Spis treści dla leniwych:

1. Opis projektu

2. Określenie wymagań

3. Architektura i zakres projektu

4. Wymagania techniczne

5. Wymagania dotyczące ruchu

6. Wymagania SEO

7. Architektura technologiczna

8. Wybór narzędzi promocji platformy (sekcja opcjonalna)

9. Wymagania graficzne

10. Tabela załączników

1. Opis projektu

Co znajduje się w tej sekcji?

Opis projektu, czyli inaczej krótki wstęp, wyjaśniający w kilku zdaniach do czego ma służyć platforma/aplikacja.

Przykład:

Celem portalu TimeExchange jest pośredniczenie w handlu usługami pomiędzy osobami prywatnymi. Użytkownik po zarejestrowaniu i zalogowaniu jest w stanie stworzyć własny profil, dzięki któremu może dodawać informacje o tym w czym się specjalizuje i jakich umiejętności poszukuje u innych poprzez wystawianie zleceń. Użytkownicy mogą wymieniać się wzajemnie swoim czasem; przykładowo: krawcowa szuka pomocy w księgowości. Za godzinę konsultacji z zakresu prowadzenia własnej działalności oferuje dwie godziny swojej pracy.

Aby móc nawiązać kontakt i/lub wystawić własne zlecenie, użytkownik musi wykupić abonament miesięczny na platformie. Głównym celem portalu jest więc pozyskiwanie ludzi z całego świata i konwertowanie ich na zarejestrowanych abonentów portalu.

2. Określenie wymagań

Co znajduje się w tej sekcji?

Badanie konkurencji i rynku (ocena rozwiązań i modeli biznesowych).

Wymagania biznesowe.

Cele platformy.

Określenie person i podstawowych elementów brandingu.

Nie istniejemy w próżni – projektowanie czegokolwiek w dobie kapitalizmu bez uprzedniego zbadania rynku i konkurencji, porównania dostępnych już rozwiązań oraz wyciągnięcia wniosków z błędów i sukcesów najlepszych w branży (lub w podobnych branżach – jeżeli znaleźliśmy całkowicie nową niszę) zwyczajnie mija się z celem. Dlaczego?

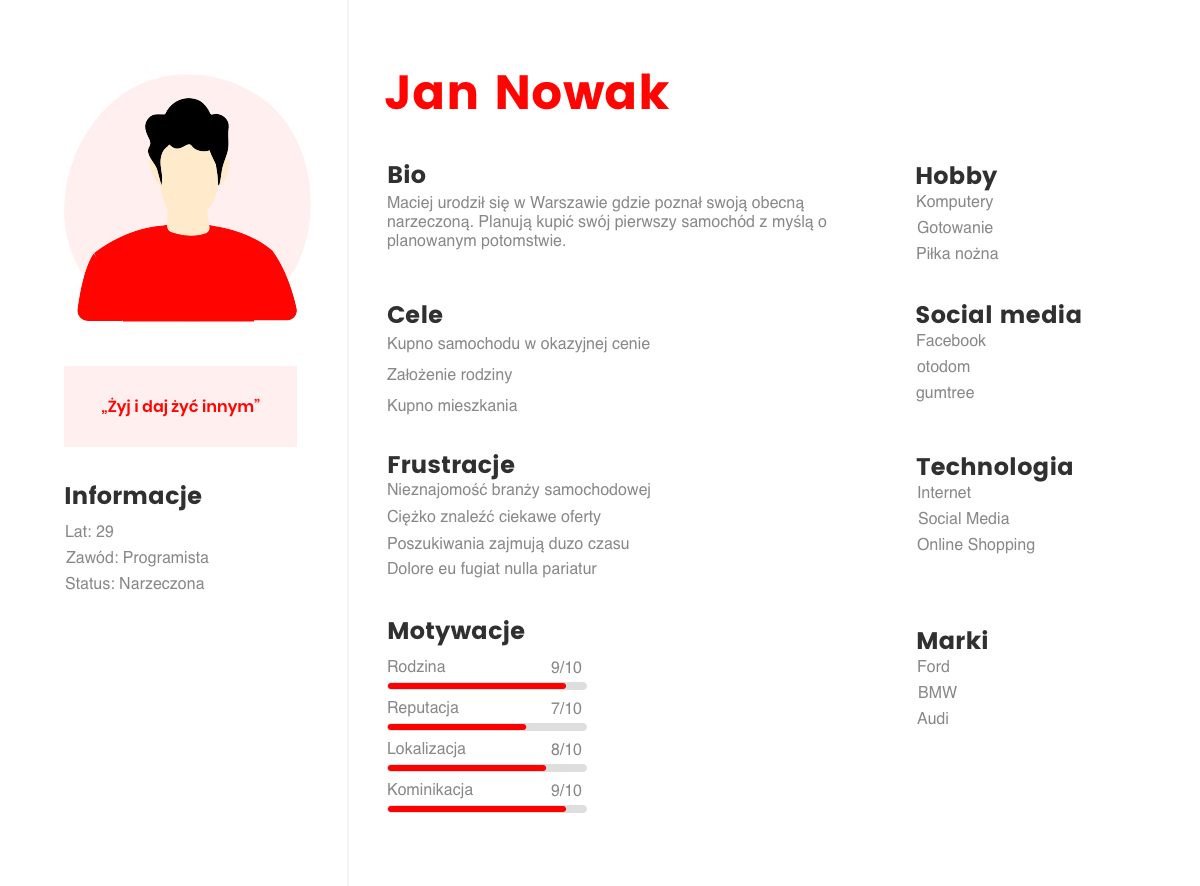
Odpowiedź jest niezwykle prosta: ponieważ może się okazać, że nasza wymarzona usługa lub produkt już istnieją. W takim wypadku, bez zaoferowania czegoś “ekstra” klientowi, ciężko oczekiwać, że zmieni on dotychczasowego dostawcę.

Nie bez znaczenia są też wnioski, jakie można wyciągnąć z badania konkurencji – ostatecznie fraza “uczyć się na błędach” nie wzięła się znikąd. A jak wiadomo, lepiej uczyć się na cudzych błędach, niż powielać je samemu (nie tylko w kontekście biznesowym).

Równie ważne jest badanie grupy odbiorców i ich potrzeb – w końcu to oni mają być klientem docelowym. Często jednak etap ten jest pomijany, a wnioski wysnuwane są na podstawie własnych przekonań, zamiast rzetelnych danych.

Rezultat? W zależności od branży, w pierwszej fazie wdrażania upada nawet 50-80% startupów. GUS także nie podnosi na duchu liczbami: z badań wynika, że w pierwszym roku od założenia upada ponad 30% nowo powstałych biznesów; kolejne 20% wytrzymuje na rynku tylko o rok dłużej.

Jeżeli jednak poprawnie przeprowadzimy wspomniane badania, stworzenie najważniejszych person oraz podstaw brandingu będzie znacznie łatwiejsze i obarczone o wiele mniejszym marginesem niedostosowania do potrzeb rynku.



Badania są pierwszym krokiem do określenia wymagań biznesowych i celów platformy/aplikacji – właśnie ten etap odróżnia proces ideacji (“wymyślania” nowego produktu/usługi), od pierwszego etapu wdrażania. Bez jasno rozpisanego planu biznesowego i realnych celów nawet najlepszy pomysł nie otrzyma dofinansowania na rozwój.

3. Architektura i zakres projektu

Co znajduje się w tej sekcji?

Architektura strony/aplikacji.

Mapa strony/aplikacji oraz ścieżki zakupowe/User Journey.

Lista szablonów strony/aplikacji oraz tabelaryczna lista z ogólnym opisem poszczególnych szablonów.

Opis zawartości kluczowych szablonów i modułów.

Opis funkcjonalności na poszczególnych ekranach.

Lista modułów do przygotowania i implementacji.

Lista niezbędnych funkcjonalności.

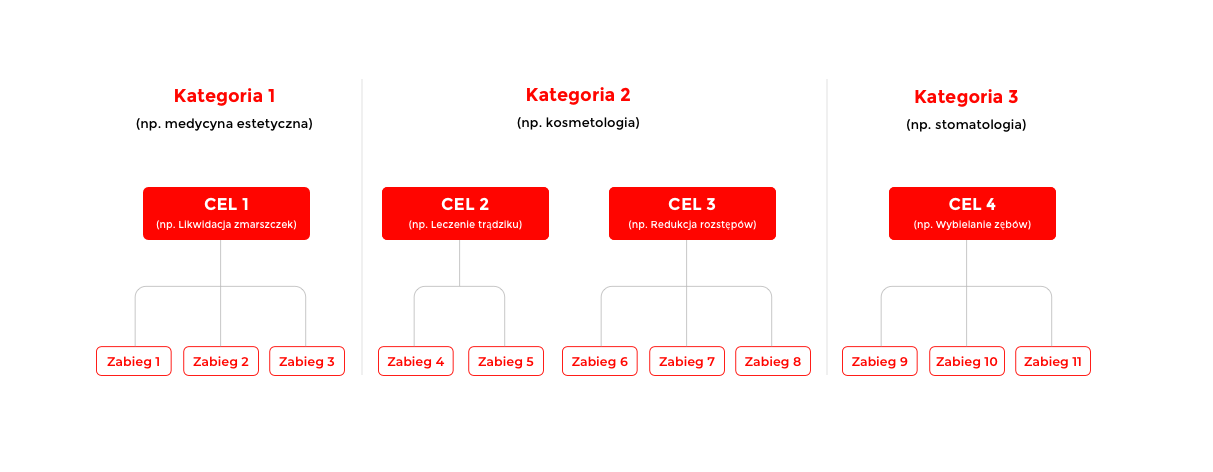
Tworzenie contentu – kto jest odpowiedzialny za tworzenie, wprowadzanie i zarządzanie contentem.

Wersje językowe.

Lista zewnętrznych integracji (CRM, emailing, Video hosting, Chat i inne).

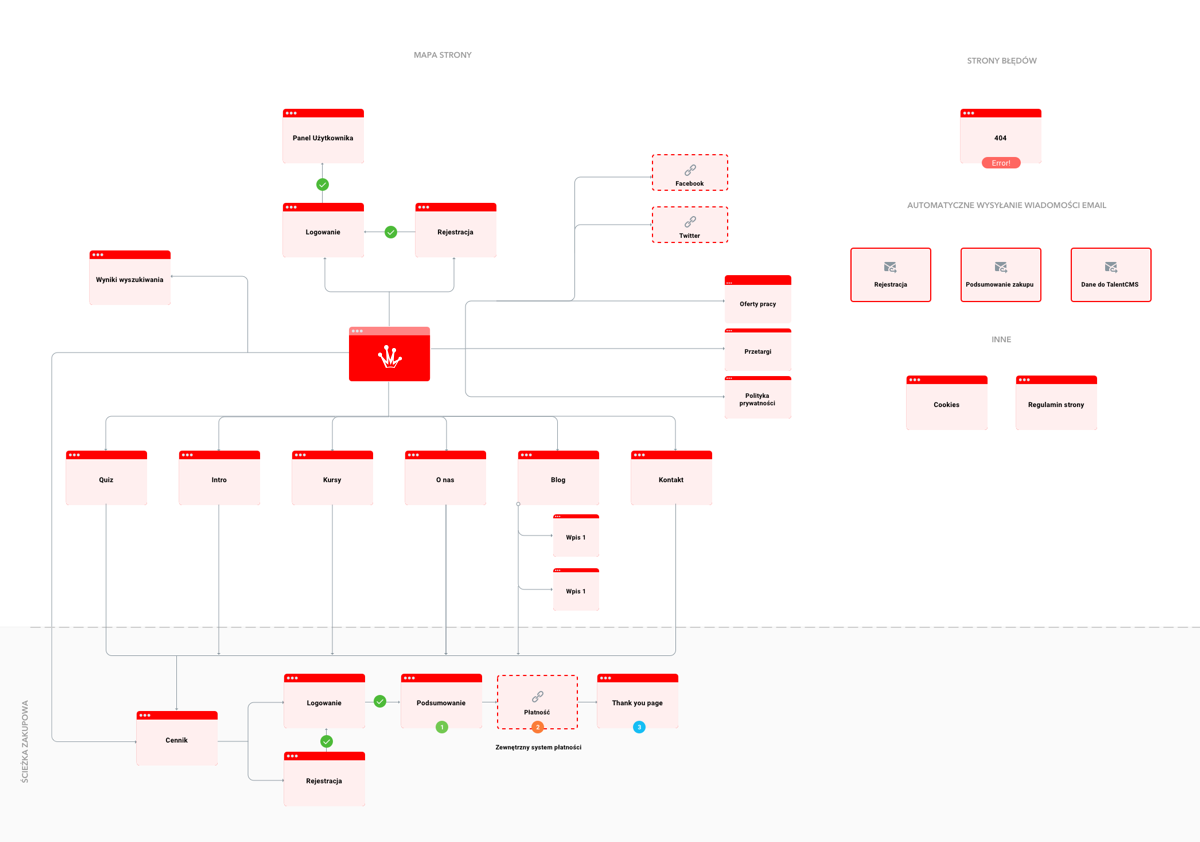
Najważniejsza część dokumentacji projektowej. Można by rzec – jej wnętrzności. Porządne opisanie elementów, które powinny się w niej znaleźć mogłoby posłużyć jako osobny wpis, postaramy się więc jak najlepiej podsumować jej główne cele.

Podobnie jak wspomniany wcześniej wymarzony dom, tak i aplikacja/platforma potrzebuje własnego planu architektonicznego. Jedyna różnica jest taka, że zamiast opisu co jest fundamentem, a co ścianą, otrzymujemy opis tego z jakich elementów będzie się składał produkt końcowy i w jaki sposób te elementy będą spełniały wyznaczone wcześniej cele biznesowe.

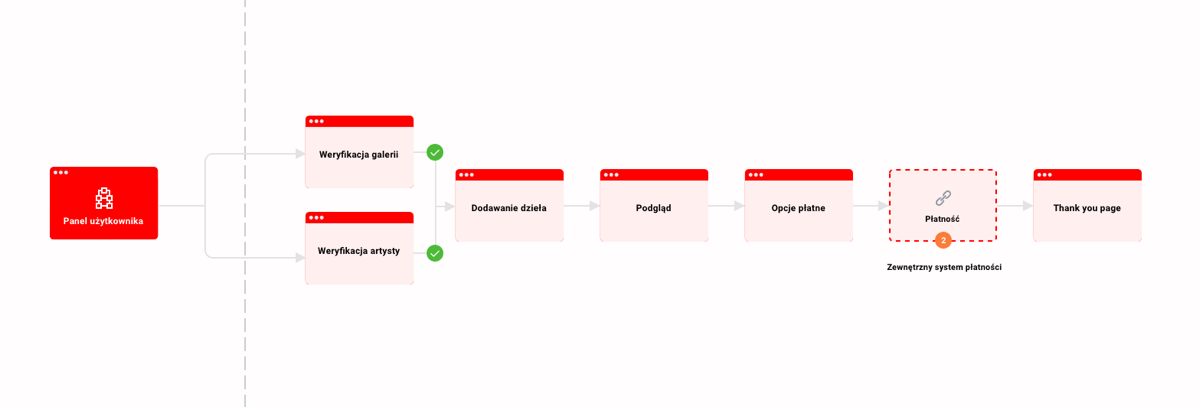


Kolejnym punktem na liście jest mapa oraz ścieżki zakupowe/User Journey.

Mapa zazwyczaj przypomina drzewo decyzyjne i jasno pokazuje wszystkie możliwe ścieżki i połączenia między poszczególnymi widokami; zaplanowanie tego na początku (przed wdrożeniem) oznacza mniejszą ilość poprawek na końcu (a tym samym mniejsze koszty aplikacji).



Ścieżki zakupowe i/lub User Journey sugerujemy zawsze opisywać na konkretnych przykładach z wykorzystaniem ustalonych wcześniej person. Zazwyczaj takich ścieżek tworzy się kilka, aby uwzględnić najważniejsze funkcje portalu/aplikacji i najbardziej prawdopodobne scenariusze interakcji. Sposób prezentacji takich danych może być różny: od diagramu, przez zwykły opis, po infografikę lub nawet komiks… wszystko po to, aby ostatecznie zarówno Klient, jak i wykonawca mieli takie samo wyobrażenie pożądanego efektu końcowego, ale także po to, aby upewnić się, że najważniejsze funkcjonalności i scenariusze zostały uwzględnione już na etapie planowania.



Następnie spisujemy listę koniecznych widoków, doprecyzowujemy szablony najważniejszych z nich oraz opisujemy ich zawartość. Pozostając przy porównaniu do budowy domu – jest to odpowiednik listy pokoi wraz z opisem przeznaczenia i zawartości każdego z nich; w tym elementów, które powinny pozostać spójne bez względu na pomieszczenie (jak np. kształt i kolor kontaktów oraz przełączników światła, które byłyby odpowiednikiem stopki lub górnej nawigacji platformy/aplikacji). Lista wymaganych widoków jest niezbędna do wdrożenia aplikacji/portalu – w innym wypadku wykonawca zwyczajnie nie wie ile/jakie widoki powinien zrealizować.

Na tym jednak nie poprzestajemy – przy analizie projektowej (tak samo jak przy projekcie budowy domu) panuje zasada: im więcej szczegółów i wskazówek odnośnie wymagań Klienta, tym lepiej. Dlatego też każdy widok powinien mieć dołączony opis funkcjonalności, które docelowo mają się na nim znaleźć.



W tej sekcji warto sprecyzować także najważniejsze wymagania odnośnie contentu: kto będzie odpowiedzialny za jego tworzenie, wprowadzanie i zarządzanie nim? Jeżeli content będzie w całości lub chociaż częściowo user-generated, to w jaki sposób powinno się nim zarządzać, dając jednocześnie użytkownikom jak najwięcej swobody? Ile i jakie wersje językowe będą potrzebne? Jak mają być realizowane – czy poza językiem ma się np. zmieniać także waluta? Jak mają wyglądać adresy URL i domena dla różnych wersji językowych? Czy konieczne będzie tłumaczenie contentu po oficjalnym starcie platformy/aplikacji i jeżeli tak, to kto będzie za to odpowiadał?

Całość zamyka lista podsumowująca najważniejsze integracje zewnętrzne, takie jak CRM (Customer Relationship Management tool / narzędzie do zarządzania relacjami z Klientem), chat, newsletter, video hosting i inne, wraz z opisem ich przeznaczenia.

4. Wymagania techniczne

Co znajduje się w tej sekcji?

Struktura adresów URL (określenie jakie mają być poszczególne linki przy różnych funkcjonalnościach).

Przekierowania (czy trzeba robić przekierowania istniejących adresów URL).

Określenie kluczowych breakpointów (na jakich rozmiarach ekranów aplikacja ma być sprawdzona oraz stworzenie tabelarycznej listy rozmiarów ekranów wraz z referencją do urządzenia).

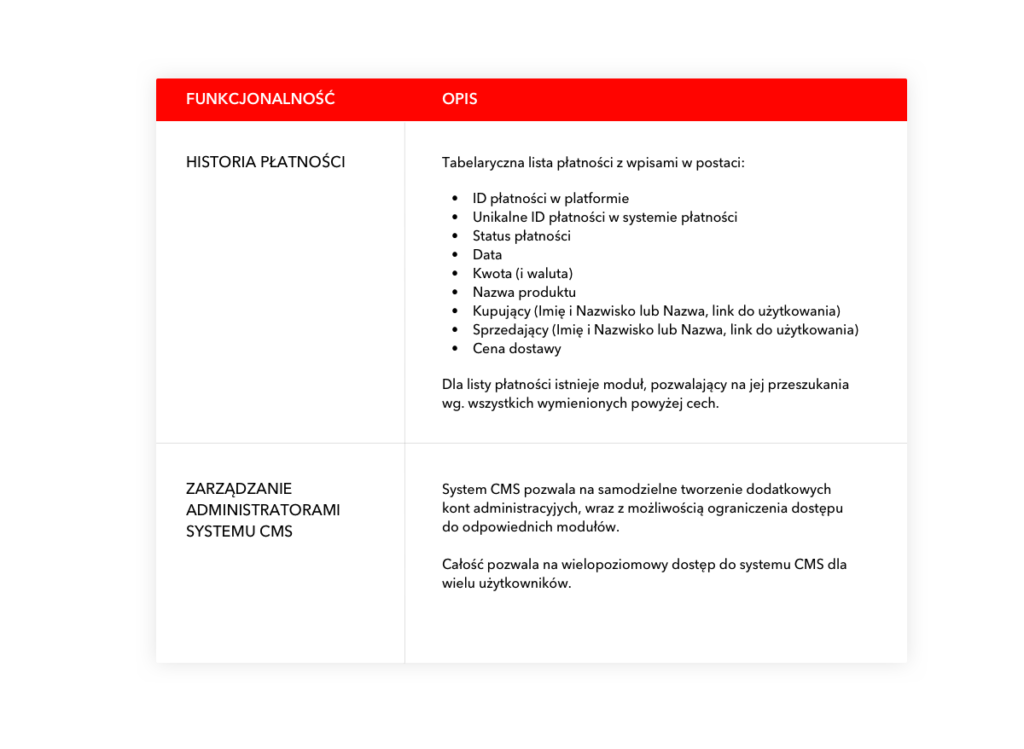
Wymagania dot. responsive design (RWD, Grid).

Wsparcie przeglądarek (lista przeglądarek dla których strona ma wyświetlać się poprawnie).

Opis modułu CMS i lista funkcjonalności.

Wymagania techniczne być może nie są najłatwiejszą decyzją (i lekturą) dla Klienta, ale z perspektywy samej platformy/aplikacji jest to część bardzo ważna. To właśnie tu znajdą się wytyczne odnośnie poprawnej budowy adresów URL wewnątrz platformy/aplikacji, zostaną doprecyzowane breakpointy (czyli inaczej: rozmiary ekranów, na jakich platforma/aplikacja ma być przetestowana przez wykonawcę), wspierane przeglądarki oraz inne wymagania odnośnie responsywności i inkluzywności (np. wersja kontrastowa dla niedowidzących). Te elementy często wpływają nie tylko na finalny dobór narzędzi, ale także na wybór języka całej aplikacji/platformy! O wiele rozsądniejsze, szybsze i tańsze jest ustalenie tych szczegółów na początku pracy projektowej, niż wprowadzanie zmian i łatanie kiepskich wyborów podczas lub nawet na koniec developmentu.

Dodatkowo, w tej sekcji powinien znaleźć się także opis modułu CMS (Content Management System – czyli inaczej: system zarządzania treścią) oraz lista jego funkcjonalności; przykładowo: które elementy gotowej platformy/aplikacji powinny być zawsze edytowalne z poziomu CMS, a które mogą być “zakodowane na twardo” (tj. będą wymagały bezpośrednich zmian w kodzie).



5. Wymagania dot. ruchu

Co znajduje się w tej sekcji?

Określenie jaki ruch musi być obsłużony przez aplikację (po wdrożeniu i rok/kilka lat od wdrożenia).

Performance (waga aplikacji i poszczególnych widoków).

Bardzo częstym błędem przy projektowaniu platformy/aplikacji jest nieprzygotowanie na duże wolumeny ruchu.

Oczywiście, w pierwszych tygodniach, a często nawet miesiącach istnienia platformy/aplikacji, ruch nie będzie przebijał sufitu. Wystarczy jednak jedna wzmianka na popularnym blogu lub stronie i może się zdarzyć, że w ciągu nocy ruch na naszej stronie wzrośnie z kilku tysięcy unikalnych użytkowników do kilkuset… a nasza infrastruktura tego nie udźwignie i strona dosłownie “utonie”.

A co się dzieje kiedy statystyczny użytkownik internetu natrafia na stronę, która nie działa?

Nigdy na nią nie wraca.

Właśnie dlatego tak ważne jest określenie jaki ruch musi być obsłużony przez platformę/aplikację po wdrożeniu i rok od wdrożenia. Dobrą praktyką jest też zastanowienie się nad celami ilościowymi na następne 5 lat, zakładając systematyczny i stabilny przyrost fanów.

W tej sekcji określa się ponadto tzw. performance, czyli wydajność. Nie raz, nie dwa spotkaliśmy się z sytuacją, w której z powodu braku przemyślanych rozwiązań na początku procesu, konieczne było wstrzymanie prac developerskich nad startupem na rzecz poprawy wydajności wdrożonych już elementów. Efekt? Bardzo częsta i nagła śmierć owych startupów, ze względu na nieprzewidziane koszty.

Aby nie powtarzać tego błędu, zawsze zwracamy naszym Klientom uwagę na określenie chociażby wstępnych wymagań dotyczących performance: ustalenie minimalnej liczby punktów w bezpłatnym narzędziu Google Page Speed czy wartości granicznej czasu wyświetlenia dowolnej strony serwisu oraz wypisanie “dobrych praktyk”, takich jak optymalizacja obrazów, kompresja tekstów itd.

6. Wymagania SEO

Co znajduje się w tej sekcji?

Opis znaczników  (lista edytowalnych elementów niezbędnych do edycji wraz z opisem i sprecyzowanym źródłem treści).

Wybór narzędzi analitycznych (Google Analytics, HotJar, Yandex itd.).

Wskazanie celów do analizy.

Sekcję tę otwiera lista znaczników oraz metatagów, które powinny być edytowalne, wraz z wytycznymi odnośnie sposobu ich realizacji i tego kto ma dostarczać ich treść. Poniżej trzy przykłady (choć jest ich oczywiście znacznie więcej):

Title – tytuł strony (zawartość znacznika <title> w sekcji <head> kodu źródłowego strony), czyli najważniejszy metatag. Wyświetla się w wynikach wyszukiwania.

Robots – tag umożliwiający kontrolę sposobu indeksowania i wyświetlania strony w wynikach wyszukiwania.

Tagi social media – tagi umożliwiające komponowanie treści w udostępnianych postach.

Niechęć do myślenia nad analityką na etapie projektowym jest o tyle ciekawa, że spora część narzędzi analitycznych jest bezpłatna – jedynym kosztem jaki musimy więc ponieść jest ich odpowiednie wdrożenie na samym początku. To, czy, kiedy i jak wykorzystamy zbierane dane zależy potem już tylko od nas samych.

Nie wspominając już o tym, że bez odpowiednio przemyślanej analityki ciężko udowodnić inwestorom osiągnięcie założonych w biznesplanie celów… 😉

7. Architektura technologiczna

Co znajduje się w tej sekcji?

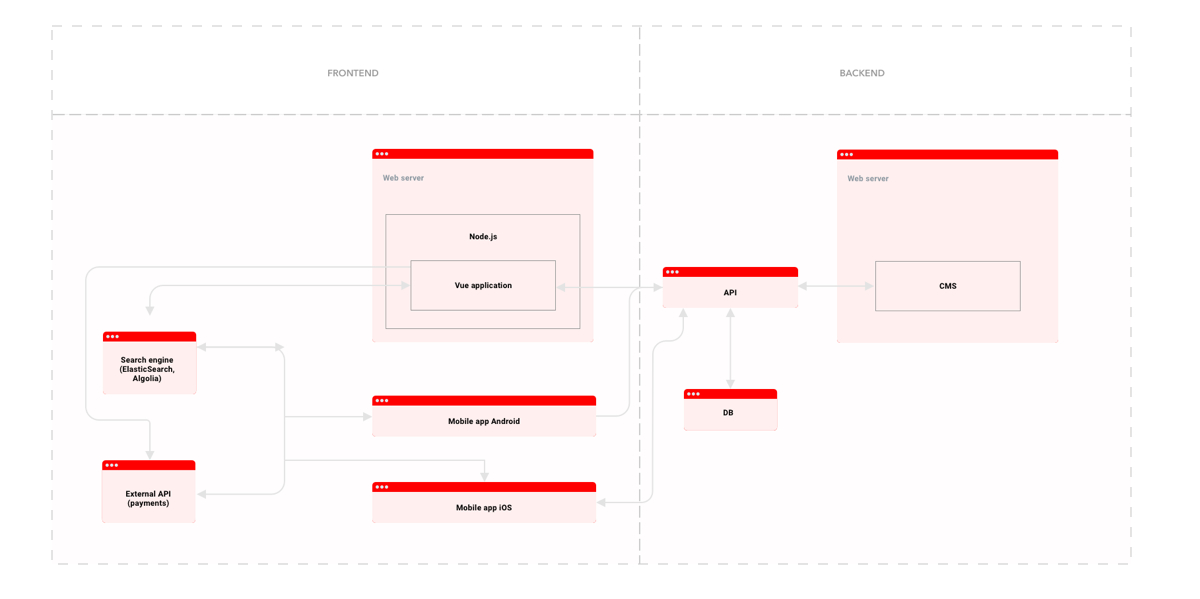
Wybór platformy technologicznej.

Opis środowiska produkcyjnego i testowego oraz repozytorium.

Transfer usług (jeżeli dotyczy).

Wymagania techniczne dotyczące serwera.

Sekcja architektury technologicznej ma za zadanie jak najlepsze rozplanowanie technicznej realizacji projektu – od jego bezpieczeństwa, przez sposób wdrożenia kluczowych funkcjonalności, utrzymanie i rozwój aplikacji/platformy, ale też dobranie najlepszych narzędzi do osiągnięcia opisanych w poprzednich rozdziałach celów.



Wybór platformy musi uwzględniać ruch opisany w sekcji 5 (Wymagania dot. ruchu), a jednocześnie zapewnić aplikacji/platformie jak największe bezpieczeństwo i łatwość developmentu. Ważnym elementem jest tutaj także dług technologiczny. Czym on jest?

Przez dług technologiczny rozumiemy ograniczenia, jakie narzucają nam konkretne rozwiązania programistyczne. To właśnie podczas tworzenia dokumentacji projektowej powinniśmy znaleźć odpowiedzi na pytania: czy wybrana przez nas technologia jest rozwijana i będzie rozwijana w przyszłości? Czy są jakiekolwiek przesłanki świadczące o tym, że technologia ta ulegnie poważnym zmianom w niedługim czasie? A może już teraz widać, że technologia “umiera” i nie posiada aktywnego drona developerów, którzy ją rozwijają?

Dług technologiczny powstaje zawsze – ponieważ każda technologia ewoluuje – można go jednak niepotrzebnie powiększyć już na starcie decydując się na wykonanie aplikacji/platformy “na wczoraj” i najlepiej po kosztach.

Przykład:

Firma X tworzy platformę w 2011 roku, korzystając z konkretnego środowiska (np. CodeIgniter, wersja 2.1.0). Decyduje się na sieciowy framework (np. Gearman) do realizacji zadań (m.in. wykonywania połączeń i wysyłania powiadomień, ale także np. do rejestracji nowych użytkowników) oraz inne integracje zewnętrzne.

Kilka lat później na rynku pojawiają się nowe wersje zarówno wykorzystanego środowiska jak i frameworka; firma jednak nie podejmuje się aktualizacji ze względu na koszta i obawę o problemy mogące wystąpić w trakcie zmian. Kolejne wersje obu narzędzi także są pomijane.

W 2017 roku framework sieciowy kończy oficjalne wsparcie dla swojego produktu, pozostawiając go tym samym z ogromnymi dziurami w bezpieczeństwie (stale rosnącymi) i coraz większym niedostosowaniem do nowszych narzędzi. W rezultacie znacznie wzrastają koszta utrzymania bieżących rozwiązań firmy X – każdy nowy problem trzeba rozwiązywać własnymi zasobami, a developerów mających chęci i doświadczenie w pracy z martwym produktem jest coraz mniej.

Jednocześnie, brak aktualizacji wybranego środowiska powiększa przepaść między platformą, a integracjami zewnętrznymi (które podlegają automatycznym aktualizacjom, niezależnym od firmy X). Z każdym dniem dług technologiczny rośnie, a przy każdej takiej aktualizacji pojawia się coraz więcej problemów z integracjami.

Efekt? Ze względu na przestarzałą architekturę platforma jest niezabezpieczona, nie spełnia wymogów GDPR, a jej podstawowe funkcjonalności przestają działać. Firma X nie może znaleźć developera, który zgodziłby się pracować w starym środowisku i na skutek tego zaczyna uginać się pod ciężarem długu technologicznego. W rezultacie traci kolejne okazje biznesowe i ostatecznie jest zmuszona przepisać całą platformę od nowa (czyt. ponieść koszta rozległych prac projektowych i developerskich).

Repozytorium oraz środowisko testowe i produkcyjne odpowiadają z kolei za to jak będzie wyglądał bieżący rozwój produktu (w jaki sposób developerzy mogę edytować/dodawać kod, jak dostarczane będą kolejne wersje na deploy). Warto już w tym momencie zdecydować się na jedną z opcji dodawania zmian: continuous integration lub dodawanie zmian manualnie, przy pomocy dodatkowych zasobów (np. admina).

Transfer usług (jeżeli dotyczy, tj. jest to nowa wersja istniejącego już produktu) może wiązać się z dodatkowymi kosztami (ze względu na niekompatybilność usług); dlatego warto sprawdzić to już na etapie projektowania i uwzględnić w kosztach developmentu.

Wymagania techniczne serwera mają za zadanie upewnienie się, że serwer (usługa) spełnia wymogi wybranego softu (np. potrzebujemy bazy danych X – czy dostawca świadczy taką usługę?).

8. Wybór narzędzi promocji platformy (sekcja opcjonalna)

Co znajduje się w tej sekcji?

Doradztwo w zakresie wyboru kanałów promocji z Inbound i Outbound.

Wskazanie konkretnych kanałów wraz z opisem i wymaganiami.

Propozycje dalszych kierunków rozwoju.

W idealnym świecie, sekcja dotycząca narzędzi promocji jest oczywistym przedłużeniem dokumentacji projektowej. To właśnie tu określa się podstawowe elementy przyszłych kampanii marketingowych, czyli kanały Inbound (aktywny marketing przychodzący; np. SEO, SEM, SERM, media społecznościowe, mikroblogi i blogi) i Outbound (push marketing; np. call center, akcje promocyjne w sklepach, badania telefoniczne, kampanie mailingowe), ustala się rodzaj treści jakie będą potrzebne oraz źródło ich pozyskiwania, a także wyznacza przyszłościowe kierunki rozwoju strategii marketingowej.

Dzięki chociaż pobieżnemu rozpisaniu strategii marketingowej jesteśmy w stanie określić czy np. częścią platformy powinien być blog. Jeżeli tak, to kto i w jaki sposób będzie go prowadził? O czym będzie pisał? Przygotowanie planu komunikacji na tym etapie jest najlepszym sposobem na ustrzeżenie się przed najczęściej spotykaną marketingową kulą u nogi: niedostosowanymi do brandu (a tym samym martwymi) kanałami komunikacyjnymi marki, które straszą brakiem odwiedzin i rzeczowego contentu.

Ma to jednak także znaczenie z perspektywy developmentu. Pozostając przy przykładzie bloga: jeżeli już na etapie planowania architektury aplikacji/platformy wiemy, że w niedalekiej przyszłości jej częścią powinien być blog, to możemy od razu dostosować odpowiednio środowisko i narzędzia, aby uniknąć problemu “doklejania” bloga na siłę do istniejącej już infrastruktury.

9. Wymagania graficzne

Co znajduje się w tej sekcji?

Zebranie i spisanie wytycznych dot. wyglądu grafiki (brandingu).

Lista materiałów (istniejące materiały klienta).

Benchmarki (przedstawienie i zebranie tabelarycznie listy benchmarków ze wskazaniem najważniejszych cech).

Wybór czcionek (kreślenie skąd pobierane będą czcionki – google fonts, myfonts, dostarczone przez klienta – oraz ich wersji językowych).

Makietowanie (makiety, wireframe).

Dlatego w tej sekcji powinny znaleźć się wszelkie informacje i wymagania dotyczące istniejącego już (lub planowanego) brandingu marki: firmowe kolory, czcionki i fonty, logo i jego użycie, ale także m.in. Tone of Voice (“ton głosu” marki, czyli przykłady tego w jaki sposób powinny być sformułowane wszelkie komunikaty – czy marka mówi o sobie jako “ja”, czy “my”?). Tu dorzuca się benchmarki (przykłady tego co Klientowi podoba się u innych i co chciałby wykorzystać jako inspirację do własnego projektu, a czego na pewno chce uniknąć) oraz makiety.

Makietowanie (z ang. mock-up), czyli inaczej przygotowane w dowolnej formie szkice aplikacji/platformy (w praktyce często ograniczane do najważniejszych widoków i funkcjonalności). Podstawową zasadą obowiązująca przy makietowaniu jest ich czytelność – powinny być zrozumiałe dla każdego członka zespołu projektowego (w tym Klienta), ponieważ docelowo mają spełniać rolę.

10. Tabela załączników

Jak nazwa wskazuje – w tej sekcji powinien znaleźć się spis wszystkich załączników (grafik, plików, dokumentów), które znalazły się w dokumentacji i/lub są niezbędne do jej pełnego zrozumienia (jak np. brandbook marki).

Dlaczego dokumentacja projektowa jest tak ważna?

Pomaga w planowaniu projektu – zarówno na etapie początkowej ideacji, jak i w kontekście planowania prac wdrożeniowych.

Pomaga w estymacji czasu i potencjalnych kosztów pracy, a także zapewnia większą kontrolę nad budżetem w trakcie prac wdrożeniowych.

Umożliwia ustalenie realnych celów projektu.

Pomaga w lepszym planowaniu i przydzielaniu zasobów.

Umożliwia śledzenie postępów oraz zapewnia lepszą kontrolę nad statusem projektu.

Ułatwia wprowadzanie zmian już na etapie wdrożeniowym.

Przygotowuje na ryzyko i zmniejsza szanse na niepowodzenie oraz nieprzewidziane przeszkody w realizacji projektu.

Ułatwia bezproblemową komunikację pomiędzy Klientem a wykonawcą.

Ułatwia zarządzanie projektem po stronie Klienta i wykonawcy.

Pomaga zdobyć wsparcie od sponsorów.

Pomaga (w przyszłości) wdrażać pracowników do projektu.

Stanowi podstawę do odbioru prac wykonanych przez wykonawcę (pozwala łatwo dostrzec ewentualne braki/rozminięcia się z projektem).

Jest wiążącym prawnie dokumentem (może być podstawą do oficjalnego dochodzenia swoich praw w wypadku niedokończonego projektu/projektu niezgodnego z ustaleniami).

Dobrze przygotowana dokumentacja jest elementem obowiązkowym każdego większego projektu. Pomaga ona w zarządzaniu budżetem, zasobami i czasem zarówno Klientowi, jak i wykonawcy; jest również niezwykle użyteczna w utrzymywaniu projektów w szybkim tempie. Znacznie ułatwia komunikację, marginalizuje szanse na niedopowiedzenia lub nieprzemyślane rozwiązania, a tym samym pozwala zaoszczędzić mnóstwo pieniędzy. Co jednak najważniejsze – dokumentacja projektowa jest prawnie wiążącym dokumentem, który może okazać się nieocenioną pomocą w sytuacji jakichkolwiek problemów przy odbiorze projektu.