**PIERWSZA CZĘŚĆ ZADANIA - TEORIA:**

**1. Dlaczego HTML i CSS nie mogą być uważane za pełne**

**języki programowania?**

HTML (HyperText Markup Language) i CSS (Cascading Style Sheets) są językami używanymi do tworzenia i stylizacji stron internetowych. Chociaż HTML i CSS są ważnymi językami używanymi w tworzeniu stron internetowych, nie posiadają one wszystkich cech i funkcji, które definiują tradycyjne języki programowania. Dopiero w połączeniu z językiem JavaScript (który jest pełnoprawnym językiem oprogramowania - po stronie frontendu czy backendu, w zależności od kontekstu), można tworzyć bardziej interaktywne i dynamiczne aplikacje internetowe.

Uzasadnienie / powody:

1. Brak pełnej logiki programowania: HTML służy głównie do tworzenia struktury treści na stronie internetowej (szkielet strony), definiowania nagłówków, paragrafów, list, obrazków itp. CSS natomiast odpowiada za stylizację i układ elementów. Obie te technologie nie posiadają rozbudowanych mechanizmów programowania.
2. HTML i CSS są głównie odpowiedzialne za prezentację treści na stronie internetowej.Nie mają wbudowanych mechanizmów manipulacji danymi, takich jak zmienne, struktury danych czy operatory. Bez tych elementów trudno jest tworzyć interaktywne aplikacje, które wymagają przetwarzania i manipulacji danych.
3. Zależność od innych języków programowania (jak powyżej).

**2. Jaki format kolumny polecacie na bloga?**

Jako, że obecnie strony wyświetlamy na różnych urządzeniach / ekranach, wybór formatu powinien zapewnić optymalne wyświetlanie treści w każdej sytuacji,a to zapewnia Responsywny Układ (RWD), który umożliwia dostosowanie kolumn do danego urządzenia (elastyczny układ, skalowalne obrazy, czytelność tekstu, responsywna nawigacja).

**3. Jak zmniejszyć obciążenie back-end za pomocą JavaScript?**

Zmniejszenie obciążenia back-end za pomocą JavaScript jest możliwe poprzez przeanalizowanie i zaprojektowanie odpowiednio architektury front-endu. Optymalizacja obciążenia back-endu za pomocą rozwiązań front-endowych zależy od konkretnego kontekstu aplikacji i wymagań, jednak są techniki, które przy projektowaniu / wywiadzie można uwzględnić:

1. Czy musimy pobierać większy zakres danych? Określenie zakresu potrzebnych / niezbędnych danych do pobierania. Możliwość użycia technologii AJAX, która umożliwia pobieranie danych asynchronicznie, bez konieczności przeładowywania całej strony.
2. Przy danych, które są rzadko zmieniane lub niezmienne można zastosować rozwiązanie przechowywania lokalnie danych po stronie klienta (mechanizmy pamięci podręcznej tj.: Local Storage / Session Storage).
3. Można zastosować pamięć podręczną przeglądarki do przechowywania statycznych zasobów, takich jak pliki JavaScript, arkusze stylów CSS i obrazy. Dzięki temu przeglądarka będzie pobierać te zasoby tylko raz, co przyspieszy ładowanie strony i zmniejszy obciążenie back-endu (mechanizmy buforowania przeglądarki).
4. Lazy loading polega na ładowaniu zasobów (np. obrazów, skryptów) dopiero w momencie, gdy są one potrzebne na stronie. Można zastosować tę technikę do odroczenia ładowania niektórych treści, takich jak obrazy znajdujące się poza widokiem użytkownika. To pozwoli zmniejszyć czas ładowania strony i obciążenie back-endu.

**DRUGA CZĘŚĆ ZADANIA - CASE:**

**1 - CRM:** „Muszę stworzyć system CRM dla programu partnerskiego. Mamy wielu użytkowników i chcę, aby system CRM działał szybko. Nasi pracownicy pracują z różnych urządzeń (laptopów, tabletów, telefonów). CRM musi rozwiązywać wszystkie problemy, z jakimi borykają się menedżerowie podczas pracy z partnerami: dodawanie nowych, przeglądanie istniejącej bazy danych, szybkie wyszukiwanie itp. Ponadto system CRM powinien dostarczać informacji statystycznych o partnerach w postaci ładnych i elastycznie dostosowywanych wykresów. Ponieważ CRM będzie używany tylko przez pracowników naszej firmy, nie mam specjalnych wymagań dotyczących wyglądu”.

Słowa klucze:

1. Rozmiar projektu - system CRM / wielu użytkowników (duży projekt).
2. Wydajność systemu / skalowalność (CRM ma działać szybko).
3. Wieloplatformowość (użytkownicy działają z wielu urządzeń).
4. Zmienność potrzeb managerskich (elastyczność) skorelowane ze zmianami w obrębie biznesu / problemami.
5. Wizualizacja elastycznie dostosowanych wykresów.

Proponowana technologia / uzasadnienie / powiązanie ze słowami kluczowymi (numeryczne):

Angular / JavaScript - przeznaczony do większych projektów **(1)** , a w tym do systemów CRM **(1)**. Angular jest bardziej gotowy (w porównaniu do React) do tworzenia aplikacji wieloplatformowych, ponieważ oferuje narzędzia i struktury, które ułatwiają pisanie i utrzymanie jednego kodu dla różnych platform, m.in.: "platform-agnostic design",zintegrowany framework Ionic **(3)**. Posiada własny modułowy system, wbudowane narzędzia do zarządzania stanem aplikacji, obsługę routingu, testowania jednostkowego i jest zoptymalizowane pod kątem osiągnięcia wysokiej wydajności **(2)** oraz aktualizacji kodu **(4)**. Posiada wsparcie i dużą społeczność użytkowników, co może być pomocne przy wprowadzaniu wszelkich zmian / utrzymywaniu aplikacji **(2, 4)**. Pomimo tego, że wymaga więcej czasu na uruchomienie aplikacji (niż np. React), Angular wykorzystuje mechanizm lazy-loading, który umożliwia dynamiczne ładowanie modułów tylko wtedy, gdy są potrzebne **(2)**.

Proponowany zespół / uzasadnienie powiązanie ze słowami kluczowymi (numeryczne):

Tworzony system jest skomplikowany, a próg wejścia przy w/w technologii dość wysoki, dlatego w zespole powinni znajdować się developerzy (Senior / Mid). Koszt wdrożenia systemu jest stosunkowo wysoki, dlatego dla zachowania jakości vs cena produktu rekomendowałabym zespół z większym doświadczeniem.

* Senior Front-end developer - Javascript, Typescript, Angular, Webpack (np. zastosowanie Code Splitting, **2**), D3.js **(5)**, Rest API, TrackJS **(2)**, HTML, CSS, Bootstrap **(2)**,
* Middle Front-end developer - Javascript, Typescript, Angular, Webpack, Bootstrap,
* wsparcie realizacji (w razie zapotrzebowania) - min. Middle Front-end developer.

**2 - E-commerce:** „Mamy duży sklep internetowy. Został on zaprojektowany 7 lat temu i od tego czasu nic z nim nie robiono. Musimy dodać nową funkcjonalność - osobiste konto kupującego, na którym będzie on mógł przeglądać informacje o swoich zamówieniach. Czas i budżet są ograniczone, więc nie możemy sobie pozwolić na stworzenie całego sklepu od nowa”.

Słowa klucze:

1. Czas i budżet są ograniczone.
2. Nowa funkcjonalność vs integracja z innymi rozwiązaniami aplikacji.
3. Sklep internetowy vs wydajność aplikacji (duży sklep internetowy).

Proponowana technologia / uzasadnienie / powiązanie ze słowami kluczowymi (numeryczne):

Vue.js - przeznaczony zarówno do małych, jak i większych projektów. Jest przystępnym, wydajnym **(3)** oraz wszechstronnym frameworkiem do tworzenia interfejsów użytkownika. Nie planujemy zmieniać całej aplikacji, a jedynie dodać gotowy komponent, a rekomendowany framework można łatwo zintegrować z istniejącymi rozwiązaniami / komponentami, co jest przydatne przy rozbudowie już istniejących sklepów internetowych **(2)**.Posiadamy w zespole jednego senior developera, który pracuje w tym frameworku, co przy wymogu nadrzędnym, tj. czas i budżet mogłoby stanowić zagrożenie w sytuacji konieczności rozszerzenia zasobów, jednak technologia ma niski prób wejścia, prostą składnię i jasną dokumentację, a to oznacza, że w razie potrzeby do zespołu można włączyć innego developera, dla którego przejście na Vue.js nie powinno stanowić problemu **(1)**.

Proponowany zespół:

* Senior Front-end developer - Javascript, Vue.js, ES6, Rest, API, HTML, CSS,
* W razie zapotrzebowania (casus czas) - Middle Front-end developer.

**3 - Photo gallery:** „Chcemy stworzyć galerię zdjęć, w której każdy użytkownik będzie mógł przechowywać zdjęcia posortowane w albumy. Albumy mają mieć swoje własne ustawienia prywatności. Pod każdym zdjęciem można zostawiać komentarze i komunikować się z innymi użytkownikami. Galeria umożliwia subskrybowanie aktualizacji albumu i otrzymywanie powiadomień o dodaniu nowych zdjęć za pośrednictwem wyskakujących w czasie rzeczywistym powiadomień na stronie”.

Słowa klucze:

1. Możliwość komentowania i komunikowania z innymi użytkownikami.
2. Możliwość subskrypcji albumu i otrzymywanie powiadomień o dodaniu nowych zdjęć.
3. Domyślnie: wieloplatformowość (standard w obecnych czasach - użytkownicy działają z wielu urządzeń).
4. Domyślnie: wydajność aplikacji.

Proponowana technologia / uzasadnienie / powiązanie ze słowami kluczowymi (numeryczne):

Biorąc pod uwagę RFP w zestawieniu z dostępnymi kompetencjami / technologiami w zespole, proponuję zastosowanie frameworku React, który oferuje pełne możliwości przeglądarek i urządzeń mobilnych, w tym również nadaje się do aplikacji SPA **(3)**. Jest to elastyczny framework, w którym ważny jest element bezpieczeństwa, umacniany przez dużą i aktywną społeczność. Ponadto, posiada szerokie narzędzia developerskie, pozwalające na dokładne zrozumienie hierarchii komponentów, śledzenie stanu i wydajności aplikacji **(4)**. React sam w sobie jest biblioteką do tworzenia interfejsu użytkownika, więc nie zapewnia wbudowanych funkcji takich jak dodawanie komentarzy czy subskrybowanie zmian (jak np. WebSockets), ale z powodzeniem można łączyć go z innymi bibliotekami, narzędziami i frameworkami, które umożliwiają implementację tych funkcjonalności **(1, 2)**.

Proponowany zespół:

* Senior Front-end developer - Javascript, React.js, Redux, ES6,Webpack, Node.js, Canvas, WebSockets **(1,2)**, Rest API, TrackJS, HTML, CSS,
* Middle Front-end developer - Javascript, React.js, Redux,ES6, Bootstrap