

QUIZ

MARCEL

ŻAK

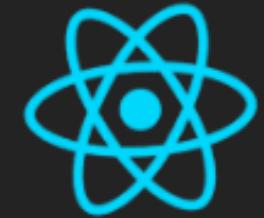
QUIZ W REACT

KROK 1 TWORZENIE

```
$ npm create vite@latest quiz -- --template react  
cd quiz  
npm install
```

KROK 2 START SERWERA

```
npm run dev
```

[HOME](#)[ABOUT](#)[MORE](#)

Vite + React

count is 0

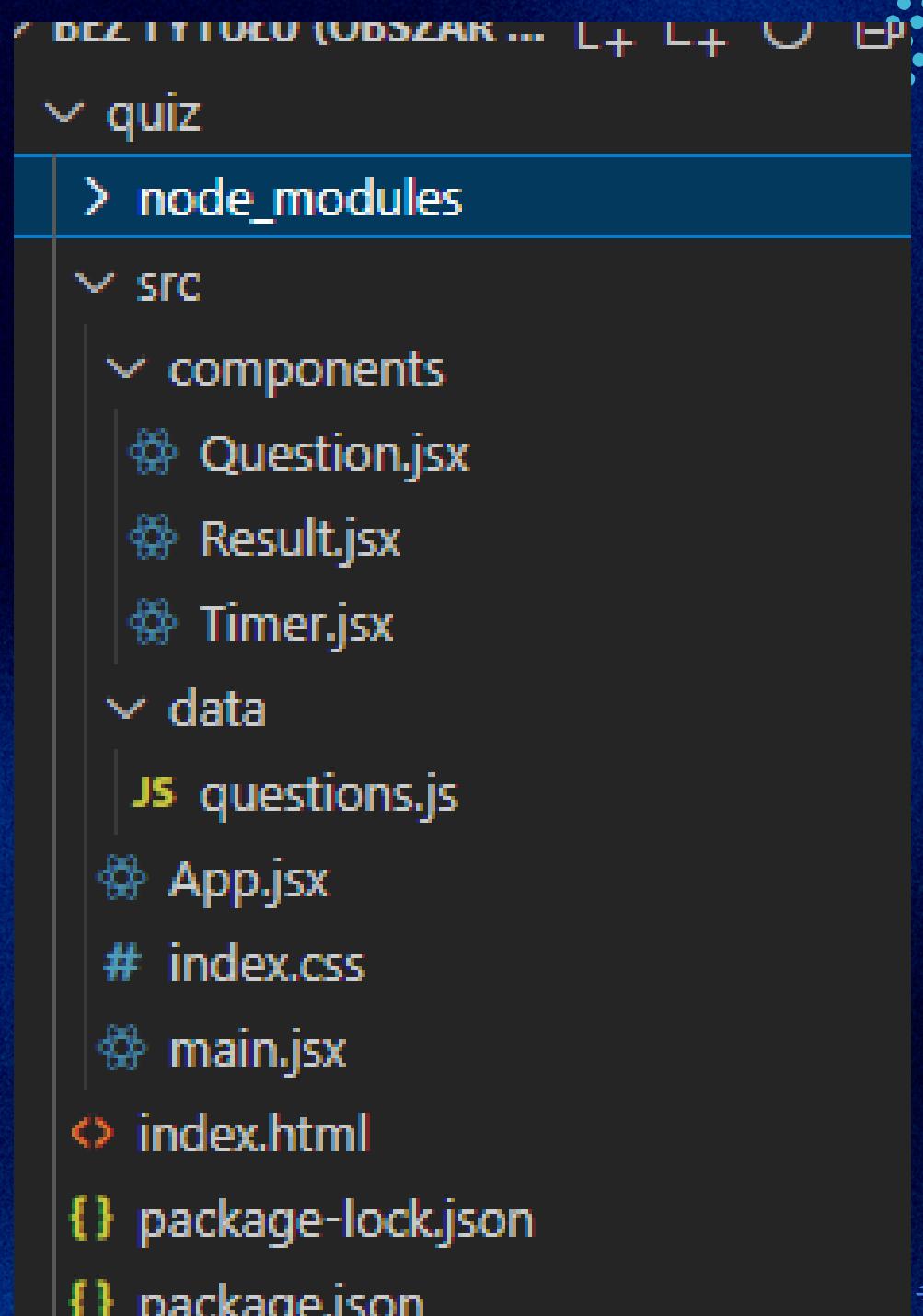
Edit src/App.tsx and save to test HMR

Click on the Vite and React logos to learn more

KROK 3

STRUKTURA

PLIKÓW



```
DESKTOP (ODSŁĄK) L+ L+ ⌂
└── quiz
    ├── node_modules
    └── src
        ├── components
        │   ├── Question.jsx
        │   ├── Result.jsx
        │   └── Timer.jsx
        ├── data
        │   ├── questions.js
        │   ├── App.jsx
        │   ├── index.css
        │   ├── main.jsx
        │   └── index.html
        └── package-lock.json
            └── package.json
```

PYTANIA

Stworzenie pytań

Aby nasza aplikacja mogła działać, potrzebujemy zestawu pytań i odpowiedzi. W tym kroku tworzymy plik questions.js w folderze data, który zawiera tablicę obiektów. Każdy obiekt reprezentuje jedno pytanie z czterema możliwymi odpowiedziami oraz indeksem poprawnej odpowiedzi. Ten plik będzie później importowany w głównym komponencie, dzięki czemu aplikacja będzie mogła dynamicznie przechodzić przez kolejne pytania.

Kod

```
export const questionsData = [
  {
    question: "Stolica Polski to:",
    answers: ["Kraków", "Warszawa", "Gdańsk", "Wrocław"],
    correct: 1,
  },
  {
    question: "Ile to  $5 + 3$ ?",
    answers: ["6", "7", "8", "9"],
    correct: 2,
  },
  {
    question: "Który pierwiastek chemiczny ma symbol O?",
    answers: ["Ołów", "Złoto", "Tlen", "Tytan"],
    correct: 2,
  },
  {
    question: "Które morze leży na północy Polski?",
    answers: ["Bałtyckie", "Czarne", "Śródziemne", "Północne"],
    correct: 0,
  },
  {
    question: "W którym roku rozpoczęła się II wojna światowa?",
    answers: ["1918", "1939", "1945", "1920"],
    correct: 1,
  },
  {
    question: "Jak nazywa się proces wytwarzania energii w Słońcu?",
    answers: ["Fotosynteza", "Rozszczepienie", "Synteza jądrowa", "Kondensacja"],
    correct: 2,
  },
  {
    question: "Kto napisał 'Pan Tadeusz'?",
    answers: ["Juliusz Słowacki", "Adam Mickiewicz", "Henryk Sienkiewicz", "Bolesław Prus"],
    correct: 1,
  },
  {
    question: "Który język programowania używany jest w React?",
    answers: ["Java", "Python", "JavaScript", "C++"],
    correct: 2,
  },
  {
    question: "Co to jest CPU w komputerze?",
    answers: ["Monitor", "Procesor", "Pamięć", "Dysk"],
    correct: 1,
  },
  {
    question: "Która planeta jest najbliższej Słońca?",
    answers: ["Wenus", "Ziemia", "Mars", "Mercury"],
    correct: 3,
  },
];
```

LOGIKA PYTAŃ

Question.jsx to komponent odpowiedzialny za wyświetlanie konkretnego pytania oraz możliwych odpowiedzi. Użytkownik może kliknąć jedną z opcji, a komponent zaznaczy ją jako wybraną i na chwilę pokaże, czy odpowiedź była poprawna (zielona) czy błędna (czerwona). Następnie po dwóch sekundach informuje główny komponent (App) o wyniku tej odpowiedzi. Dodatkowo, jeśli użytkownik nie udzieli odpowiedzi w ciągu pięciu sekund, automatycznie uruchamia się funkcja, która traktuje pytanie jako błędnie udzielone i pokazuje poprawną odpowiedź. Dzięki temu komponent potrafi działać zarówno interaktywnie (kliknięcie), jak i reagować na upływ czasu.

```
import React, { useState, useEffect } from 'react';
import Timer from './Timer';

const Question = ({ question, onAnswer }) => {
  // Jeśli pytanie jeszcze się nie załadowało, pokaż komunikat
  if (!question) return <div>Ładowanie pytania...</div>;

  // Stan przechowujący indeks wybranej odpowiedzi (lub null, jeśli brak)
  const [selected, setSelected] = useState(null);
  // Stan informujący, czy pytanie zostało już odpowiedziane
  const [answered, setAnswered] = useState(false);

  // Po zmianie pytania resetujemy stany wyboru i odpowiedzi
  useEffect(() => {
    setSelected(null);
    setAnswered(false);
  }, [question]);

  // Obsługa kliknięcia odpowiedzi
  const handleAnswer = (index) => {
    if (answered) return; // jeśli już odpowiedziano, ignoruj kliknięcia
    setSelected(index); // ustaw wybraną odpowiedź
    setAnswered(true); // oznacz pytanie jako odpowiedzione

    // Po 2 sekundach wywołaj callback informujący o wyniku
    setTimeout(() => {
      onAnswer(index === question.correct);
    }, 2000);
  };

  // Funkcja wywoływana, gdy czas na odpowiedź minął
  const handleTimeUp = () => {
    if (!answered) {
      setAnswered(true); // oznacz pytanie jako odpowiedzione
      setSelected(null); // brak wybranej odpowiedzi

      // Po 2 sekundach wywołaj callback z wynikiem false (niepoprawnie, bo brak odpowiedzi)
      setTimeout(() => {
        onAnswer(false);
      }, 2000);
    }
  };
}
```

```
return (
  <>
  {/* Timer wyświetlany tylko, gdy pytanie nie zostało jeszcze odpowiedzione */}
  {!answered && <Timer duration={5} onTimeUp={handleTimeUp} />}

  {/* Wyświetlenie pytania */}
  <div className="question">{question.question}</div>

  {/* Lista odpowiedzi */}
  <div className="answers">
    {question.answers.map((ans, idx) => {
      const isCorrect = idx === question.correct; // czy ta odpowiedź jest poprawna
      const isSelected = idx === selected; // czy jest wybrana przez użytkownika

      // Klasa CSS zależna od stanu - pokazuje kolor poprawnej i niepoprawnej odpowiedzi po udzieleniu odpowiedzi
      let className = '';
      if (answered) {
        if (isCorrect) className = 'correct'; // zielony dla poprawnej odpowiedzi
        else if (isSelected && !isCorrect) className = 'incorrect'; // czerwony dla wybranej złej odpowiedzi
      }

      return (
        <button
          key={idx}
          onClick={() => handleAnswer(idx)}
          className={className}
          disabled={answered} // po odpowiedzi blokujemy dalsze klikanie
          type="button"
        >
          {ans}
        </button>
      );
    })}
  </div>
);

export default Question;
```

ODLICZANIE

Timer.jsx odpowiada za odliczanie czasu dla każdego pytania. Wizualnie przedstawia to jako pasek postępu nad pytaniem, który stopniowo się skraca. Komponent ten jest bardzo dynamiczny – działa przez pięć sekund i wywołuje określoną funkcję w Question, jeśli czas się skończy. Działa bardzo płynnie, aktualizując pasek co 50 milisekund. Gdy użytkownik udzieli odpowiedzi, timer zniknie lub przestaje działać, co zapobiega dalszemu odliczaniu.

[HOME](#)[ABOUT](#)[MORE](#)

```
import React, { useEffect, useState } from "react";

const Timer = ({ duration, onTimeUp }) => {
  // Stan przechowujący procent szerokości paska (100% = pełny pasek, 0% = pasek pusty)
  const [widthPercent, setWidthPercent] = useState(100);

  useEffect(() => {
    // Resetujemy szerokość paska do 100% przy każdym rozpoczęciu nowego odliczania
    setWidthPercent(100);

    // Zapisujemy czas rozpoczęcia odliczania (w ms)
    const start = Date.now();

    // Ustawiamy interwał, który co 50 ms aktualizuje pasek postępu
    const interval = setInterval(() => {
      // Obliczamy ile czasu upłynęło od startu (w ms)
      const elapsed = Date.now() - start;

      // Obliczamy procent pozostałego czasu, skalując do wartości 0-100%
      // (elapsed / (duration * 1000)) to ułamek czasu, więc odejmujemy go od 100%
      const percent = Math.max(100 - (elapsed / (duration * 1000)) * 100, 0);

      // Aktualizujemy stan szerokości paska procentowego
      setWidthPercent(percent);

      // Jeśli pasek osiągnie 0% (czas się skończył)
      if (percent <= 0) {
        // Zatrzymujemy interwał
        clearInterval(interval);
        // Wywołujemy callback informujący o końcu czasu
        onTimeUp();
      }
    }, 50); // odświeżamy pasek co 50 ms

    // Cleanup: przy odmontowaniu komponentu lub zmianie duration/onTimeUp
    // usuwamy interwał, żeby nie przeciągał pamięciowo
    return () => clearInterval(interval);
  }, [duration, onTimeUp]);

  // Renderujemy pasek czasu jako div z szerokością ustawioną na aktualny procent
  return (
    <div className="timer">
      <div className="timer-progress" style={{ width: `${widthPercent}%` }} />
    </div>
  );
}

export default Timer;
```



Result.jsx to bardzo prosty komponent, którego zadaniem jest wyświetlenie informacji o zakończeniu quizu. Pokazuje użytkownikowi komunikat, że to już koniec, oraz jego końcowy wynik, czyli ile pytań zostało udzielonych poprawnie spośród wszystkich.

```
import React from "react";

// Komponent wyświetlający wynik końcowy quizu
// Przyjmuje dwie propsy:
// - score: ilość poprawnych odpowiedzi użytkownika
// - total: łączna liczba pytań w quizie
const Result = ({ score, total }) => (
  <div
    style={{
      textAlign: "center", // wyśrodkowanie tekstu
      marginTop: 40           // odstęp od góry (piksele)
    }}
  >
    <h2>Koniec quizu!</h2> /* Nagłówek informujący o zakończeniu */
    <p>Twój wynik: {score} / {total}</p> /* Wyświetlenie wyniku */
  </div>
);

export default Result;
```

Stolica Polski to:

Kraków

Warszawa

Gdańsk

Wrocław

Stolica Polski to:

Kraków

Warszawa

Gdańsk

Wrocław

Stolica Polski to:

Kraków

Warszawa

Gdańsk

Wrocław

Koniec quizu!

Twój wynik 0 / 10

HOME

ABOUT

MORE

KONIEC

Marcel Źak 4p