej przeznaczonej do tworzenia i pracy z diagramami PERT (Program Evaluation Review Technique), które używane są do planowania projektu i wyznaczania w nim ścieżki krytycznej.

W projekcie przedstawiono opis docelowej aplikacji (częściowo wykraczający poza projekt zaliczeniowy), wersję MVP (zakres projektu zaliczeniowego) oraz poszczególne elementy projektu, a w szczególności:

- Plan projektu w postaci historyjek użytkownika
- Architektura oprogramowania (w tym diagramy klas i interacji)
- Opis wybranego stosu technologicznego
- Szczegóły implementacji
- Procedurę wdrożenia aplikacji na platformie Azure

Spis treści

1 Opis aplikacji

Aplikacja ma za zadanie umożliwienie użytkownikowi zbudowania diagramu PERT (Program Evaluation Review Technique) na który składają się kamienie milowe (milestone'y) oraz zadania (task) pomiędzy nimi.

Użytkownik może umieszczać poszczególne kamienie milowe na obszarze roboczym oraz tworzyć połączenia między nimi. Połączenia symbolizują zadania do wykonania. Istnieją dwa specjalne kamienie milowe: rozpoczęcie projektu oraz jego zakończenie.

Dla każdego z połączeń (zadań), użytkownik jest w stanie dodać opis oraz szacowany czas jego wykonania (optymistyczny, najbardziej prawdopodobny i pesymistyczny).

Istnieje również możliwość dodawania „sztucznych" aktywności (t.j. aktywności bez zadań, które pokazują jedynie zależności pomiędzy poszczególnymi kamieniami milowymi)

Na podstawie podanych czasów (oraz wszystkich zależności pomiedzy zadaniami) aplikacja dokonuje obliczeń kiedy każdy z kamieni milowych powinien być osiągnięty.

Aplikacja jest w stanie pokazać ścieżkę krytyczną w projekcie.

Aplikacja jest w stanie określić, czy utworzony diagram nie jest błędny (graf powinien być spójny i acykliczny, a każdy z jego wierzchołków powinien być połączony ścieżką do wierzchołka końcowego)

Istnieje możliwość wyeksportowania diagramu w formacie graficznym (np. SVG). Istnieje również możliwość eksportu diagramu do formatu JSON, który umożliwi późniejszą jego edycję.

Aplikacja zapewnia uwierzytelnianie i autoryzację użytkowników.

Zalogowany użytkownik ma możliwość zapisu diagramu na serwerze oraz późniejsze jego odczytanie.

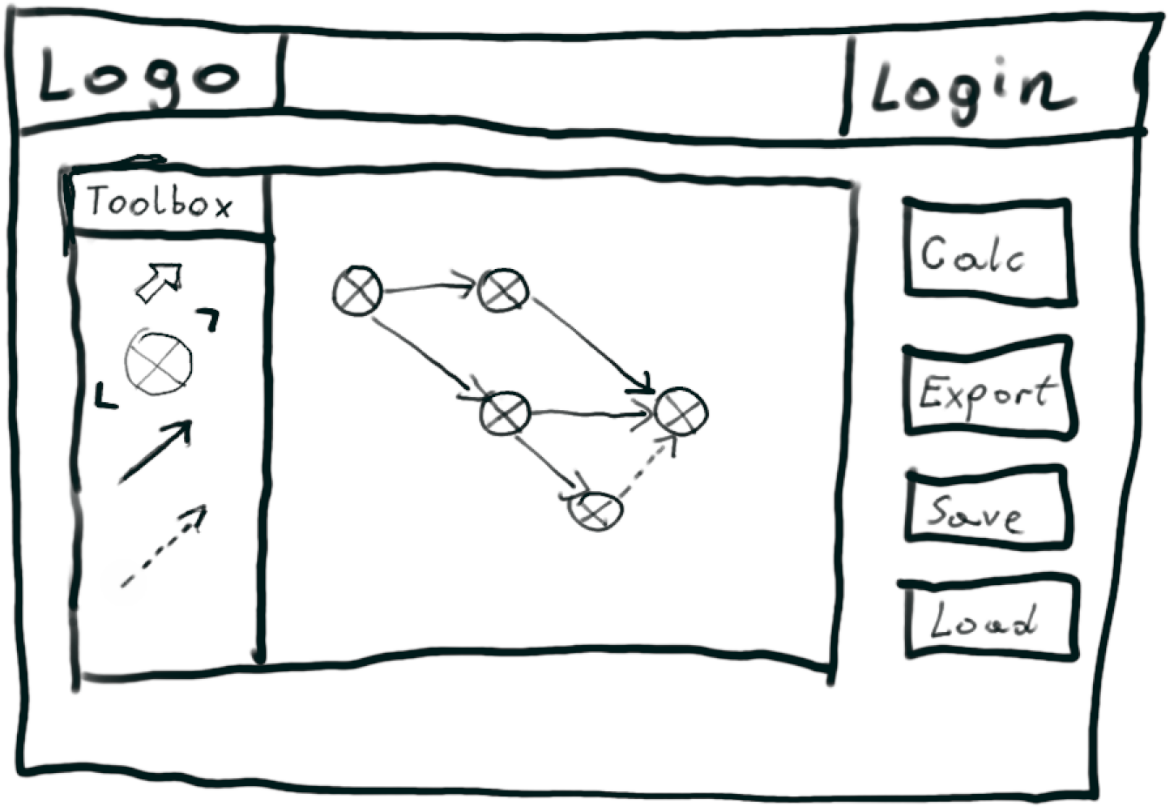
Z przygotowanego diagramu PERT istnieje możliwość wygenerowania odpowiadającego mu diagramu Gantta.

Istnieje możliwość zapisu danej wersji diagramu i jego późniejsza aktualizacja, dzięki czemu jesteśmy w stanie nie tylko dokonywać adaptacji, ale również ocenić jakość naszych estymat w trakcie prowadzenia projektu.

Aplikacja powinna prawidłowo działać we wszystkich nowoczesnych przeglądarkach. Aplikacja jest rozwijana w technologii desktop-first - edycja diagramów na wyświetlaczach telefonów komórkowych nie jest rozważana.

Aplikacja może wspierać pracę na tablecie (zdarzenia dotykowe powinny być obsługiwane).

1.1 Wygląd interfejsu użytkownika



Rysunek 1: Główne okno aplikacji

2 Wymagania

Wymagania projektowe zostaną zebrane w formie mapy historyjek użytkownika.
Kluczowe funkcjonalności aplikacji to

- obsługa widoku strony www
- tworzenie diagramów
- analiza diagramów (np. dokonywanie obliczeń w celu wyznaczenia ścieżki krytycznej)
- generacja dodatkowych widoków (np. diagram Gantta)
- import i eksport diagramów
- obsługa sesji użytkownika (uwierzytelnianie, autoryzacja, zapisywanie diagramów na serwerze)

Każda z historyjek użytkownika powinna być związana z jedną z tych funkcjonalności.
Każda z historyjek ma przypisanego aktora. Zidentyfikowano następujące role:

- programista - przygotowuje aplikację
- autor - buduje diagram
- odbiorca - przegląda diagram
- użytkownik = autor+odbiorca
- właściciel aplikacji - posiada aplikację (może np zechcieć wyświetlać reklamy w celach zarobkowych, czy zbierać dane statystyczne)

2.1 Historyjki użytkownika tworzące „chodzący szkielet”

Jest to minimalna funkcjonalność aplikacji wymagana, żeby uznać ją za użyteczną

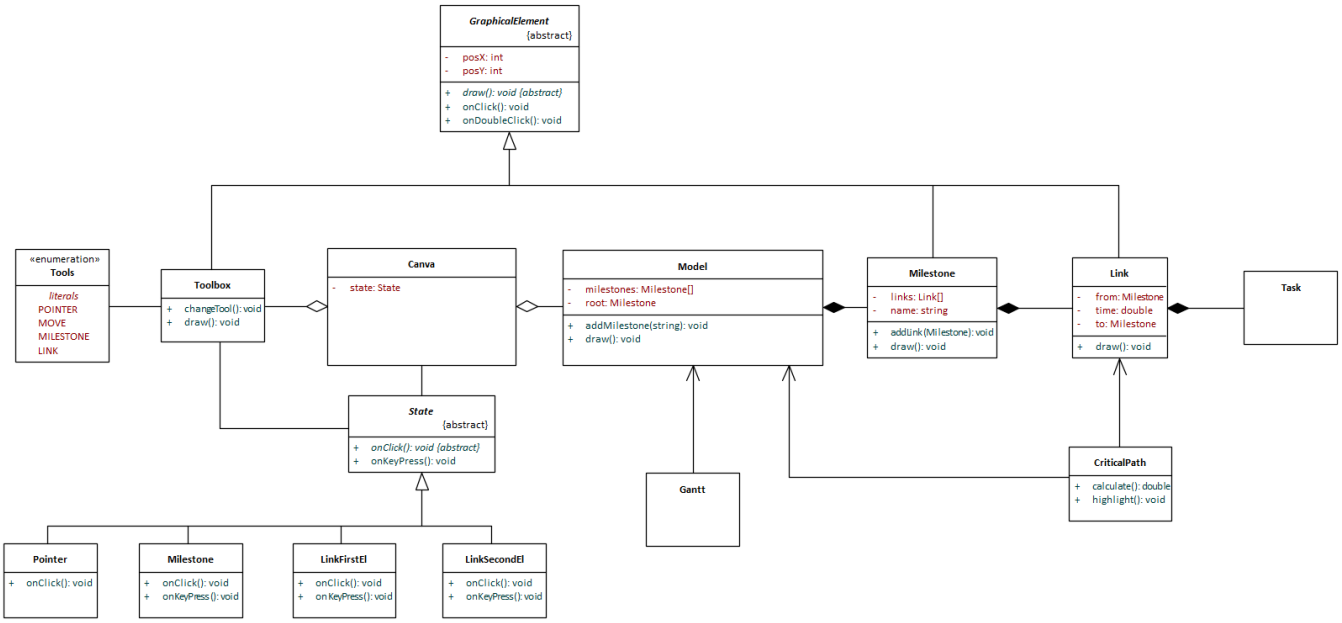
- jako programista chcę mieć zdefiniowany stos technologiczny, żebym wiedział w czym działać
- jako użytkownik chcę mieć dostępny widok strony podzielony na logiczne sekcje, żebym mógł korzystać ze wszystkich funkcjonalności
- jako programista chcę wiedzieć jak wygląda wysokopoziomowa architektura, żebym wiedział w jaki sposób zbudować system
- jako autor chce mieć dostępny toolbox żebym mógł wybrać jaką akcję chcę wykonać
- jako autor chcę rozpoczynać pracę od pustego obszaru roboczego jedynie z początkowym milestonem
- jako autor chcę móc dodać milestone do obszaru roboczego żebym mógł zacząć tworzyć plan projektu
- jako autor chcę móc połączyć dwa milestone’y linią reprezentującą wymaganą pracę, żebym mógł określić zależności czasowe
- jako autor chcę mieć możliwość przesuwania milestone’ów, żebym mógł przearanżować wygląd diagramu
- jako autor chcę móc dodać opis milestone’a, żebym wiedział czego on dotyczy
- jako autor chcę móc edytować nazwę milestone’a żebym mógł poprawić źle wprowadzoną nazwę
- jako autor chcę móc dodać złożoność czasową danego zadania, żebym mógł obliczyć kiedy kolejny milestone może zostać osiągnięty
- jako autor chcę móc edytować złożoność czasową, żeby dostosować plan do aktualnej sytuacji
- jako autor chcę móc dodać puste zależności, żebym mógł podkreślić zależności pomiędzy milestone’ami
- jako odbiorca chcę widzieć obliczone czasy osiągnięcia poszczególnych milestone’ów, żebym wiedział, czy plan projektu jest zgodny z celami biznesowymi

3 Architektura

Na podstawie opisu zidentyfikowano następujące klasy

- User
- Toolbox
- Canva
- Model (diagram PERT)
- Milestone
- Link (połączenie pomiędzy dwoma milestone'ami)
- Task (zadania powiązane z danym linkiem)
- CriticalPath
- Gantt

Rysunek 2 przedstawia wstępną propozycję diagramu klas.



Rysunek 2: Diagram klas (pierwsza iteracja)

Każdy z elementów, który można narysować rozszeraz klasę **GraphicalElement**

3.1 Struktura danych

Obiekt **Model** składa się z listy kamieni milowych (obiekty **Milestone**). Do tego zawiera on wskaźnik na element początkowy (wymagane jest to do obliczania ścieżki krytycznej).

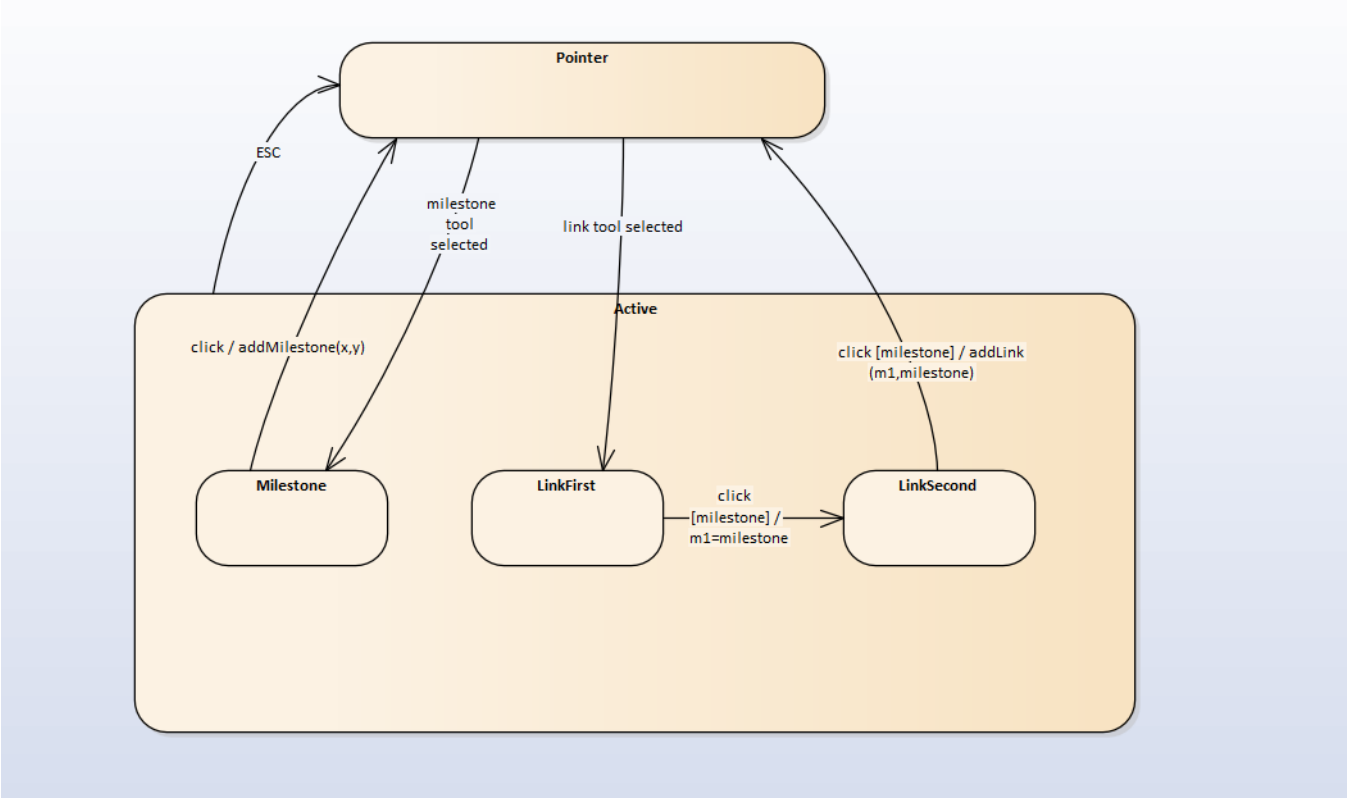
Każdy milestone może być połączony z dowolną ilością innych kamieni milowych, dlatego zawiera on listę obiektów klay **Link**. Połączenia są skierowane od danego obiektu do innego.

Linki muszą być umieszczone zarówno w źródłowym, jak i docelowym milestone'ie, ponieważ przy przemieszczaniu dowolnego z nich, należy każdy link aktualizować

3.2 Maszyna stanu

Toolbox zawiera stan związany z wybranym narzędziem. Najbardziej skomplikowana (jak do tej pory) operacja to połączenie dwóch milestone'ów - musimy rozróżnić, czy wybrany został już pierwszy z nich, czy już wybrano docelowy. Również musimy sprawdzać, czy źródłowy i docelowy milestone to dwa różne obiekty.

Maszyna stanu została przedstawiona na Rysunku 3



Rysunek 3: Maszyna stanu obiektu Toolbox

4 Stos technologiczny

Przy budowie aplikacji użyto następujących technologii

4.1 Frontend

- HTML5
- CSS3
- JavaScipt
- Bootstrap - aspekty wizualne
- Konva.js - rysowanie w canvas
- jQuery - komunikacja z DOM, czy wsparcie AJAX
- Mocha.js, Chai.js, Sinon - testy jednostkowe oraz integracyjne
- Require.js - używanie wielu plikow js i eksport funkcji/klas pomiędzy nimi
- **coś do szablonów stron (Backbone.js?)**

4.2 Połączenie

Połączenie Frontendu z Backendem zrealizowano przy użyciu REST API. Do testów REST API użyto **axios/superagent/request/...**

Zapytania wysyłane są przez frontend przy użyciu technologii AJAX.

4.3 Backend

- Node.js
- Express.js
- **coś do unit testów**