Brief na wykonanie usługi informatycznej

Sprawna realizacja wdrożenia informatycznego - oprócz wiedzy, know-how i zaangażowania - w dużej mierze zależy od wiedzy wykonawcy nt celów biznesowych klienta, przeznaczenia oprogramowania, środowiska biznesowego czy planowanych integracji z innymi systemami IT.

Pod patronatem Izby Gospodarki Elektronicznej powstał dla Państwa przewodnik – wzór briefu technologicznego, który pozwoli na łatwiejsze i sprawniejsze rozpoczęcie prac nad nowym wdrożeniem informatycznym. To także podpowiedź, jak bardzo złożonym procesem jest tworzenie oprogramowania i w jaki sposób nieodpowiednio przygotowane zapytanie ofertowe może powodować bardzo dużą rozbieżność wycen od potencjalnych wykonawców.

Należy mieć na uwadze, że wybór niektórych opcji może powodować wyższe koszty wdrożenia oprogramowanie. Jednak ma to również wymierny, korzystny wpływ na bezpieczeństwo, stabilność i możliwość rozwoju kodu. Rekomendujemy kierować się zasadą, że zawsze należy wpisać czy decydujemy się lub nie na dany element wdrożenia. Wynika to z różnicy domyślnych praktyk stosowanych przez potencjalnych wykonawców w zakresie wyceny oprogramowania. Zdarza się, że brak danych spowoduje olbrzymią dysproporcję otrzymywanych wycen, co jedynie utrudni wybór optymalnej oferty.

Zapraszamy do korzystania i mamy nadzieję, że ten dokument pomoże Państwu we współpracy z dostawcami rozwiązań informatycznych.

1. Wprowadzenie

Na start wpisz kilka zdań, które w skrócie przedstawią koncepcję oczekiwanego wdrożenia IT.

1.1. Słownik

Wylistuj wszelkiego typu:

- słownictwo branżowe wraz z opisem wyjaśniającym,
- pojęcia branżowe zawierające definicję,
- rozwinięcia skrótów,

używane do opisu projektu. Dzięki temu potencjalny wykonawca pozna Twoje środowisko biznesowe.

1.2. Aktorzy

To osoby lub systemy, które będą wykorzystywały funkcjonalności nowego rozwiązania IT, np. Użytkownik niezalogowany, Użytkownik zalogowany, Moderator treści, Administrator, System obsługi płatności itp.

2. Opis przedmiotu zamówienia

Rozdział generalnie powinien zawierać ogólne informacje o projekcie.

2.1. Cel projektu

Cel biznesowy projektu jaki ma zostać osiągnięty w wyniku planowanego wdrożenia informatycznego.

2.2. Harmonogram

Oczekiwany plan czasowy realizacji projektu jaki powinien lub musi być przestrzegany. Jeśli projekt nie posiada harmonogramu, rozdział powinien zawierać informację o jego braku.

2.3. Otoczenie biznesowe

Określ istniejące otoczenie biznesowe nowego projektu. Poprzez otoczenie biznesowe rozumie się wszelkiego typu inne systemy, które są aktualnie wykorzystywane przez zamawiającego, które będą w jakimkolwiek stopniu powiązane z planowanym wdrożeniem. Warto określić jakie to systemy (np. księgowy, CRM, ERP, produkcyjny, rozliczeniowy) i/lub w jaki sposób mają być ze sobą powiązane. Nie trzeba uwzględniać systemów IT, które dopiero będą wykorzystywane ze względu na realizację zamawianego projektu.

2.4. Wymagania technologiczne

Wszelkiego typu wymagania odnośnie technologii realizacji projektu - np. system musi być implementowany w języku JAVA, a baza danych to MySQL. Jeśli nie ma sprecyzowanych wymagań zostaną one zasugerowane.

2.5. Wersje językowe

Dodaj informację o tym jakie wersje językowe system musi zawierać przy pierwszym uruchomieniu oraz czy i o jakie języki jest planowana rozbudowa w przyszłości.

2.6. Migracja danych

Odpowiedz na pytanie: czy do systemu mają być inicjująco migrowane jakiekolwiek dane z innego lub innych systemów? Jeśli żadne dane nie będą migrowane, należy tę informację również zamieścić.

2.7. Integracja z systemami zewnętrznymi

Lista systemów z jakimi oczekiwane rozwiązanie ma się integrować - informacje o API lub innych metodach integracji. W rozdziale nie trzeba opisywać API, wystarczy informacja jaki to system i skąd można uzyskać dostęp do dokumentacji.

3. Wymagania funkcjonalne

Główny rozdział zawierający wszelką posiadaną wiedzę o planowanej funkcjonalności systemu.

3.1. Funkcjonalności front end (dla użytkownika)

Znane funkcjonalności określane jako front-end - czyli funkcjonalności, z których korzysta użytkownik końcowy za pomocą dostępnego interfejsu.

3.2. Funkcjonalności back end (dla administratora)

Znane funkcjonalności określane jako back-end - czyli funkcjonalności, z których korzysta określony pracownik w celu konfiguracji systemu, sprawdzania jego działania itp.

3.3. Model uprawnień

Informacje na temat modelu uprawnień jaki należy przyjąć przy projektowaniu systemu. Modele można podzielić na 3 główne grupy:

- Uprawnienia permanentne – ustalone na etapie specyfikacji rozwiązania, jakakolwiek zmiana uprawnień dla określonego typu użytkownika związana jest z koniecznością modyfikacji kodu systemu przez programistę. Model najprostszy, a tym samym najszybszy w implementacji i najtańszy w wytworzeniu, ale wymaga każdorazowej ingerencji programisty przy chęci jakiejkolwiek modyfikacji.
- Uprawnienia w oparciu o role – na etapie specyfikacji projektu ustalane są role jakie mogą pełnić użytkownicy w systemie. Każdemu użytkownikowi można przypisać wiele ról, a każda z nich posiada zdefiniowane uprawnienia. Administrator ma możliwość modyfikacji powiązania użytkowników z rolami, a tym samym samodzielnego przyznawania określonych uprawnień wybranym użytkownikom. Role posiadają permanentnie przypisane uprawnienia i jakakolwiek zmiana uprawnień przypisanych do roli związana jest z koniecznością modyfikacji kodu systemu. Model średnio złożony, czas implementacji dłuższy niż w poprzednim modelu, co powoduje wzrost kosztów wytworzenia, ale możliwa jest ograniczona konfiguracja uprawnień bez udziału programisty.
- Uprawnienia dynamiczne – najbardziej złożony model uprawnień, działający analogicznie jak uprawnienia w oparciu o role, z tą różnicą że administrator systemu ma możliwość tworzenia nowych ról oraz przypisywania poszczególnych uprawnień do ról. Tym samym administrator może dowolnie modyfikować uprawnienia i dowolnie przydzielać je użytkownikom systemu. Ingerencja programisty nie jest potrzebna. Najbardziej złożony model uprawnień, wydłużony proces implementacji czyni model najbardziej kosztownym, ale umożliwia dowolne zmiany w przydzielaniu uprawnień bez ingerencji programisty w kod systemu już po oddaniu systemu do użytku.

3.4. Typy raportów

Informacje na temat oczekiwanych danych w postaci raportów, statystyk lub innej formy agregacji i prezentacji danych. Poprzez wszelkiego typu informacje rozumie się wszystkie wymagania co do tego, jakie dane mają być prezentowane, w jaki sposób i jakim użytkownikom. Im dokładniej zostaną opisane wymagania raportów, tym bardziej precyzyjnie da się określić pracochłonność takiego rozwiązania. Rozdział może zawierać przykłady, odwołania czy schematy koncepcyjne.

4. Wymagania niefunkcjonalne

Wymagania, które nie są funkcjonalnościami systemu, np. system powinien być łatwo skalowalny, strona główna systemu powinna się ładować w czasie poniżej 3 sekund, itp.

4.1. Projekty graficzne

Dodaj projekty graficzne jeśli są już przygotowane lub wskazówki do opracowania projektu graficznego, np. przykłady interfejsów stron, które się podobają. Dodatkowo jeśli istnieją - przekaż np. księgę znaku oraz materiały niezbędne do opracowania projektu graficznego oraz jego implementacji. Warto zastanowić się nad przeprowadzeniem badań interfejsu z udziałem grupy kontrolnej.

5. Wymagania przebiegu testów

Rozdział powinien zawierać wszelkiego typu wymagania związane z jakimkolwiek testowaniem systemu, np.:

- czy należy wykonywać „code review”,
- czy należy pisać testy jednostkowe,
- czy system ma być testowany manualnie czy automatycznie,
- czy należy stosować „code freeze” itp.

Każde dodatkowe wymaganie doprecyzowuje sposób realizacji projektu. Mogą one mieć wpływ na zwiększenie wyceny wstępnej u potencjalnych wykonawców, którzy nie stosują domyślnie tych metod poprawy jakości. Należy pamiętać, że jeśli planowany system ma nam służyć przez długi okres czasu i podlegać rozwojowi, to środki zaoszczędzone na jakości w etapie pierwszym mogą w kolejnych etapach podnosić koszty prac rozwojowych.

6. Wymagania bezpieczeństwa

Określ wymagania związane z bezpieczeństwem systemu oraz danych, np.:

- czy system ma być dostępny tylko po zalogowaniu się użytkownika,
- czy połączenie ma być szyfrowane
- czy system ma być dostępny przez internet, czy tylko z sieci lokalnej (lub VPN),
- czy system ma przejść testy penetracyjne, itp.

Każde dodatkowe wymaganie doprecyzowuje sposób realizacji projektu.

7. Wymagania wdrożeniowe

Jakie są wymagania związane z wdrożeniem systemu, np.:

- czy wdrożenie systemu ma być realizowane przez wykonawcę,
- czy mają być przygotowane materiały wdrożeniowe,
- czy wykonawca ma zagwarantować migrację danych,

- jakie jest oczekiwane wsparcie w okresie bliskim wdrożenia (np. helpdesk w godz. pracujących, inne) itp.

Każde dodatkowe wymaganie doprecyzowuje sposób realizacji projektu.

8. Wymagania szkoleniowe

Jeśli to zostało określone, dodaj wymagania dotyczące szkolenia z obsługi systemu, np.:

- czy ma być wykonane szkolenie z obsługi systemu,
- gdzie ma się odbyć szkolenie,
- ile czasu ma trwać szkolenie,
- czy mają zostać przygotowane materiały szkoleniowe z obsługi systemu, itp.

Każde dodatkowe wymaganie doprecyzowuje sposób realizacji projektu.

9. Wymagania utrzymaniowe

Rozdział powinien zawierać wszelkiego typu wymagania związane z utrzymaniem systemu, np.

- Jaki ma być poziom SLA,
- czy przewiduje się rozwój systemu, jeśli tak to orientacyjnie w jakim stopniu, itp.

Wymagania utrzymaniowe są oddzielnym zagadnieniem nie mającym wpływu na realizację i koszt samego projektu. Są ustalane osobno i często określone inną umową z wykonawcą.

10. Wymagania odnośnie umowy realizacyjnej

Wszelkie warunki wymagane względem umowy realizacyjnej, w szczególności:

- Czy kupowane są prawa autorskie czy licencja, a jeśli licencja to na jaki okres czasu (bezterminowa zgodnie z prawem kończy się po 5 latach), warto pamiętać, że trudno będzie uzyskać prawa autorskie do 100% kupowanego kodu ponieważ wykluczałoby to możliwość zastosowania w programowaniu szeregu narzędzi dostępnych na zasadach licencji otwartych.
- Harmonogram realizacji i daty końcowe
- Oczekiwane parametry obsługi gwarancyjnej – czas reakcji, czas obejścia problemu, czas usunięcia problemu, typy błędów (krytyczny, poważny, drobny).

Słownik pojęć

Użytkownik końcowy – jest to osoba, która ostatecznie będzie korzystała z systemu, czyli np. pracownik firmy w przypadku projektów wspierających pracę przedsiębiorstwa, klient sklepu internetowego w przypadku projektu sklepu itd.

System skalowalny – cecha systemu określająca możliwości zwiększenia wydajności systemu poprzez zwiększenie zasobów fizycznych (serwerów).

SLA – (ang. Service Level Agreement) czyli umowa o gwarantowanym poziomie świadczenia usług stosowana głównie w celu określenia w gwarantowany czas w jakim zostanie naprawiony błąd określonej kategorii.

Szyfrowane połączenie – protokół SSL stosuje się w celu zabezpieczenia poufnych danych wprowadzanych przez użytkowników w systemach i aplikacjach dostępnych przez Internet.

API – ściśle określony a poziomie kodu źródłowego - zestaw reguł i ich opisów, w jaki programy komputerowe komunikują się między sobą. API określa składniki oprogramowania np. aplikacje, biblioteki czy system operacyjny, dzięki czemu integracja przebiega sprawniej i generuje mniej błędów.

Skalowalność systemu – możliwość rozbudowy systemów informatycznych w przypadku zwiększonego zapotrzebowania na zasoby sprzętowe lub zasoby programu. Skalowalność służy poprawieniu wydajności systemu.

Code review – badanie kodu źródłowego oprogramowania. Jego celem jest odszukanie i naprawa błędów pominiętych w początkowej fazie rozwoju oprogramowania, a także poprawienie ogólnej jakości kodu przed oddaniem go do użytku.

Testy jednostkowe – weryfikacja poprawności działania pojedynczych elementów (jednostek) oprogramowania.

Code freeze – oznacza, że dany fragment kodu jest zakończony i zamknięty dla dalszych modyfikacji. Dopuszcza się jedynie poprawę kodu, jeśli wystąpił błąd krytyczny oprogramowania. Metodę stosuje się na ogół w końcowej fazie prac nad systemem.

VPN – wirtualna sieć prywatna, która umożliwia zdalną komunikację przez sieć prywatną lub sieć publiczną (Internet) z zasobami na serwerze. Dzięki VPN można np. zdalnie i bezpiecznie pracować nad oprogramowaniem.