

# Eksperyment frontendowy

<b>Założenia</b>	<b>3</b>
<b>Cel</b>	<b>3</b>
<b>Opis aplikacji</b>	<b>3</b>
<b>User stories</b>	<b>3</b>
<b>Testowane biblioteki</b>	<b>4</b>
<b>Co chcemy sprawdzić w praktyce</b>	<b>4</b>

# Założenia

Front-end platformy ma być odseparowany od back-endu, ma komunikować się przez API. Umożliwi to w przyszłości ewentualną zmianę technologii w jakiej pisany jest front-end. Back-end platformy ma realizować części Model oraz Controller wzorca architektonicznego MVC, natomiast front-end – część View.

W drodze analizy kilku bibliotek/frameworków (Aurelia, Ember, React, Vue) zdecydowano, że eksperyment zostanie przeprowadzony przy wykorzystaniu bibliotek React (wraz z Redux) oraz Vue (z Vuex).

## Cel

Celem eksperymentu JS jest napisanie wybranych funkcjonalności front-endu aplikacji w celu wydania rekomendacji co do używanych bibliotek.

## Opis aplikacji

Aplikacja ma być platformą i udostępniać inne aplikacje.

Użytkownik platformy może:

- utworzyć nowe konto,
- zalogować się do istniejącego konta,
- wylogować się,
- uruchomić jedną z dostępnych na platformie aplikacji.

Platforma posiada ekran logowania/rejestracji.

Po zalogowaniu użytkownik widzi ekran z częścią główną i sidebarem.

W sidebarze wyświetlana jest nazwa użytkownika, przycisk wylogowania oraz posortowana lista „Top 5” aplikacji, które były przez użytkownika najdłużej używane. Czas spędzony przez użytkownika w konkretnej aplikacji jest zliczany.

W głównym oknie platformy wyświetlane są dostępne aplikacje w formie ikon/obrazów wraz z informacją o aplikacji. Po uruchomieniu aplikacji przez użytkownika, aplikacja ładuje się w głównym oknie (w iframe), zaczyna się zliczanie czasu spędzonego w aplikacji, w sidebarze sortowana jest na bieżąco lista Top 5 (uaktualniany jest czas spędzony w danej aplikacji oraz pozycja aplikacji na liście). Po zamknięciu aplikacji czas przestaje być zliczany i pojawia się znowu widok wszystkich dostępnych aplikacji.

## User stories

1. Jako developer chcę mieć gotowy stack front-endowy, aby zacząć pracę nad logiką biznesową.

- a. Struktura katalogów
  - b. Konfiguracja Webpack, ESLint, Babel
  - c. Gotowe CSS modules z sassem
  - d. Komendy do uruchomienia aplikacji w dev modzie oraz zbudowania gotowego bundla na produkcję
2. Jako użytkownik chcę zalogować się do platformy, aby z niej skorzystać.
    - a. Formularz logowania
    - b. Załadowanie widoku dostępnych aplikacji po pomyślnym zalogowaniu
  3. Jako użytkownik chcę widzieć sidebar, abym mógł sprawdzić Top 5 aplikacji.
    - a. Sidebar zawiera nazwę użytkownika, przycisk do wylogowania
    - b. Lista Top 5 aplikacji pobrana za pomocą zapytania HTTP
  4. Jako użytkownik, po zalogowaniu, chcę widzieć listę aplikacji do wyboru.
    - a. Lista aplikacji jako grid z ikonami i krótkim opisem aplikacji
    - b. Uruchomienie gry ładuje iframe, w którym wyświetlona jest inna strona/aplikacja
  5. Jako użytkownik chcę widzieć licznik Top 5 odświeżany na bieżąco.
    - a. Komunikacja po protokole WebSocket

## Testowane biblioteki

- React + Redux
- Vue + Vuex

## Co chcemy sprawdzić w praktyce

- narzędzia (Webpack, Babel, ESLint) i boilerplate'y (czy łatwo zrobić setup i upgrade bibliotek? czy łatwo zbudować wersję produkcyjną?)
- component approach
- CSS modules
- state management (store)
- async zapytania
- WebSockets vs store
- routing
- server-side rendering