# Wyższa Szkoła Przedsiębiorczości i Administracji w LublinieWyższa Szkoła Przedsiębiorczości i Administracji w Lublinie

Kierunek: Informatyka

Specjalność: Projektowanie i eksploatacja systemów informatycznych

Tryb studiów: **niestacjonarny**

Poziom studiów: **I stopnia**

**Aplikacja webowa z wykorzystaniem biblioteki React**

(Tytuł pracy dyplomowej w języku angielskim)

Autor: **Damian Marek Jamróz**Nr albumu: **60192**  
Promotor: **dr inż. Sylwester Korga**

**Oświadczenie kierującego pracą.**

Oświadczam, iż niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

………………………….. …………………………..

(data) (podpis)

**Oświadczenie autora pracy**

Świadomy odpowiedzialności prawnej oświadczam, iż niniejsza praca dyplomowa została napisana przeze mnie samodzielnie nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. Nr 24, poz 83 z poźn. zm.) oraz dóbr osobistych chronionych prawem cywilnym dodatkowo nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej podmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Ponadto oświadczam, iż niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

………………………….. …………………………..

(data) (podpis)

**Spis treści**

1. Wprowadzenie
2. Technologie
3. Specyfikacja aplikacji

# Wstęp

W tym rodziale przedstawiony zostanie wstęp, cel oraz zakres pracy inżynierskiej.

## 1.1 Wstęp

Współczesny świat jest świadkiem rewolucji cyfrowej, która przekształca sposób, w jaki żyjemy, pracujemy i komunikujemy się. Aplikacje webowe, jako kluczowe narzędzia tej transformacji, odgrywają centralną rolę w codziennym życiu milionów ludzi. Nie są one już tylko prostymi stronami internetowymi, ale zaawansowanymi systemami, które umożliwiają realizację skomplikowanych zadań, dostęp do szerokiej gamy usług i nieograniczoną interakcję społeczną. W tym kontekście, rozwój aplikacji webowych nie jest już tylko kwestią technologiczną, ale staje się elementem kultury współczesnej, wpływającym na kształtowanie nowych modeli biznesowych, edukacyjnych i społecznych.

Wybór biblioteki React do stworzenia aplikacji webowej wynika z potrzeby odpowiedzi na te zmieniające się wymagania. React, będąc jednym z najbardziej innowacyjnych i dynamicznie rozwijających się narzędzi, umożliwia tworzenie aplikacji, które są nie tylko funkcjonalne, ale również estetycznie atrakcyjne i łatwe w obsłudze. Jego architektura oparta na komponentach pozwala na modularne budowanie aplikacji, co przekłada się na lepszą organizację kodu, łatwość w utrzymaniu i możliwość ponownego wykorzystania kodu. To sprawia, że React jest idealnym wyborem dla projektów, które wymagają szybkości, efektywności i elastyczności.

W tej pracy inżynierskiej, poprzez pryzmat biblioteki React, zostanie przedstawione, jak nowoczesne technologie webowe mogą być wykorzystane do tworzenia aplikacji, które nie tylko spełniają swoje podstawowe funkcje, ale również wzbogacają doświadczenia użytkowników i otwierają przed nimi nowe możliwości. Zostanie to osiągnięte poprzez dogłębną analizę możliwości, jakie oferuje React, oraz praktyczne zastosowanie tych możliwości w procesie tworzenia aplikacji webowej.

## 1.2 Cel pracy

Głównym celem niniejszej pracy inżynierskiej jest dogłębne zbadanie i zaprezentowanie możliwości, jakie oferuje biblioteka React w kontekście tworzenia nowoczesnych aplikacji webowych. Praca ta ma na celu nie tylko teoretyczne przedstawienie zalet Reacta, ale również praktyczne wykorzystanie tej biblioteki do zaprojektowania i stworzenia aplikacji, która będzie służyć jako narzędzie do monitorowania rynku kryptoaktywów oraz zarządzania własnym portfolio inwestycyjnym.

W ramach pracy zostanie przeprowadzone porównanie Reacta z innymi popularnymi rozwiązaniami dostępnymi w ekosystemie JavaScript, takimi jak Angular i Vue (czy takie coś powinno być czy nie?). Analiza ta pozwoli na wyciągnięcie wniosków dotyczących specyfiki każdego z tych narzędzi oraz ich przydatności w różnych scenariuszach programistycznych. Szczególny nacisk zostanie położony na aspekty takie jak szybkość tworzenia interfejsów użytkownika, reużywalność komponentów, deklaratywne podejście do budowy aplikacji, jednokierunkowy przepływ danych oraz wszechstronność w integracji z innymi technologiami i API.

Założeniem Reacta, które stanowi o jego wyjątkowości, jest umożliwienie szybkiego i efektywnego tworzenia interfejsów użytkownika, które są zarówno skalowalne, jak i łatwe w utrzymaniu. Biblioteka ta, dzięki swojemu deklaratywnemu charakterowi i jednokierunkowemu przepływowi danych, umożliwia tworzenie aplikacji, które są nie tylko wydajne, ale również zapewniają doskonałe doświadczenia dla użytkowników końcowych. W kontekście rosnącego zainteresowania rynkiem kryptowalut, aplikacja webowa stworzona w ramach tej pracy będzie stanowiła praktyczny przykład wykorzystania Reacta do budowy narzędzia, które nie tylko dostarczy aktualnych informacji o cenach kryptoaktywów, ale również umożliwi użytkownikom zarządzanie ich inwestycjami w sposób intuicyjny i dostosowany do ich indywidualnych potrzeb.

Praca ta ma również na celu podkreślenie znaczenia wyboru odpowiednich narzędzi technologicznych w procesie tworzenia aplikacji webowych, które są coraz bardziej złożone i wymagające. Wybór Reacta jako głównego narzędzia do realizacji projektu nie jest przypadkowy, lecz wynika z jego zdolności do radzenia sobie z wyzwaniami współczesnego programowania webowego oraz z jego rosnącej popularności wśród programistów na całym świecie.

## 1.3 Zakres pracy

Realizacja celów określonych w pracy inżynierskiej wymaga przeprowadzenia szeregu skomplikowanych i wieloetapowych działań. Poniżej przedstawiono poszczególne etapy, które zostaną wykonane w celu osiągnięcia założonego celu:

1. **Przegląd i analiza dostępnych technologii**: Dokładne zbadanie i ocena technologii webowych dostępnych na rynku, z naciskiem na te, które są najbardziej relewantne dla tworzenia aplikacji webowych, w tym bibliotek i frameworków JavaScript.
2. **Charakteryzacja biblioteki React**: Szczegółowy opis architektury, filozofii oraz kluczowych cech Reacta, które przyczyniają się do jego popularności i efektywności w tworzeniu interfejsów użytkownika.
3. **Porównanie Reacta z alternatywnymi rozwiązaniami**: Kontrastowe zestawienie Reacta z innymi frameworkami takimi jak Angular i Vue, pod kątem wydajności, elastyczności, łatwości użycia i wsparcia społeczności. (czy to potrzebne?)
4. **Opracowanie wymagań funkcjonalnych i niefunkcjonalnych aplikacji**: Zdefiniowanie kluczowych funkcji, jakie ma spełniać aplikacja, oraz wymagań niefunkcjonalnych, takich jak wydajność, bezpieczeństwo i skalowalność.
5. **Projektowanie architektury bazy danych**: Stworzenie schematu bazy danych, który będzie efektywnie przechowywał dane niezbędne dla działania aplikacji, z uwzględnieniem najlepszych praktyk w zakresie modelowania danych.
6. **Implementacja backendu aplikacji**: Opracowanie serwera aplikacji, w tym API, które umożliwi komunikację między frontendem a bazą danych, oraz systemu logowania i autoryzacji użytkowników.
7. **Budowa modułu usługowego aplikacji**: Realizacja logiki biznesowej aplikacji, w tym algorytmów odpowiedzialnych za przetwarzanie danych i realizację funkcji aplikacji.
8. **Tworzenie interfejsu użytkownika**: Projektowanie i implementacja frontendu aplikacji z wykorzystaniem Reacta, zapewniającego intuicyjną i responsywną obsługę przez użytkowników.
9. **Pisanie testów jednostkowych i integracyjnych**: Opracowanie testów, które zapewnią weryfikację poprawności działania poszczególnych komponentów aplikacji oraz ich współpracy.
10. **Testowanie manualne i automatyczne**: Przeprowadzenie testów manualnych w celu identyfikacji błędów użytkowych oraz testów automatycznych w celu zapewnienia ciągłej jakości kodu.
11. **Wstępne testy użytkowe**: Organizacja sesji testowych z udziałem potencjalnych użytkowników, aby ocenić użyteczność aplikacji i zebrać opinie, które pomogą w dalszym doskonaleniu produktu.
12. **Wdrożenie aplikacji**: Publikacja gotowej aplikacji w środowisku produkcyjnym, co umożliwi jej dostępność dla szerszego grona odbiorców.
13. **Walidacja i optymalizacja aplikacji**: Monitorowanie działania aplikacji po wdrożeniu, analiza feedbacku użytkowników oraz wprowadzanie niezbędnych poprawek i optymalizacji.

# Przegląd wykorzystanych technologii

Rozdział ten zostanie poświęcony przedstawieniu technologii, bibliotek oraz frameworków wykorzystanych do stworzenia aplikacji.

## 2.1 HTML: Szkielet nowoczesnych aplikacji internetowych

HTML, czyli HyperText Markup Language, jest standardowym językiem znaczników używanym do tworzenia struktur stron internetowych i aplikacji webowych. Od momentu jego stworzenia w 1991 roku przez Tima Bernersa-Lee, HTML ewoluował, dostosowując się do rosnących wymagań nowoczesnego internetu.

HTML zaczął jako prosty sposób na formatowanie i prezentowanie treści w sieci WWW (World Wide Web). Z biegiem lat, kolejne wersje HTML wprowadzały nowe znaczniki i atrybuty, które pozwalały na tworzenie coraz bardziej zaawansowanych i interaktywnych stron internetowych. HTML5, najnowsza wersja HTML, zawiera wsparcie dla multimediów, grafiki, animacji i wielu innych funkcji, które wcześniej wymagały dodatkowych wtyczek lub skomplikowanego kodu JavaScript.[1]

Podstawowe elementy HTML:

* Znaczniki: HTML używa znaczników do definiowania różnych części strony, od nagłówków i akapitów po formularze i obrazy.
* Elementy: Każdy znacznik tworzy element, który może zawierać atrybuty określające jego właściwości lub zachowanie.
* Struktura: HTML definiuje strukturę strony, umożliwiając przeglądarkom internetowym interpretowanie i wyświetlanie treści zgodnie z intencją twórcy.

HTML jest nieodłącznym elementem każdej strony internetowej i stanowi podstawę, na której budowane są bardziej złożone technologie, takie jak CSS (Cascading Style Sheets) do stylowania elementów na stronie i JavaScript do dodawania interaktywności. Współpracuje z Reactem (biblioteka JavaScript), który dynamicznie manipuluje DOM (Document Object Model) stworzonym przez HTML, aby tworzyć bogate i interaktywne widoki dla użytkownika.

## 2.2 JavaScript: Język, który zdefiniował nowoczesne strony internetowe

JavaScript, zaprojektowany przez Brendana Eicha i wprowadzony w 1995 roku, szybko stał się nieodłącznym elementem internetu dzięki swojej zdolności do tworzenia interaktywnych stron webowych. Język ten, pierwotnie znany jako Mocha, a następnie LiveScript, ostatecznie przyjął nazwę JavaScript, stając się standardem w przeglądarkach internetowych po formalnym zatwierdzeniu standardu ECMAScript w czerwcu 1997 roku przez Ecma International.

JavaScript, początkowo wprowadzony jako Mocha, przeszedł znaczącą transformację, stając się językiem, który zrewolucjonizował sposób, w jaki interaktywne strony internetowe są budowane. W 1997 roku, po przyjęciu standardu ECMAScript, JavaScript zyskał solidne fundamenty, które umożliwiły jego dalszy rozwój. Kluczowe wersje języka, takie jak ECMAScript 3 z 1999 roku, ugruntowały jego pozycję, a ECMAScript 5 z 2009 roku wprowadziło istotne ulepszenia, w tym metody dla tablic i tryb ścisły. Największym przełomem była jednak wersja ECMAScript 6 (ES6) z 2015 roku, która wprowadziła zaawansowane funkcje, takie jak klasy, moduły, let i const, funkcje strzałkowe, i promisy, co znacznie poszerzyło możliwości programistów.

Jedną z kluczowych cech JavaScript jest jego dynamiczne typowanie. Oznacza to, że typy danych są określane w trakcie wykonania programu, co przynosi zarówno zalety, jak i wyzwania. Z jednej strony, dynamiczne typowanie zapewnia dużą elastyczność w tworzeniu i prototypowaniu kodu, umożliwiając programistom szybkie wprowadzanie zmian i adaptację do nowych wymagań. Z drugiej strony, może to prowadzić do trudniejszego wykrywania błędów, ponieważ problemy z typami danych mogą ujawnić się dopiero podczas wykonania programu, a nie w fazie kompilacji.

W praktyce, dynamiczne typowanie w JavaScript może prowadzić do błędów w czasie wykonania, które są trudne do zidentyfikowania i naprawienia, szczególnie w większych projektach. Brak ścisłego typowania może również wpływać na wydajność, ponieważ silnik JavaScript musi ciągle interpretować typy danych. W odpowiedzi na te wyzwania, społeczność programistyczna opracowała różne narzędzia i praktyki, aby ułatwić zarządzanie typami danych i poprawić jakość kodu. Przykładem takiego narzędzia jest TypeScript.

JavaScript jest nie tylko wszechstronny, ale również unikalny jako jedyny język, który działa natywnie w przeglądarkach internetowych, co czyni go kluczowym narzędziem dla front-end deweloperów. Umożliwia on tworzenie dynamicznych stron internetowych, manipulację DOM, a także obsługę zdarzeń użytkownika. Dzięki Node.js, JavaScript znalazł również zastosowanie po stronie serwera, co pozwala na tworzenie kompleksowych aplikacji webowych w jednym języku. Bogaty ekosystem bibliotek i frameworków, takich jak React, Angular, Vue.js, wspiera deweloperów w tworzeniu złożonych aplikacji webowych, a narzędzia takie jak Webpack i Babel umożliwiają nowoczesne zarządzanie kodem.

JavaScript jest nieodzownym elementem współczesnych technologii webowych. Jego zastosowania obejmują nie tylko tradycyjne strony internetowe, ale także aplikacje jednostronicowe (SPA), aplikacje mobilne (za pomocą React Native), oraz aplikacje desktopowe (przy użyciu Electron). Wraz z rozwojem technologii, JavaScript znajduje również zastosowanie w dziedzinach takich jak Internet Rzeczy (IoT) oraz sztuczna inteligencja (AI), dzięki narzędziom takim jak TensorFlow.js.

JavaScript jest wspierany przez aktywną i zaangażowaną społeczność programistów. Organizowane są liczne konferencje, takie jak JSConf, które promują wymianę wiedzy i rozwój języka. Dzięki ciągłej ewolucji i wsparciu społeczności, JavaScript pozostaje jednym z najważniejszych języków programowania w świecie technologii, z perspektywą dalszego rozwoju i nowych zastosowań.

## 2.3 Typescript: Rozszerzenie JavaScript o Typowanie Statyczne

W miarę rozwoju JavaScriptu i wzrostu skomplikowania aplikacji webowych, społeczność programistów zaczęła poszukiwać sposobów na ulepszenie skalowalności i zarządzania kodem. W odpowiedzi na te potrzeby, Microsoft wprowadził TypeScript w 2012 roku, nadbudowę nad JavaScriptem, która wprowadza opcjonalne statyczne typowanie i nowoczesne funkcje językowe, które są kompilowane do czystego JavaScriptu. Statyczne typowanie pozwala na wychwytywanie błędów związanych z typami danych już na etapie kompilacji, zanim kod zostanie uruchomiony, co przyczynia się do zwiększenia niezawodności aplikacji.

Kluczowe cechy TypeScript:

* Statyczne typowanie: TypeScript umożliwia definiowanie typów dla zmiennych, argumentów funkcji i obiektów, co pomaga w wykrywaniu błędów przed uruchomieniem kodu.
* Interfejsy: Pozwalają na definiowanie kształtu obiektów, co jest szczególnie przydatne w dużych aplikacjach i przy pracy zespołowej.
* Klasy i dziedziczenie: TypeScript wspiera paradygmaty programowania obiektowego, takie jak klasy i dziedziczenie, co ułatwia organizację kodu.
* Dekoratory: Umożliwiają dodawanie metadanych i logiki do klas i ich członków, co może być wykorzystane w zaawansowanych wzorcach projektowych.
* Moduły: TypeScript wspiera moduły ES6, co pozwala na lepszą organizację kodu i jego ponowne wykorzystanie.
* Narzędzia: Zintegrowane narzędzia do refaktoryzacji i autouzupełniania w popularnych edytorach kodu, co znacznie przyspiesza pracę deweloperów.

TypeScript jest szczególnie ceniony w projektach, gdzie wymagana jest wysoka jakość kodu i łatwość utrzymania. Jego zastosowanie znacznie ułatwia współpracę w zespołach, gdzie różni programiści mogą pracować nad tym samym kodem. Dzięki TypeScript, programiści mogą pisać bardziej przewidywalny i łatwiejszy do zarządzania kod, co jest kluczowe w dużych aplikacjach webowych.

## 2.4 React: Biblioteka, która odmieniła tworzenie interfejsów użytkownika

React, stworzony przez zespół Facebooka i po raz pierwszy zaprezentowany w 2013 roku, zrewolucjonizował sposób budowania interaktywnych interfejsów użytkownika. Jako deklaratywna biblioteka JavaScript, React umożliwia tworzenie składników (komponentów) UI, które są zarówno efektywne, jak i elastyczne, co czyni go jednym z najbardziej wpływowych narzędzi w dzisiejszym świecie front-endu.

React został stworzony, aby sprostać wyzwaniom związanym z budowaniem dużych aplikacji z dynamicznymi interfejsami użytkownika. W odpowiedzi na ograniczenia istniejących rozwiązań, inżynierowie Facebooka opracowali bibliotekę, która wprowadzała innowacyjne podejście do renderowania komponentów. Z czasem React zyskał na popularności, stając się fundamentem dla wielu aplikacji internetowych, w tym głównych produktów Facebooka, takich jak Instagram i WhatsApp.

Kluczowe cechy Reacta:

* Komponentowy model budowy aplikacji: React zachęca do tworzenia reużywalnych komponentów, które zarządzają własnym stanem, co przekłada się na modularność i łatwość w utrzymaniu kodu.
* JSX - JavaScript Syntax Extension: React wprowadził JSX, rozszerzenie składni JavaScript, które pozwala na mieszanie HTML (HyperText Markup Language) z JavaScriptem, co ułatwia tworzenie i zarządzanie interfejsem użytkownika.
* Wirtualny DOM (Document Object Model): React wykorzystuje wirtualny DOM, co pozwala na optymalizację aktualizacji interfejsu użytkownika, minimalizując interakcje z kosztownym w utrzymaniu prawdziwym DOM.
* Jednokierunkowy przepływ danych: Architektura Reacta opiera się na jednokierunkowym przepływie danych, co ułatwia zarządzanie stanem aplikacji i przewidywanie jej zachowania.

Zaawansowane możliwości Reacta:

* Hooki: Wprowadzone w wersji 16.8, hooki pozwalają na wykorzystanie stanu i innych funkcji Reacta w komponentach funkcyjnych, co otworzyło nowe możliwości w projektowaniu aplikacji.
* Context API: Umożliwia przekazywanie danych przez drzewo komponentów bez konieczności przekazywania propsów na każdym poziomie, co znacznie upraszcza zarządzanie stanem w dużych aplikacjach.

## 2.5 Redux: Architektura zarządzania stanem dla aplikacji JavaScript

(tu pytanie, używam w projekcie Reduxa natomiast korzystam z Redux Toolkit czyli takiej wersji która jest całkowicie oparta na Reduxie ale pozwala zmniejszyć ilość kodu, czy powinienem o tym wspominać czy sama główna libka wystarczy?)

Redux to otwarta biblioteka JavaScript, która służy do centralnego zarządzania stanem aplikacji. Jest najczęściej używana z bibliotekami takimi jak React lub Angular do budowania interfejsów użytkownika. Podobnie do architektury Flux stworzonej przez Facebooka, Redux został zaprojektowany przez Dana Abramova i Andrew Clarka.

Redux powstał jako odpowiedź na potrzebę lepszego zarządzania stanem w dużych aplikacjach JavaScript, gdzie przepływ danych może być skomplikowany i trudny do śledzenia. Inspiracją dla jego twórców była chęć stworzenia bardziej przewidywalnego sposobu zarządzania stanem, który ułatwiłby debugowanie i testowanie aplikacji.

Kluczowe koncepcje Reduxa:

* Jedno źródło prawdy: Redux utrzymuje cały stan aplikacji w jednym obiekcie drzewa, co ułatwia śledzenie zmian i zarządzanie stanem.
* Stan jest tylko do odczytu: Jedynym sposobem na zmianę stanu w Reduxie jest wysłanie akcji, czyli obiektu opisującego, co się stało.
* Czyste funkcje reduktorów: Reduktory to czyste funkcje, które określają, jak stan aplikacji zmienia się w odpowiedzi na akcje.

W Reduxie dane przepływają w jednym kierunku:

1. Akcje są wysyłane.
2. Reduktory obliczają nowy stan.
3. Zmiany stanu są odzwierciedlane w interfejsie użytkownika.

Zalety stosowania Reduxa

* Przewidywalność: Dzięki ścisłym zasadom Redux sprawia, że stan aplikacji jest bardziej przewidywalny i łatwiejszy do zarządzania.
* Łatwość debugowania: Redux DevTools i inne narzędzia ułatwiają śledzenie, cofanie i ponowne odtwarzanie akcji, co znacznie ułatwia debugowanie.
* Elastyczność: Redux może być używany z dowolną biblioteką lub frameworkiem JavaScript, choć jest najczęściej stosowany z Reactem.

Redux jest szczególnie przydatny w złożonych aplikacjach, gdzie wiele komponentów musi mieć dostęp do wspólnego stanu. Umożliwia on łatwe przekazywanie danych między komponentami i zapewnia spójność stanu w całej aplikacji. Jego podejście do zarządzania stanem stało się standardem w rozwoju aplikacji React i jest szeroko stosowane.

## 2.6 Vite: Nowoczesne narzędzie do budowy aplikacji webowych

Vite jest nowoczesnym narzędziem do budowy aplikacji webowych, które zyskało uznanie dzięki swojej prędkości i wydajności. Zaprojektowany z myślą o zapewnieniu szybszego i bardziej efektywnego doświadczenia deweloperskiego dla nowoczesnych projektów webowych, Vite składa się z dwóch głównych części: serwera deweloperskiego oraz polecenia budowania.

Vite został stworzony, aby sprostać potrzebie szybszego czasu uruchamiania i gorącej wymiany modułów (Hot Module Replacement - HMR) w porównaniu do istniejących narzędzi takich jak Webpack. Jego celem jest uproszczenie procesu konfiguracji i zwiększenie prędkości rozwoju, szczególnie w dużych aplikacjach z wieloma zależnościami.

Kluczowe cechy Vite:

* Szybkość i wydajność: Dzięki wykorzystaniu natywnych modułów ES w przeglądarkach, Vite oferuje błyskawiczne uruchamianie projektów i aktualizacje HMR.
* Łatwość konfiguracji: Vite jest dostarczany z sensownymi ustawieniami domyślnymi, co pozwala na szybkie rozpoczęcie pracy bez konieczności skomplikowanej konfiguracji.
* Wsparcie dla różnych frameworków: Vite może być używany z popularnymi frameworkami takimi jak Vue, React, a także z czystym JavaScriptem, oferując szablony i wsparcie dla TypeScript, JSX i Sass.

Przepływ pracy z Vite:

1. Serwer deweloperski: Vite zapewnia serwer deweloperski z bogatymi funkcjami, takimi jak szybki HMR, który znacznie przyspiesza proces tworzenia i testowania interfejsów użytkownika.
2. Budowanie dla produkcji: Vite wykorzystuje Rollup do optymalizacji zasobów statycznych, zapewniając wydajne budowanie gotowe do wdrożenia w środowisku produkcyjnym.

Zalety stosowania Vite:

* Szybkie odświeżanie modułów: Vite zapewnia konsekwentnie szybkie aktualizacje HMR, niezależnie od rozmiaru aplikacji.
* Nowoczesne środowisko deweloperskie: Vite pozwala pominąć krok tworzenia pakietów podczas dewelopmentu, serwując przeglądarce natywne moduły ES.
* Optymalizacje wydajności: Domyślne konfiguracje Vite zapewniają wiele optymalizacji wydajnościowych w gotowych do produkcji pakietach.

Vite jest częścią rosnącego ekosystemu narzędzi front-endowych, które ułatwiają tworzenie nowoczesnych aplikacji webowych. Jego podejście do budowania projektów jest dobrze przyjęte przez społeczność deweloperów i ma duży potencjał do dalszego rozwoju i adaptacji do zmieniających się potrzeb branży webowej.

Vite, dzięki swojej prostocie i wydajności, stał się popularnym wyborem wśród deweloperów aplikacji webowych, oferując szybkie i elastyczne środowisko pracy, które jest łatwe w konfiguracji i wspiera najnowsze standardy webowe.

Vite, axios, chart.js, styled-components, mui, slick-carousel, react query, redux toolkit,

# Literatura

[1] HTML: HyperText Markup Language.

https://developer.mozilla.org/en-US/docs/Web/HTML. Dostęp data.

[2] JavaScript.

https://developer.mozilla.org/en-US/docs/Web/JavaScript. Dostęp data.

[3] TypeScript for JavaScript Programmers.

https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html. Dostęp data.

[4] React Reference Overview.

<https://react.dev/reference/react>. Dostęp data.

[5] HTML: HyperText Markup Language.

https://developer.mozilla.org/en-US/docs/Web/HTML. Dostęp data.

[6] HTML: HyperText Markup Language.

https://developer.mozilla.org/en-US/docs/Web/HTML. Dostęp data.

[7] HTML: HyperText Markup Language.

https://developer.mozilla.org/en-US/docs/Web/HTML. Dostęp data.

[8] HTML: HyperText Markup Language.

https://developer.mozilla.org/en-US/docs/Web/HTML. Dostęp data.