Symulacja pracy z Tablicą Kanban - aplikacja webowa - Sprawozdanie końcowe

Wykonał: Bartosz Zakrzewski
Data: 31.05.2021

Spis treści

1 Podsumowanie projektu

1.1 Tematyka

Projekt miał na celu stworzenie aplikacji webowej, która pozwalałaby na symulację pracy z tablicą Kanban. W takiej symulacji kilku członków zespołu developerskiego pracuje nad projektem, w którym występują różne rodzaje zadań.

1.2 Podsumowanie

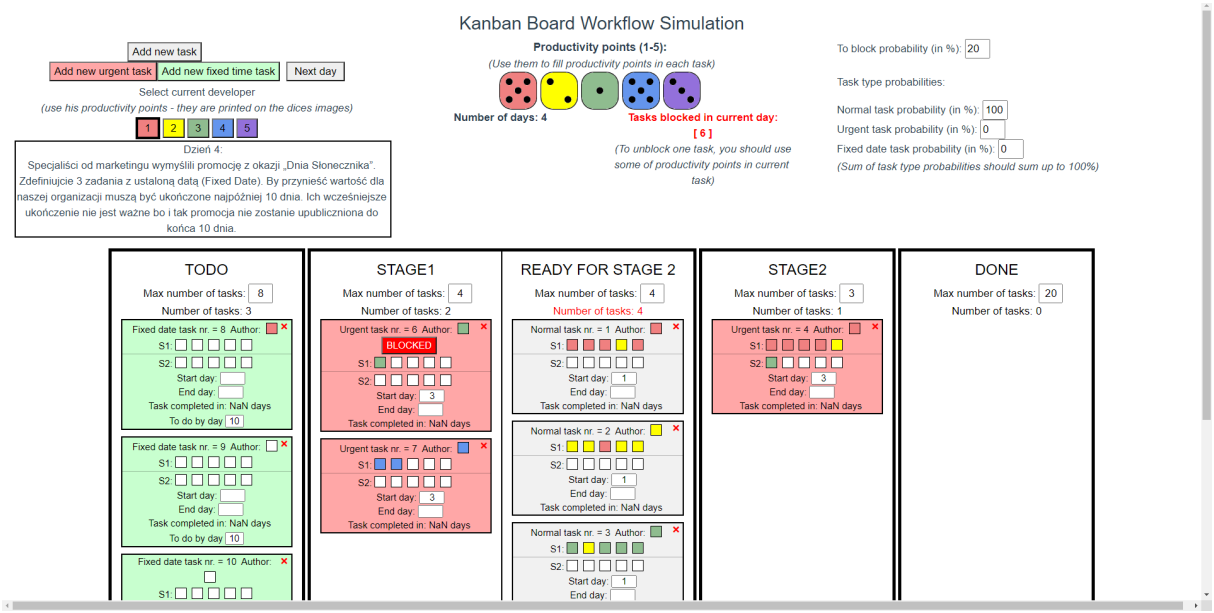
- Udało się wykonać zadaną funkcjonalność.
- Czas trwania projektu to około 2 miesiące (Kwiecień - Czerwiec 2021).
- Aplikacja webowa została stworzona za pomocą frameworka JavaScript Vue.js (i technologii HTML i CSS).
- Aplikacja jest aplikacją Frontend, co oznacza, że nie ma rejestracji, ani możliwości zapisania wyników pracy (Użytkownik może dokonać symulacji i wynik zapisać np. za pomocą zrzutu ekranu).
- Strona została zhostowana za pomocą Github Pages:
  https://zakrzewskib.github.io/KanbanWorkflowSimulation/.

2 Wykonana funkcjonalność

2.1 Zakres pracy

- Definiowanie limitów ilości zadań w poszczególnej kolumnie;
- Oznaczanie wykonanej pracy - wypełnianie progresu wewnątrz danego zadania/taska;
- Mechanizm rzucania kostką 1-5 - jest to generowanie produktywności danego dnia dla danej osoby;
- Generowanie blokerów - daje znać użytkownikowi, że z zadaniem są trudności, Blokera da się usunąć np. wykorzystując punkty progresu;
- Przypisywanie osoby do zadania (oznaczenie kolorystyczne);
- 3 rodzaje taska: Zwykły, Urgent, Fixed Date (nazwa i np. dodatkowo kolorystyka);
- Zadanie ma pole do wpisania dnia początkowego i końcowego;
- Ustalanie prawdopodobieństwa generowania blokerów;
- Symulacja jest oparta na historiach opisujących kolejne dni pracy;

2.2 Wygląd aplikacji



Rysunek 1: Wygląd aplikacji

2.3 Poszczególne elementy aplikacji

2.4 Kolumny

- Jeżeli zadania są w kolumnach TODO lub READY FOR STAGE 2 lub DONE nie mogą być zablokowane.
- W każdej kolumnie jest określona poglądowa maksymalna ilość zadań.
- Zadania można przenosić między kolumnami.

2.5 Zadania

- Są trzy rodzaje zadań: Normal, Urgent i Fixed Date.
- Poszczególne rodzaje zadań są oznaczone kolorami.
- Każde zadanie ma:
  - autora,
  - punkty progresu,
  - dzień rozpoczęcia i zakończenia,

  - czas trwania wykonywania zadania,
  - przycisk usunięcia zadania ('x').
- Dodatkowo zadanie FixedDate ma określone do jakiego dnia trzeba je ukończyć.

2.6 Dodatkowe

- Trzy przyciski dodawania nowego zadania.
- Wybór którego developera punkty zużywamy.
- Kostki do gry pokazujące produktywność każdego z developerów.
- Przycisk rozpoczynający nowy dzień - powoduje on losowanie nowych punktów produktywności, blokowanie losowo zadań i zwiększanie się liczby dni symulacji.
- Informacja jakie zadania zostały zablokowane danego dnia.
- Prawdopodobieństwa zablokowania danego zadania i jaki task zostanie stworzony po naciśnięciu przycisku 'Add new task' (można je zmienić).
- Historie opisujące kolejne dni. Po 10 dniach symulacji nie ma już więcej narracji, aczkolwiek użytkownik może nadal prowadzić symulację.

3 Narzędzia

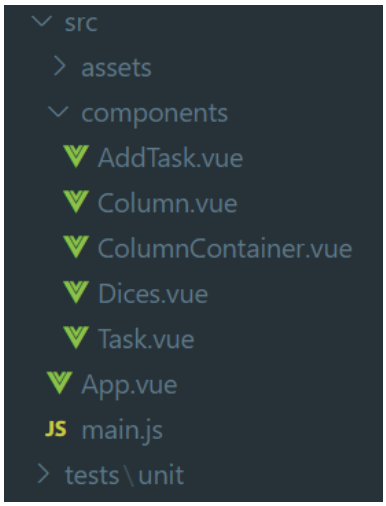
3.1 Środowisko pracy

- IDE - Visual Studio Code 1.56.2
- System kontroli wersji git 2.29.2.windows.3.
- Repozytorium zdalne na stronie github:
  https://github.com/zakrzewskib/KanbanWorkflowSimulation
- Strona została zhostowana za pomocą Github Pages:
  https://zakrzewskib.github.io/KanbanWorkflowSimulation/.

3.2 Struktura projektu

3.3 Komponenty

Aplikacja składa się z komponentów:



Rysunek 2: Komponenty projektu

Aplikacja nie ma wielu komponentów i dodatkowo główny komponent ma dużą liczbę linii - było to prostsze do implementacji w technologii Vue.js.

3.4 Pakiety

npmPackages:

- @vue/babel-helper-vue-jsx-merge-props: 1.2.1
- @vue/babel-helper-vue-transform-on: 1.0.2
- @vue/babel-plugin-jsx: 1.0.4
- @vue/babel-plugin-transform-vue-jsx: 1.2.1
- @vue/babel-preset-app: 4.5.12
- @vue/babel-preset-jsx: 1.2.4
- @vue/babel-sugar-composition-api-inject-h: 1.2.1
- @vue/babel-sugar-composition-api-render-instance: 1.2.4
- @vue/babel-sugar-functional-vue: 1.2.2
- @vue/babel-sugar-inject-h: 1.2.2

- @vue/babel-sugar-v-model: 1.2.3
- @vue/babel-sugar-v-on: 1.2.3
- @vue/cli-overlay: 4.5.13
- @vue/cli-plugin-babel: 4.5.0 => 4.5.12
- @vue/cli-plugin-eslint: 4.5.0 => 4.5.12
- @vue/cli-plugin-router: 4.5.13
- @vue/cli-plugin-unit-jest: 4.5.0 => 4.5.13
- @vue/cli-plugin-vuex: 4.5.13
- @vue/cli-service: ^4.5.13 => 4.5.13
- @vue/cli-shared-utils: 4.5.13 (4.5.12)
- @vue/component-compiler-utils: 3.2.0
- @vue/preload-webpack-plugin: 1.1.2
- @vue/test-utils: ^1.0.3 => 1.2.0
- @vue/web-component-wrapper: 1.3.0
- bootstrap-vue: ^2.21.2 => 2.21.2
- eslint-plugin-vue: ^6.2.2 => 6.2.2
- jest-serializer-vue: 2.0.2
- portal-vue: 2.1.7
- vue: ^2.6.12 => 2.6.12
- vue-eslint-parser: 7.6.0
- vue-functional-data-merge: 3.1.0
- vue-hot-reload-api: 2.3.4
- vue-jest: 3.0.7
- vue-loader: 15.9.6 (16.2.0)
- vue-style-loader: 4.1.3
- vue-template-compiler: ^2.6.11 => 2.6.12
- vue-template-es2015-compiler: 1.9.1

4 Uruchomienie kodu

Jeżeli pobierzemy kod z repozytorium na Githubie (i np. dodamy jakąś funkcjonalność) wynik pracy można zobaczyć za pomocą yarn - menadżera pakietów:

1. yarn install - instalacja yarn'a;
2. yarn serve - aby zobaczyć poglądowo naszą aplikację;

Można to także wykonać za pomocą innego menadżera pakietów - npm:

1. npm install - instalacja npm;
2. npm run serve - zobaczenie aplikacji;

Aby zbudować projekt i zhostować aplikację webową należy:

1. yarn build - utworzy nam się folder dist; (lub npm run build)
2. folder dist jest gotowy na wdrożenie;
3. następnie należy postępować zgodnie z instrukcją na https://cli.vuejs.org/guide/deployment.html#github-pages, aby skorzystać z serwisu Github Pages. W repozytorium pliki potrzebne do użycia Github pages to deploy.sh i vue.config.js.

5 Źródła

- Pomysł na aplikację: https://www.okaloa.com/.
- Zakres projektu został zaprezentowany przez opiekuna projektu: mgr. inż. Krzysztofa Marka.
- Framework Vue.js: https://vuejs.org/