Czy pamiętasz jeszcze czasy, gdy strony internetowe budowane były w jednym edytorze tekstu? HTML i CSS były wszystkim, co trzeba było opanować, aby je stworzyć. Na myśl o tych czasach na mojej twarzy pojawia się uśmiech. Dlaczego? Po pierwsze, wyobrażam sobie, jak łatwe było kiedyś tworzenie stron internetowych (a może nie było, skoro mamy teraz edytory UI?). Po drugie, pamiętam, jak śmiesznie wyglądały strony zbudowane w ten sposób, zwłaszcza w porównaniu z dzisiejszymi. W dzisiejszych czasach każde oprogramowanie, niezależnie od tego, czy jest to strona internetowa, aplikacja czy jakikolwiek inny projekt, wymaga kilku elementów technologii, aby zapewnić odpowiednią funkcjonalność i wygląd. Procesy tworzenia stron internetowych ewoluowały, rozwój aplikacji również. Aby lepiej je zrozumieć, chciałbym poświęcić kilka słów znaczeniu stosów w aspekcie tworzenia aplikacji internetowych

Co to jest Tech Stack i jaka jest jego historia?

Stos technologiczny to połączenie technologii, języków programowania, narzędzi i frameworków używanych do tworzenia i uruchamiania aplikacji. Aplikacje mogą mieć różną liczbę elementów w swoim stosie technologicznym, a ogólna złożoność wzrasta wraz z każdym nowym dodanym stosem. Istnieje od 3 do 5 głównych grup typów stosów technologicznych, w zależności od tego, jak próbujesz je pogrupować. Są to:

- **Stos frontend** – część aplikacji po stronie klienta. Na przykład HTML i JS, które są uruchamiane w przeglądarce.
- **Backend stack** – część aplikacji po stronie serwera. Często oprócz logiki znajdują się tutaj połączenia API
- **Stos bazy danych** – czasami uważany za część stosu backendu

  .

- **Stos DevOps** – konfiguracja serwera, konteneryzacja, technologia wirtualizacji, zautomatyzowane testowanie i wdrażanie
- **Web3 stack** – zdecentralizowana/inteligentna warstwa kontraktowa. Czasami uważana za część stosu Frontend lub część stosu Backend

  .

Wydaje się więc, że czasami nie jest łatwo przypisać określoną część warstwy do

konkretnego stosu technologicznego. Może to być na przykład kwestia sporna – czy Redis jest częścią DevOps czy Backend? Ale tak naprawdę ta klasyfikacja nie jest tak ważna, jak faktyczne relacje między różnymi stosami – ale wrócę do tego później. Na razie można zauważyć inną obserwację, że stos technologiczny aplikacji internetowych rozrósł się z czasem:

1. Tylko frontend – były to najprostsze strony internetowe napisane w HTML. Można argumentować, że potrzebny był serwer (taki jak Apache), aby zapewnić nawet prostą stronę HTML, ale z drugiej strony można było uruchomić ją lokalnie bez serwera, po prostu za pomocą przeglądarki. Tak więc najczęściej obejmowało to HTML i JS

   .

2. Frontend + backend + serwer – teraz następuje przeskok z jednego do trzech stosów. Kiedy pojawiła się potrzeba posiadania dynamicznych stron internetowych, wprowadzono PHP. Tak więc teraz strony HTML, które mają być dynamiczne, mogą korzystać z backendu PHP. Ale backend PHP wymaga serwera do działania. Tak więc do uruchomienia takiej aplikacji potrzebne były 3 stosy (rozważam tutaj „serwer” jako odpowiednik Apache).
3. Frontend + backend + serwer + baza danych – tu zaczyna się zabawa, bo gdy pozwala się na przechowywanie i ponowne wykorzystanie danych, można wykonać bardzo wiele zadań – od prostej „księgi gości” czy zarządzania użytkownikami, po bardzo złożone aplikacje. I tutaj warto zauważyć, że istnieje kilka najpopularniejszych konfiguracji dla takiego podejścia, takich jak:
   1. **LAMP** – czyli Linux + Apache + MySQL + PHP
   2. **WAMP** – czyli Windows + Apache + MySQL + PHP
   3. **MEAN** – czyli MongoDB, Express.js, AngularJS i Node.js
   4. I lista jest długa, w dowolnej kombinacji

      .

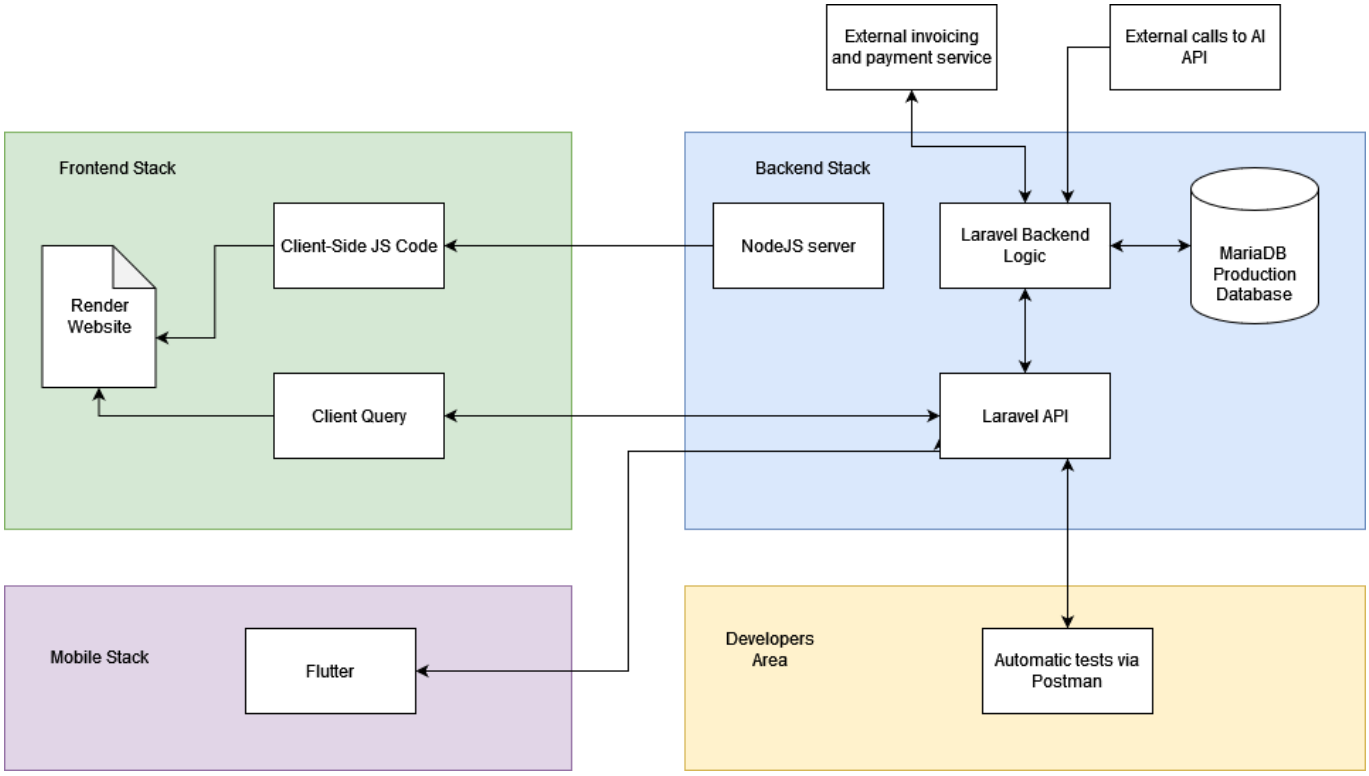
4. Ale Windows i Linux, a nawet ta sama dystrybucja Linuksa, mogą mieć inną konfigurację – na przykład inny zestaw pakietów – więc całe środowisko było czasami trudne do odtworzenia. Tak więc, aby środowisko aplikacji było odtwarzalne, a proces wdrażania bardziej skalowalny i automatyczny, potrzebny jest kolejny stos – DevOps – który zawiera co najmniej konteneryzację (taką jak Docker z lub bez Kubernetes lub LXC, warto również wspomnieć o Terraform) i CI / CD do automatyzacji. Teraz, w razie potrzeby,

można dodać do tego stosu sam hosting – taki jak AWS.

5. Do naszej obecnej analizy moglibyśmy dodać również stos Web3. Jest to dość nowy temat, ale już bardzo złożony i rozwinięty, ponieważ zapewnia zdecentralizowane połączenie między różnymi użytkownikami. Taką usługą może być na przykład Blockchain Wallet i Identity Layer.
6. I jest najnowsza warstwa, którą należy tutaj rozważyć – warstwa AI. Nie jest to tak nowy temat, jak można by się spodziewać – ponieważ za początek AI możemy uznać gdzieś w latach pięćdziesiątych XX wieku – ale teraz moc obliczeniowa i szybki postęp w jednocześnie przetwarzanych operacjach umożliwiły jej codzienne użytkowanie. Niektórzy mogą argumentować, że będzie to część stosu backendowego, ale moim zdaniem powinno to być traktowane oddzielnie.

Prawdopodobnie możemy się spodziewać, że w przyszłości stos technologiczny będzie coraz bardziej skomplikowany, ale można sobie tylko wyobrazić i zgadywać, co może być następne. Na przykład AR, a może coś, co wywodzi się z obliczeń kwantowych. Ale na razie zamknijmy tę listę dla rozważanego zakresu – aplikacji internetowych.

Stosy technologiczne a frameworki i narzędzia



Należy wyraźnie powiedzieć: stos technologiczny to nie to samo co framework, chociaż najczęstszym podejściem jest użycie jednego frameworka dla każdego stosu. Ma to wiele sensu, ponieważ wtedy wielu programistów pracuje w tym samym frameworku, który wykorzystuje tę samą logikę i „filozofię” w całym stosie.

Warto również zauważyć, że chociaż pisałem o różnych frameworkach Agile, NIE są to frameworki techniczne. Dla przypomnienia: Agile jest metodologią opartą na zestawie zasad przewodnich nakreślonych w Manifeście Agile, więc frameworki Agile są **ramami zarządzania projektami**. Agile zapewnia sposób zarządzania projektem i procesem rozwoju, ale nie narzuca żadnych konkretnych narzędzi, języków programowania ani struktur technicznych. Jest kompatybilny z różnymi frameworkami technicznymi, językami i narzędziami, ponieważ chodzi bardziej o podejście do sposobu planowania, iteracji i dostarczania pracy. Frameworki techniczne to zestawy bibliotek oprogramowania, narzędzi lub komponentów zaprojektowanych, aby pomóc programistom w bardziej wydajnym tworzeniu aplikacji, które zapewniają uporządkowany sposób pisania kodu i obsługi typowych zadań związanych z tworzeniem aplikacji.

Tak więc fameworks pod względem stosu technologicznego może być na przykład:

- Dla stosu frontendowego – ReactJS (który jest najczęściej używany przez Sailing Byte), Vue, Angular
- Dla backendu – Laravel (najczęściej używany przez Sailing Byte), NodeJS, Django, CodeIgniter
- Dla stosu devops – Kubernetes może być uważany za framework, ISPConfig również

Istnieją również **narzędzia i biblioteki**, które same w sobie nie są frameworkami, ale często (choć niekoniecznie!) są uważane za część frameworków, ale mogą być również używane w różnych frameworkach. Jako nietechniczny właściciel firmy najczęściej nie będziesz nimi w ogóle zainteresowany, ponieważ mogą, ale nie muszą być używane przez programistów do osiągnięcia celu i zaczynamy tutaj dotykać programistycznej części projektu. Takimi narzędziami są na przykład:

- Dla frontendu – Webpack, Babel
- Dla backendu – Composer, Artisan
- Dla devops – Docker, Apache, Nginx, GitLab CI/CD

Stos technologiczny i obowiązki programistów

Powróćmy do prostszego, nietechnicznego aspektu stosów technologicznych. Jest to konsekwencja wszystkich powyższych, ale aby nieco uporządkować wiedzę i upewnić się, że jesteśmy na tej samej stronie, nazwijmy to konkretnie. W zależności od tego, w którym stacku deweloper się specjalizuje, jego stanowisko będzie nazywane odpowiednio.

Full Tech Stack Developer

Pełny stos do tworzenia stron internetowych obejmuje kompletny zestaw technologii potrzebnych do zbudowania aplikacji internetowej dla danego projektu. Tak więc programista full stack to ktoś, kto potrafi obsługiwać WSZYSTKIE technologie związane z danym projektem. Jak przedstawiono powyżej, może to być całkiem sporo technologii.

W Sailing Byte uważamy, że programiści są bardziej wydajni, jeśli są ekspertami w jednym temacie, ale mają ogólną wiedzę na inne powiązane tematy. Dlatego nie pracujemy z programistami pełnego stosu, ale raczej z programistami specjalizującymi się w jednym stosie, którzy rozumieją, że stosy muszą być połączone między sobą, aby utworzyć całą funkcjonalną aplikację. Taka

specjalizacja nazywana jest „T-Shaped people”.

Front End Developerzy

Deweloperzy front-end są odpowiedzialni za wszystkie elementy strony internetowej, z którymi odbiorca wchodzi w interakcję. Obejmuje to grafikę, wygląd animacji, różne czcionki i układ. Jednym z najważniejszych elementów tworzenia stron internetowych, którym zarządzają programiści front-end, jest responsywne projektowanie stron internetowych. Określa on wygląd strony internetowej na różnych urządzeniach. Programiści front-end używają takich języków jak HTML (tak, wiem, HTML jest raczej strukturą danych niż językiem, ale nie wdawajmy się w takie szczegóły…) lub JavaScript.

Deweloperzy back-end

Kod strony internetowej jest tym, o co dba back-end web developer. Ci specjaliści od tworzenia stron internetowych zajmą się wszystkimi połączeniami z serwerem, bazami danych stron i funkcjonalnością wewnętrznej architektury stron internetowych. Najczęściej pracują oni w takich językach jak PHP, Python, Java czy MySQL.

DevOps

Osoba taka jest ogólnie odpowiedzialna za środowisko, w którym rezyduje aplikacja. Obejmuje to między innymi takie elementy jak architektura serwera, sam serwer, automatyzacja wdrażania, wirtualizacja, automatyczne skalowanie i tak dalej. Czyli ponownie – jest to stanowisko ściśle związane z technologiami takimi jak Docker, Linux, AWS, Kubernetes, LXC, GitLab CI/CD i tak dalej.

Inni specjaliści i deweloperzy

Łatwo zauważyć, że lista jest długa. Aby wymienić tylko kilka, możesz pomyśleć o **mobile developerze** specjalizującym się na przykład we Flutter lub **AI developerze** specjalizującym się w różnych LLM. Ale zasadniczo idea pozostaje ta sama – jeśli w projekcie wykorzystywany jest konkretny stos, który będzie rozwijany, to najprawdopodobniej będziesz potrzebował programisty, który specjalizuje się w tym stosie, zna frameworki i narzędzia, które są używane w takim stosie i jest w stanie pracować z nimi w taki sposób, aby jego „silos” skutecznie komunikował się z innymi stosami.

Dlaczego wybór odpowiedniego stosu technologicznego jest ważny?

Istnieje wiele powodów, dla których odpowiednio dobrany stos technologiczny jest ważny, takich jak wysokie bezpieczeństwo, funkcjonalność czy dostępność. Jednak z perspektywy biznesowej wyróżnia się jeden – skalowalność. Jest to związane ze zdolnością aplikacji internetowej do obsługi rosnącej liczby użytkowników. W końcu taki jest cel każdego oprogramowania wykorzystywanego w biznesie: przyciąganie coraz większej liczby użytkowników. Oprogramowanie tworzone przez twórców stron internetowych musi być tworzone z myślą o wzroście.

I być może zauważyłeś, że organizowanie silosów Development Tech Stack brzmi jak separacja, podczas gdy twoja aplikacja musi działać jako całość. Podczas gdy separacja może brzmieć źle, w rzeczywistości jest całkiem pomocna pod względem organizacji pracy, szczególnie w przypadku większych aplikacji. Stosy komunikujące się między sobą będą wymagały pewnego rodzaju komunikacji między sobą – na przykład między frontendem a backendem może

istnieć API. To API może być ustandaryzowane – a jeśli jest ustandaryzowane, może być testowalne – co zapewnia testowalność. Takie testy byłyby testami czarnej skrzynki, ponieważ nie zagłębiamy się w kod samego frontendu lub backendu, ale po prostu przeprowadzamy testy, czy komunikacja na punktach końcowych jest dobra w obie strony. Będzie to prawdą na przykład w przypadku aplikacji haedless, gdy backend udostępnia API, z którym frontend się łączy, a to API jest testowane na Postmanie, a jednocześnie możliwe jest podłączenie nowego frontendu (takiego jak Open API lub automatyzacja N8N/Zapier, a nawet aplikacja mobilna) bez odtwarzania całego backendu.

>

Z perspektywy biznesowej, wybór odpowiedniego stosu technologicznego jest również ważny dla dalszego zarządzania. Nie chciałbyś inwestować w coś, co stanie się przestarzałe (lub już jest przestarzałe!), ponieważ będzie to oznaczać ślepą uliczkę dla bezpieczeństwa aplikacji, ograniczoną liczbę dostępnych programistów, mniejszą zdolność do wdrażania nowych funkcji i

tak dalej. Zasadniczo, do rozwoju będziesz potrzebował software house'u z doświadczonymi programistami dla wybranego stosu technologicznego.

Przykładowe stosy technologiczne dla aplikacji mobilnych i aplikacji internetowych

Chociaż istnieją setki możliwych zestawów stosów technologicznych, niektóre są używane częściej niż inne i opiszmy kilka z nich, które są dość powszechnie używane – wraz z ich odmianami. Chociaż można się było spodziewać, że ta sekcja będzie oddzielna dla aplikacji webowych i oddzielna dla mobilnych, to tutaj wszystko będzie na jednej liście – to dlatego, że można rozszerzyć każdą aplikację używając innego stosu. Na przykład, można dodać Flutter do stosu LAMP, pod warunkiem, że PHP dostarcza niezbędne punkty końcowe.

LAMP i WAMP

Znaczenie – Linux/Windows + Apache + MySQL + PHP. Wspomniane już tutaj dwa stosy technologiczne, bardzo podobne do siebie – różnią się jedynie systemem operacyjnym. Jeśli są napisane poprawnie (i nie używają specyficznych funkcjonalności systemu operacyjnego), część stosu „AMP” może być przenoszona pomiędzy tymi dwoma. Kolejną częścią stosu, na którą warto zwrócić uwagę, jest „MySQL” – obecnie często zastępowany przez MariaDB. Jednak zarówno MySQL, jak i MariaDB są relacyjnymi bazami danych, a w niektórych projektach lepiej jest używać nierelacyjnych baz danych, takich jak NoSQL. W takim przypadku można nazwać taki stos LANP.

Stos MEAN

Nazwa składa się z pierwszych liter używanych technologii: MongoDB, Express.js, AngularJS oraz Node.js. Lubię jednak myśleć o tym stosie w bardziej nieformalnym znaczeniu *mean (dobry).* Cały stos może być napisany w JavaScript, a kod

może być kopiowany w całej aplikacji. To sprawia, że korzystanie z MEAN Stack jest proste jak bułka z masłem. Co więcej, jest on również darmowy i open-source oraz tworzy aplikacje charakteryzujące się wysoką elastycznością i skalowalnością. MEAN jest szeroko stosowany do dostarczania aplikacji internetowych.

Aplikacja mobilna wykorzystująca Firebase i Flutter

Ten stos wykorzystuje Flutter, wieloplatformowy zestaw narzędzi UI firmy Google, do tworzenia natywnie wyglądających aplikacji dla systemów iOS i Android z jednej bazy kodu. Dart jest podstawowym językiem Fluttera, wybranym ze względu na łatwość tworzenia ekspresyjnych interfejsów użytkownika i aplikacji o wysokiej wydajności. Firebase służy jako zaplecze, zapewniając podstawowe usługi, takie jak bazy danych w czasie rzeczywistym (Firestore), uwierzytelnianie, przechowywanie w chmurze i analizy. Bezserwerowe podejście Firebase upraszcza zarządzanie zapleczem, umożliwiając synchronizację danych w czasie rzeczywistym i

szybkie prototypowanie. Do zarządzania stanem można użyć Provider lub Bloc, aby zapewnić wydajną obsługę stanu w całej aplikacji. Ten stos jest idealny dla startupów i małych zespołów ze względu na szybką konfigurację, skalowalność i elastyczność zarówno w zakresie usług programistycznych, jak i backendowych.

Aplikacja mobilna + desktopowa wykorzystująca Electron

W tym stosie Electron jest podstawowym frameworkiem, umożliwiającym tworzenie wieloplatformowych aplikacji desktopowych przy użyciu technologii internetowych, takich jak JavaScript, HTML i CSS. Electron umożliwia tworzenie natywnych aplikacji desktopowych dla systemów Windows, macOS i Linux poprzez spakowanie aplikacji internetowej z silnikiem przeglądarki opartym na Chromium. W przypadku komponentu mobilnego, React Native lub Flutter można zintegrować z bazą kodu Electron, aby stworzyć ujednolicone podejście do kodu na różnych platformach. Node.js jest używany do procesów zaplecza w Electron, umożliwiając

aplikacji interakcję z systemem plików i innymi natywnymi zasobami. Ta konfiguracja może również integrować interfejsy API lub bazy danych, takie jak SQLite do lokalnego przechowywania danych lub MongoDB Atlas do obsługi baz danych w chmurze, umożliwiając płynny dostęp do danych na różnych urządzeniach.

Aplikacja webowa + mobilna z wykorzystaniem Laravel i AI

Ten stos wykorzystuje Laravel, framework PHP do tworzenia aplikacji internetowych z architekturą MVC, zarówno po stronie serwera aplikacji internetowych, jak i mobilnych. Laravel upraszcza uwierzytelnianie, routing i zarządzanie danymi za pomocą narzędzi takich jak Eloquent ORM do interakcji z bazą danych i Blade do tworzenia szablonów. W przypadku sztucznej inteligencji Python jest powszechnie używany do opracowywania modeli uczenia maszynowego, które są następnie eksponowane jako interfejsy API (np. przy użyciu Flask lub FastAPI) i konsumowane przez Laravel. Vue.js lub React

mogą być sparowane z Laravel w celu uzyskania dynamicznego, interaktywnego frontendu, zwłaszcza w wersji webowej. Flutter lub React Native mogą być używane w aplikacji mobilnej, łącząc się z interfejsami API Laravel' w celu uzyskania dostępu do danych i funkcji opartych na sztucznej inteligencji, takich jak silniki rekomendacji lub rozpoznawanie obrazów.

Stos technologiczny Web3 dla zdecentralizowanej aplikacji (dApp)

Na przykład prosta aplikacja dApp do zdecentralizowanego głosowania może wykorzystywać następujący stos:

- **Warstwa Blockchain: Ethereum dla konsensusu i obsługi transakcji.**
- **Smart Contracts: Solidity i Hardhat do pisania i wdrażania logiki głosowania.**
- **Zdecentralizowane przechowywanie: IPFS do przechowywania informacji o kandydatach i danych do głosowania.**
- **Frontend: React i Ethers.js do budowania interfejsu użytkownika i łączenia się z**

Ethereum.

- **Wallets: MetaMask do uwierzytelniania użytkowników i podpisywania transakcji.**
- **Oracles (jeśli wymagane): Chainlink do wprowadzania wszelkich danych spoza łańcucha (jeśli dotyczy).**
- **Usługi poza łańcuchem: Graf do wydajnego indeksowania i pobierania danych z łańcucha bloków.**

Twoi eksperci w tworzeniu aplikacji i stron internetowych

Jeśli znaczenie stosów, kwestie techniczne lub tworzenie stron internetowych wydają się trochę za dużo, Sailing Byte może Ci pomóc. Zarezerwuj telefon, aby porozmawiać o swoim projekcie lub już istniejącej stronie internetowej lub aplikacji. Mamy wieloletnie doświadczenie i praktykę w tworzeniu oprogramowania dla naszych klientów. Skontaktuj się z nami, pozwól nam działać, a zapewniamy, że nie pożałujesz.