

Karol Wójciszko

# Jak napisać dokumentację IT?



**Karol Wójciszko**

# **Jak napisać dokumentację IT?**

**Wydawnictwo**

**Karol Wójciszko WedługPlanu.pl**

**ISBN 978-83-961712-0-7**

**[kontakt@wedlugplanu.pl](mailto:kontakt@wedlugplanu.pl)**

*Nie kopiuj proszę tego e-booka i nie rozpowszechniaj go w nielegalny sposób.*

*Dziękuję!*

# Spis treści

|   |    |
|---|----|
| Wprowadzenie .....                                      | 5  |
| O Autorze .....   | 6  |
| Czego się dowiesz? .....                                | 8  |
| Spotkanie z klientem - JAK aby było produktywnie? ..... | 10 |
| Jak notować na spotkaniu z klientem? .....              | 15 |
| 3.1. Mapa myśli.....                                    | 15 |
| 3.2. Rejestr wymagań.....                               | 20 |
| 3.3. Kartka Papieru .....                               | 22 |
| 3.4. Dobre praktyki wywiadu z klientem .....            | 22 |
| Zastanów się co chcesz opisać.....                      | 24 |
| 4.1. Nie bój się diagramów UML.....                     | 24 |
| Jak pisać aby być rozumianym?.....                      | 26 |
| 5.1. Jaki styl pisania wybrać?.....                     | 27 |
| 5.2. Jeden dokument czy rozbić na kilka?.....           | 29 |
| 5.3. Od czego zacząć pisanie?.....                      | 31 |
| 5.4. Prefiksy i nazwy nagłówków .....                   | 32 |
| 5.5. Jeden akapit = jedno zadanie .....                 | 33 |
| 5.6. Nazwy akapitów .....                               | 35 |
| 5.7. Obgryź tekst do kości .....                        | 36 |
| 5.8. Jednoznaczność opisów .....                        | 37 |

|  |    |
|--|----|
| 5.9. Nie dubluj opisów .....   | 38 |
| 5.10. Unikaj rozważań w dokumencie .....   | 38 |
| 5.11. Bądź precyzyjny w wartościach .....  | 39 |
| 5.12. Używaj odpowiedniej formy gramatycznej .....                                       | 39 |
| Zabezpiecz się dokumentacją .....  | 41 |
| Organizacja pracy podczas pisania .....  | 44 |
| Nie masz czasu na dokumentację? .....  | 47 |
| 8.1. Pisząc RAZ, mówisz to samo WIELE RAZY .....   | 49 |
| Co po przeczytaniu? .....  | 50 |
| 9.1. Wiedza na e-mail .....  | 51 |
| 9.2. Webinar .....   | 51 |
| 9.3. Mini kurs: Jak zbierać wymagania od klienta aby poznać prawdziwe oczekiwania? ..... | 52 |
| 9.4. Trzy darmowe wideo .....  | 52 |
| 9.5. Kurs Dokumentacja.Pro .....   | 53 |
| 9.6. Pozostała wiedza .....  | 53 |
| Zakończenie .....  | 54 |

# Wprowadzenie

Nie chcę tracić Twojego czasu na czytanie wprowadzeń, więc napiszę tak krótko jak to możliwe. Ten ebook jest o tym jak napisać dokumentację pod projekt IT od zera.

Skupiam się na tym, żeby nauczyć Cię jak tworzyć dokumentację, która przynosi wartość organizacji i projektom. Mam dość czytania dokumentacji - gniotów, z których nic nie wynika.

Starłem się, aby ten ebook był tak krótki jak to możliwe - zależy mi abyś przerobił go w jeden wieczór i mógł już jutro zastosować kilka wskazówek, których się nauczysz.

Tak więc unikam rozbudowanych i niepotrzebnych opisów - przechodzę od razu do sedna. Otwórz notatnik i notuj proszę „quick tip”, które przydadzą Ci się w pracy już jutro :)

Do stworzenia tej treści popchnęło mnie również, że w internecie nie ma praktycznych wskazówek jak pisać dokumentację. Wszystko jest „rozważaniem akademickim”, które ciężko przenieść na prawdziwe problemy małych i średnich firm IT.

Dlatego skupię się na tym, aby dać Ci praktyczne wskazówki, które będziesz mógł wdrożyć już kolejnego dnia w swojej pracy.

# O Autorze

Pewnie zastanawiasz się kim jestem, żeby pisać o tworzeniu dokumentacji IT?

Nie jestem doktorantem, ani teoretykiem - jestem praktykiem, który wkłada całą parę w to żeby projekty, w których uczestniczę kończyły się na czas. Przebrnąłem przez wiele różnych „gatunków” dokumentacji i ich różnych stylów. Wyciągnąłem wnioski co się sprawdza, a na co szkoda czasu. Dlatego będę starał Ci się dać esencję wiedzy o tym jak pisać dokumentację IT tak szybko jak to możliwe.

Na co dzień prowadzę software house, gdzie uczestniczę w takich etapach projektu jak:

- zarządzanie projektem
- zarządzanie zespołem
- przygotowanie dokumentacji (analitka IT)
- wycena projektów i prowadzenie klienta

Uczestniczę praktycznie w każdym etapie życia projektu - co daje mi możliwość dać Ci praktyczne wskazówki, które pomagają w pracy.

Ebook od praktyków dla praktyków.

W wolnej chwili tworzę blog - [WedlugPlanu.pl](https://wedlugplanu.pl) i tworzę wiedzę w postaci kursów, webinarów i ebooków. Sprawdź je w moim sklepie na [edu.wedlugplanu.pl](https://edu.wedlugplanu.pl).

Więcej o mnie na <https://wedlugplanu.pl/about> lub w pierwszym odcinku mojego podcastu - <https://wedlugplanu.pl/start>





# 01

## Czego się dowiesz?



Tworząc dokumentację podzieliłem ją na cztery główne etapy - przez, które Cię przeprowadzę. Zależy mi aby ten ebook był dla Ciebie maksymalnie praktyczny - więc nie zdziw się, że opisuję rzeczy „z grubej rury”.

**Krok 1** - zebranie wymagań od klienta. Dowiesz się jak prowadzić rozmowę z klientem, aby poznać PRAWDZIWE i WSZYSTKIE wymagania projektowe. Celem tego kroku jest dowiedzenie się co trzeba opisać. W przypadku gdy piszesz projekt „dla siebie” - możesz pominąć ten krok bo wszystkie wymagania masz w głowie.

Nauczysz się przeprowadzać skuteczny wywiad z klientem.

**Krok 2** - to zaprojektowanie aplikacji, którą opiszesz w kolejnym kroku. Musisz zrobić tzw. „syntezę” czyli zebrać wiele szczegółów od klienta w jeden obrazek. Chodzi o budowanie konkretnego rozwiązania, które jest szyte na miarę pod Twojego klienta. To wiedza związana z architekturą IT.



Pomogą Ci w tym m.in. diagramy UML i makietowanie funkcjonalne - jednak nie zdecydowałem się opisać tego w tym ebooku. Forma tekstowa totalnie się nie sprawdza do wyjaśniania takich rzeczy. Gdybym to starał się opisać, ebook byłby zbyt długi. Dlatego jeśli interesuje Cię kwestia projektowania procesów, diagramów i makietowania to odsyłam Cię na Dokumentacja.Pro po więcej informacji.

**Krok 3** - znając wymagania i mając zaprojektowaną aplikację, możemy przejść do pisania dokumentacji.

Nauczysz się pisać zrozumiałym językiem dla zespołu programistów i klienta.

**Krok 4** - wskazówki dotyczące nastawienia podczas pisania i produktywności pracy.

Nauczysz się zabezpieczać dokumentacją i dowiesz się jak ją napisać, żeby nie „zajechać” się robotą.



# 02

## Spotkanie z klientem - JAK aby było produktywnie?

Byłem świadkiem wielu spotkań z klientem, które trwały 3-4h a wychodząc z nich wraz nie było wiadomo co tak naprawdę trzeba zrobić w projekcie. Klient uciekał w nieistotne anegdoty, lub prowadzący rozmowę nie zadawał właściwych pytań - nie posuwaliśmy się w przód projektu.

Zdarzało się też, że wychodząc ze spotkania miałem cztery strony A4 notatek odręcznych, które wyglądały jak hieroglify i nie mogłem z nich niczego odczytać. Notatki nie miały żadnej uporządkowanej struktury więc było praktycznie niemożliwe, żeby na drugi dzień po spotkaniu „odszyfrować” co autor miał na myśli.

Spotkania, które prowadzimy z klientem w celu poznania wymagań projektowych (tzw. wywiad z klientem), często są prowadzone chaotycznie i bez planu. Skaczemy po projekcie, nie wiemy od czego zacząć ani na czym skończyć.

W tym kroku przedstawię Ci Framework 4 pytań projektowych, dzięki którym poznasz cały zakres projektu w uporządkowanej formie. Zanim do niego przejdziemy niezbędne będzie żebym przedstawił Ci definicję kilku kluczowych pojęć, których znajomość jest niezbędna do zrozumienia dalszych akapitów.

1. **wymaganie projektowe** - jest to "jedna rzecz", którą musicie zrobić. Przykładem wymagania projektowego może być np. (usłyszane od klienta): "chciałbym integrację z systemem mailingowym". Kolejnym wymaganiem mogłoby być "chcę aby email wysyłał się do każdego nowego subskrybenta o 12:00". Każda "z tych rzeczy" jest wymaganiem projektowym.
2. **zakres** - jest to zbiór "rzeczy", czyli zbiór wymagań projektowych. Jeśli chodzi o zakres to koniecznie przeczytaj o [trójkącie projektowym](#). Dla ambitnych zadanie z gwiazdką aby pogooglać o **WBS** (work breakdown structure).
3. **analiza** - jest to proces myślowy nad zakresem projektu. Analizując zakres - będziesz starał się go zrozumieć i zaprojektować konkretne rozwiązania
4. **dokumentacja** - jest to przelanie twojej analizy projektowej (aka. przedwykonawczej) na papier. W naszym przypadku na e-papier, czyli spisanie naszych przemyśleń i sposobu realizacji do Word/ Google Spreadsheet czy Visual Paradigm lub Enterprise Architect.
5. **synteza** - bardzo powiązane z dokumentacją. Synteza jest to forma projektowania rozwiązania, czyli układania z pozoru niepowiązanych elementów w jeden wielki "big picture", na podstawie, którego powstaje dokumentacja.

Wracając do **Frameworka 4 pytań...**

Pierwszym pytaniem, które powinieneś zadać to zapytać o cel projektu. Musisz zrozumieć po co robicie dany projekt, z punktu widzenia biznesowego.

## 1. Jaki jest cel projektu?

To jedna z najbardziej pomijanych kwestii podczas tworzenia dokumentacji. Słuchamy klienta jak świnia grzmotu, spisujemy wszystko co nam mówi kiedy tak naprawdę zapominamy o tym aby zrozumieć najważniejszą rzecz: **po co realizujemy ten projekt?** Kiedy zrozumiesz motywację klienta, dlaczego chce konkretnego wdrożenia projektu nakreśli Ci się kontekst w głowie. To właśnie ten kontekst pomoże Ci szufladkować sobie w głowie dalsze opowieści o features. Poświęć na to pierwsze kilka minut rozmowy, aby zrozumieć po co w ogóle startujecie dany projekt.

Drugim pytaniem jest:

## 2. Kto będzie miał dostęp do aplikacji?

Mówiąc dostęp, mam tutaj na myśli **role projektowe**. Prawdopodobnie usłyszysz coś w rodzaju, że Pan Zbyszek powinien je mieć, powinna mieć również dostęp cała księgowość itp. Odpowiedzi ubieraj w role projektowe, od razu dopytuj kim jest Pan Zbyszek. Dzięki temu poznasz lepiej środowisko projektowe i aktorów aplikacji.

Przykładowa kategoryzacja może być taka, że wrzucisz Zbyszka i Marka w rolę „Użytkownicy”, a np. dział księgowy w rolę „Administradora” - bo z punktu widzenia systemu powinni mieć szerszy dostęp.

Nie przejmuj się jeśli na tym etapie nie wymieniacie wszystkich ról / dostępów - będzie jeszcze czas by o to dopytać.

Kolejnie bierzemy „pod lupę” każdą z ról projektowych i pytamy:

## 3. Jakie funkcjonalności mają być dostępne dla tej roli?

Na to pytanie prawdopodobnie odpowiedź klienta będzie najbardziej obszerna. Będzie wymieniał poszczególne funkcje, dostępne dla danej roli. Gdy omówisz jedną z ról, przejdź do kolejnej. Postępuj tak, aż do ostatniej roli do omówienia każdej.

Podczas omawiania funkcjonalności - twojego serca projektowego na pewno usłyszysz wiele słów kluczowych z ust klienta, które mogą być dla Ciebie niejasne. **Słowa kluczowe** to wszelkie elementy, których nie rozumiesz, a wydają się być ważne.

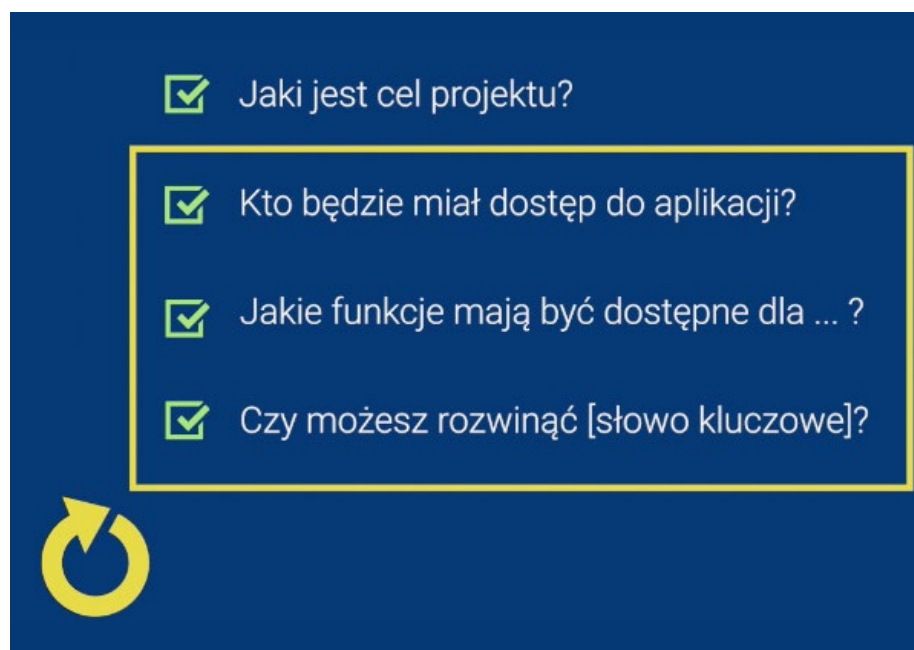
Przykładem takiego słowa kluczowego może być zdanie „my statusy zamówień trzymamy w takim Excelu na dropboxie” - powinna zapalić Ci się żółta lampka o jaki excel chodzi?

Po przejściu przez wszystkie role i funkcjonalności - dopytaj o słowa kluczowe pytaniem:

#### **4. Mógłbyś mi powiedzieć więcej o [ słowo kluczowe ]?**

Często możesz tym pytaniem otworzyć puszkę Pandory. Dzięki temu zaczniesz omawiać tzw. szarą strefę. Być może podczas omawiania jej, usłyszysz kolejne pojęcia - wtedy dopytuj o nie traktując je jako słowa kluczowe.

W pewnym momencie uznasz, że temat powoli się wyczerpuje. Wtedy warto powtórzyć pytanie drugie o role projektowe. Podczas opowiadania o projekcie być może klientowi przyjdą do głowy nowe role, a wraz z nimi nowe funkcjonalności.



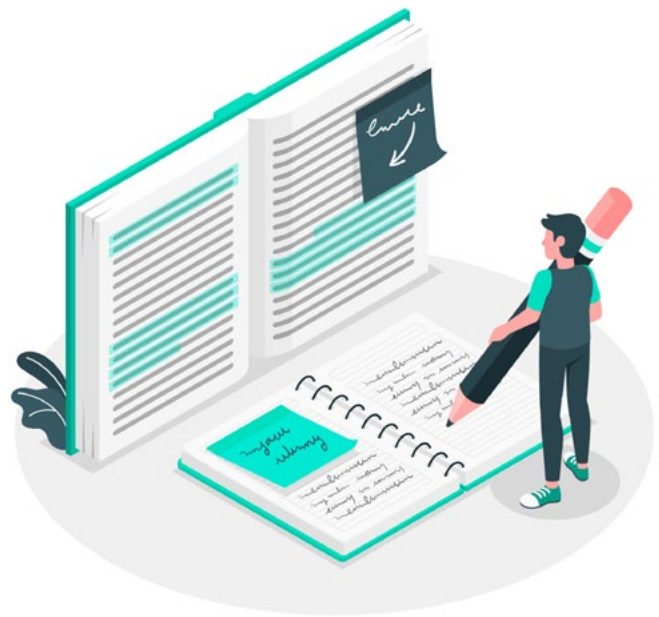
Należy takim trybem „pętli” powtarzać pytanie od 2 do 4 do momentu, aż poznamy cały projekt i nic nie będzie przychodziło klientowi do głowy.

Tak więc podsumowując - powyższe cztery pytania zadawane w odpowiedniej kolejności pomogą Ci przejść przez cały zakres.

Podejrzewam, że zapaliła Ci się w głowie lampka - jak notować informację od klienta, aby były czytelne nawet rok po spotkaniu. Właśnie o tym będzie kolejny rozdział.



# 03



## Jak notować na spotkaniu z klientem?

Istotne jest żeby forma notowania była dla Ciebie wygodna. Przedstawię Ci dwie główne metody tworzenia notatek + jedna pomocnicza. Dobierz pod siebie tą, która Ci najbardziej odpowiada.

### 3.1. Mapa myśli

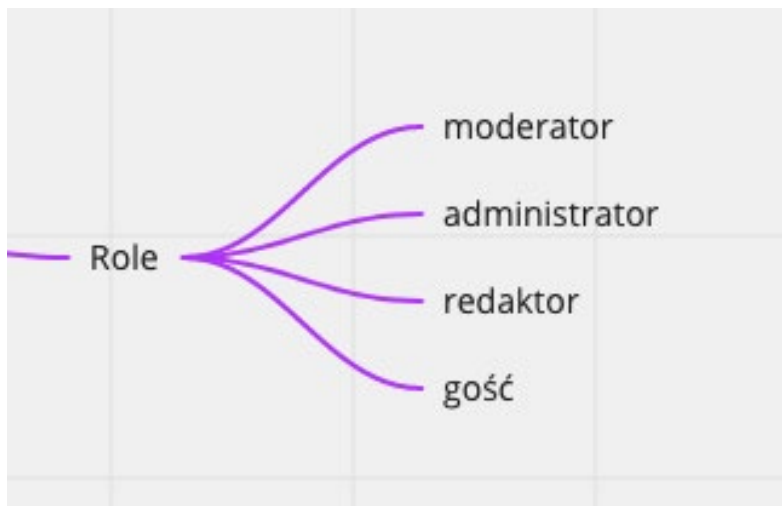
Forma notowania w mapie myśli jest moją ulubioną formą notowania. Polecam Ci zacząć notować w mapie myśli od pewnego wzoru, który może wyglądać tak:



Jak widzisz taki wzór składa się z 5 odnóg, które za chwilę Ci omówię.

- ✅ **Role** - role to odnoga, na pytanie drugie z Framework'a 4 pytań. Wpisujesz tutaj ludzi imiennie lub od razu kategoryzując ich na role projektowe. Generalizując będziesz miał w tej odnodze wszystkich aktorów projektowych.

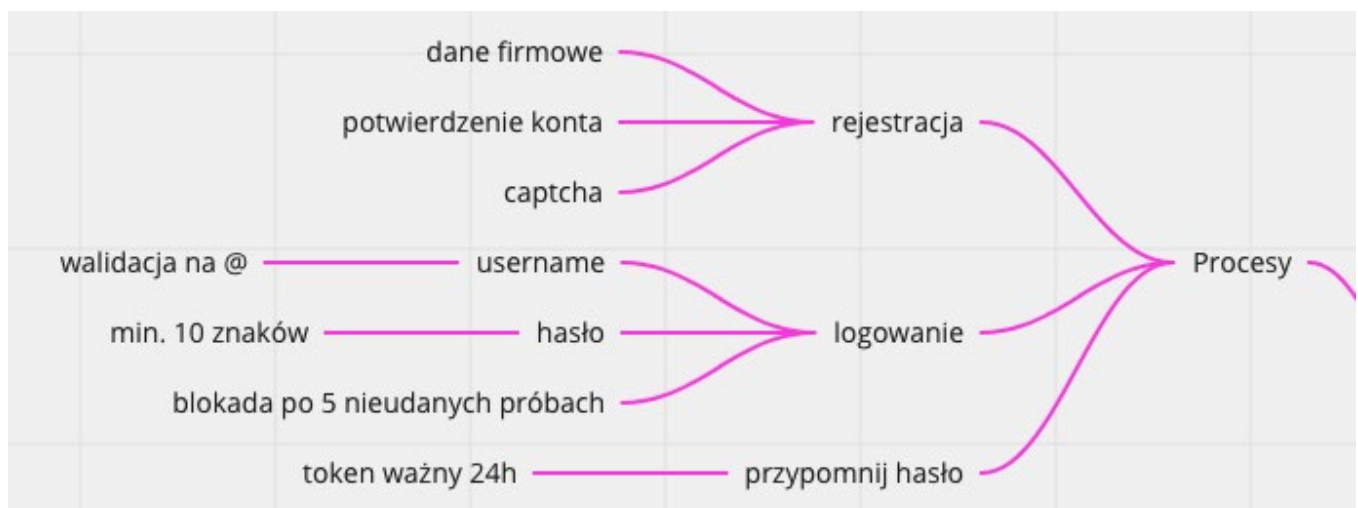
W podstawowej wersji uzyskasz mniej więcej taki wygląd odnogi Role.



Rozmawiając z klientem o rolach, będziesz później pytał o funkcjonalności do tych ról (pytanie trzecie z Frameworka). Jest na to przeznaczona odnoga moduły, ale nic nie stoi na przeszkodzie, aby te elementy notować w odnodze Role. W takim przypadku możesz od konkretnej roli zrobić odnogę z funkcjonalnościami, które wymieni klient.

- ✅ **Procesy** - w tej odnodze możesz w prosty sposób trzymać informację o procesach projektowych. Proponuję Ci trzymać się porządku, gdzie w każdej odnodze od Procesy trzymasz osobny proces. Przechodząc w głąb konkretnych procesów trzymasz kroki procesu.

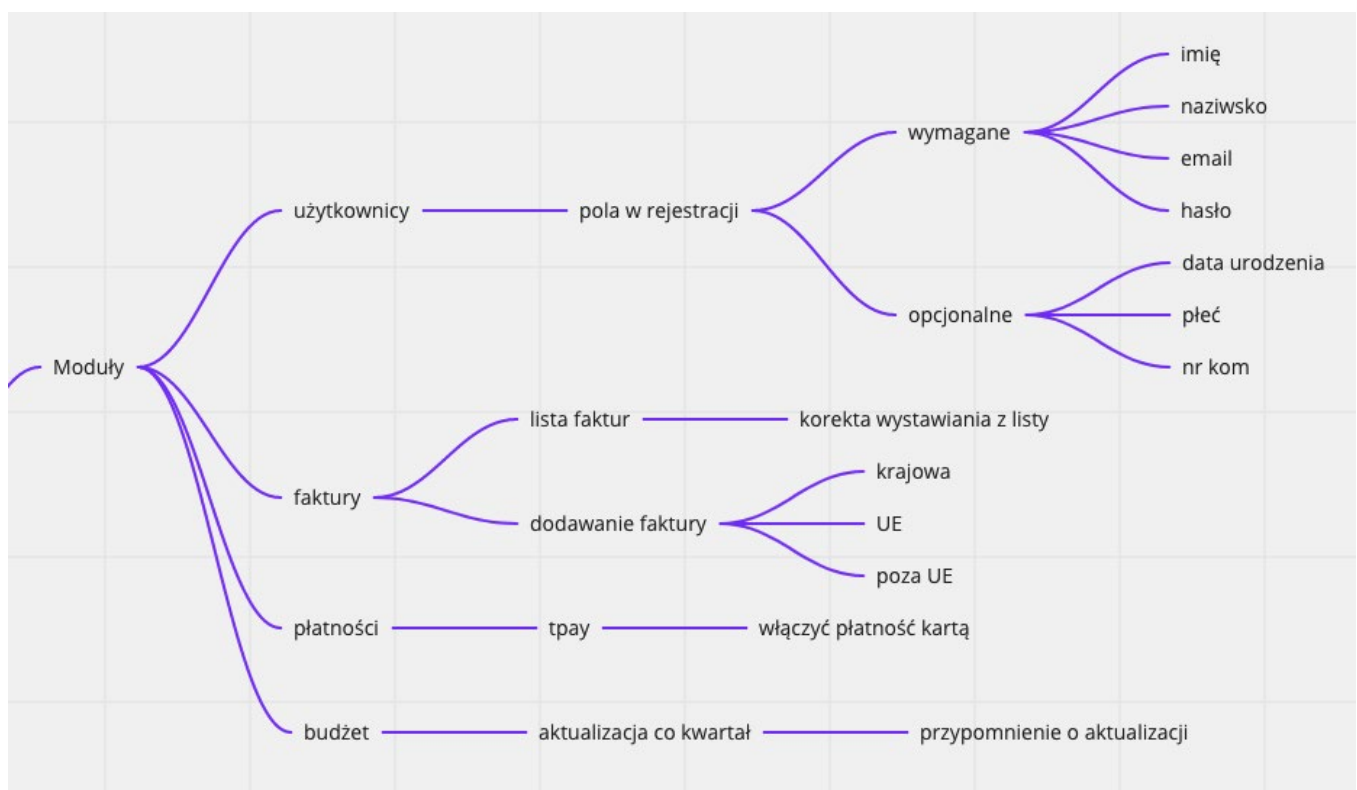
Całość może wyglądać jak na poniższym obrazku:



Jak widzisz mamy tutaj trzy procesy: rejestrację, logowanie i przypomnij hasło. Każdy z nich ma w sobie informacje, które przekazał klient. Możesz tutaj notować również informacje związane z procesem, które nie są krokiem procesu. Przykładem takiej informacji jest np „token ważny 24h” w odnodze przypomnij hasło.

✅ **Problemy** - tę odnogę należy rozumieć jako problemy i wyzwania w projekcie. Analizując rozwój istniejącego projektu usłyszysz od klienta wiele feedbacku, na temat tego co jest uciążliwe, co nie działa lub co musicie zoptymalizować. Jest to odnoga przeznaczona na takie elementy. Ja wpisuję w nią również wszelkie obawy klienta. Taką obawą może być zdanie: „poprzednia firma próbowała połączyć się z naszym CRM, ale nie byli w stanie tego skonfigurować” - może to być dla Ciebie sygnał, że pewne elementy mogą być trudniejsze niż się spodziewasz.

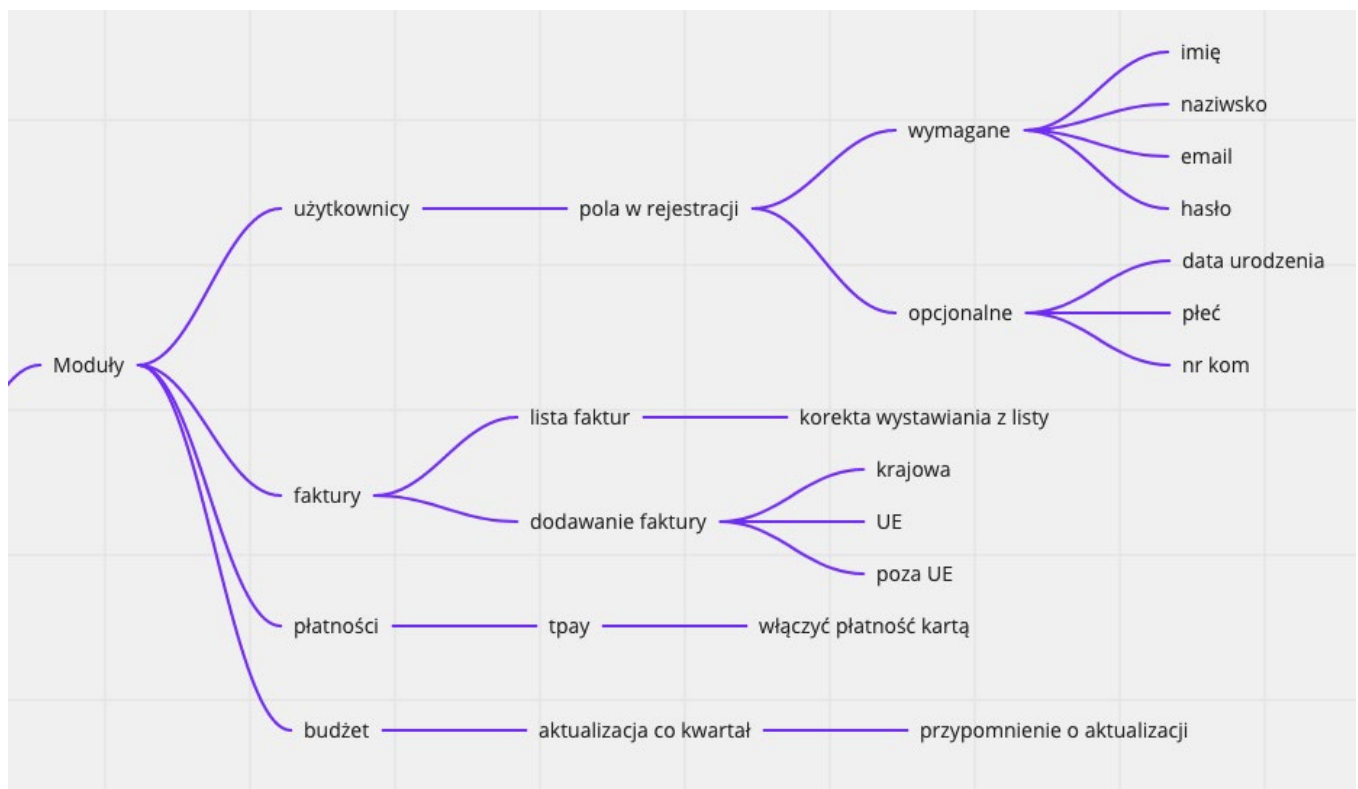
Przykład takiej odnogi może wyglądać jak na poniższym obrazku:



✅ **Moduły** - to będzie zdecydowanie najbardziej rozbudowana odnoga. Powinieneś tutaj wpisać wszystkie rzeczy, które usłyszysz od klienta z pytania 3 Frameworka - jakie funkcjonalności powinna mieć dana rola. Na tym etapie dobrze, jak będziesz w stanie agregować funkcjonalności w moduły. Tak jak wcześniej wspomniałem - jeśli wygodniej jest Ci zapisywać te elementy w odnoge Role, to nic nie stoi na przeszkodzie.

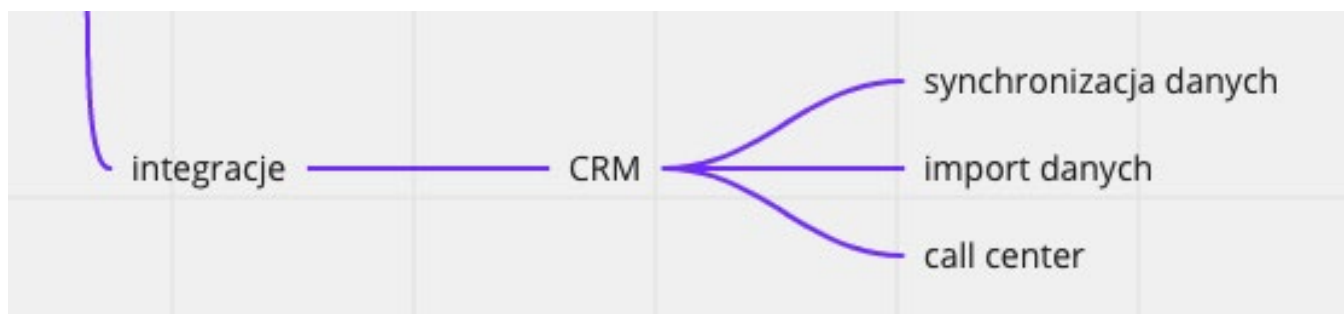
Często odnoga moduły traktowana jest jak „worek bez dna” - pasuje tu właściwie wszystko co usłyszysz. Spodziewaj się, że ta odnoga będzie „spuchnięta”.

Całość może wyglądać jak na poniższym obrazku:



- ✓ **Integracje** - jest to odnoga, w której możesz wpisywać połączenia Twojego projektu z innymi (zewnętrznymi). Możesz tutaj wpisać np integrację z CRM/ERP, mailingiem i innymi.

Przykładowe uzupełnienie tej odnogi może wyglądać jak na poniższym obrazku.



Zalety mapy myśli:

- szybkość notowania
- automatyczna hierarchia (zwiększająca zrozumienie)
- estetyczne notatki (względem notowania na kartce papieru gdzie zdarzają się skreślenia)

## 3.2. Rejestr wymagań

Rejestr wymagań to najprościej mówiąc tabelka w Excelu z odpowiednimi kolumnami. Możesz przyjść na spotkanie z klientem z przygotowanym szablonem, w którym będziesz przygotowywał notatki.

Wzór może wyglądać na przykład tak:

|   | A  | B               | C          | D   | E   | F  | G   |
|---|----|-----------------|------------|---|---|--|---|
| 1 |    | Data spotkania: | 01-01-2021 |   |   |  |   |
| 2 |    |                 |            |   |   |  |   |
| 3 | ID | Moduł           | MeSCoW     | Geneza  | Wymaganie biznesowe   | Mechanizm realizacji   | Ograniczenia  |
| 4 | #1 | Użytkownicy     | M          | Księgowość generuje raporty manualnie, klient (p. Marek) zaproponował zautomatyzowanie tego bo realizują się z księgową godzinowo i schodzi jej na to zbyt wiele czasu. | Generowanie raportu zawierającego:<br>1. ....<br>2. ....<br>3. ....<br><br>Raport byłby dostępny z poziomu menu "raporty". Przykładowy raport (załącznik) | Do realizacji użyjemy biblioteki open source "Duper Report". | Czas generowania raportu dla księgowej powinien być mniejszy niż 0.5s. Może to być wyzwaniem bo danych będzie ok. 5 mln rekordów. |
| 5 |    |                 |            |   |   |  |   |



Opiszę Ci jak rozumieć poszczególne kolumny:

- **ID** - jest to identyfikator wymagania. W celu przyśpieszenia notowania możesz je uzupełnić po spotkaniu.
- **Moduły** - jest to Moduł danego wymagania, który pomoże Ci zrozumieć szerszy kontekst. Ważne jest aby trzymać się jednej formy tekstowej dla danego modułu. Np. jeśli zdecydujemy się, że moduł nazywa się Użytkownicy, to wpisujemy go zawsze w takiej formie (zamiast np. „Użytkownik”).
- **MoSCoW** - słuchając wymagania możemy od razu ustalić jego priorytet. MoSCoW to akronim:
  - **M** - must; dane wymaganie jest konieczne do realizacji.
  - **S** - should; dane wymaganie powinno być zrealizowane.
  - **C** - could; dane wymaganie dobrze gdyby były realizowane (nice to have).
  - **W** - will not - dane wymaganie nie będzie zrealizowane w danym etapie (tylko na przykład później).

Literki „o” są wyłącznie po to aby ułatwić wymowę. Dzięki tej metodzie możesz sprawnie określić priorytet danego wymagania. Dzięki temu unikniesz koncertu życzeń ze strony klienta.

- **Geneza** - kolumna, gdzie wpisujesz cel lub powód danego wymagania. Pomaga to zrozumieć jego kontekst i potrzebę.
- **Wymaganie biznesowe** - to serce Twojego rejestru. Będzie to najobszerniejsza komórka, w której wpiszesz wszystko co o danym wymaganiu przekazał Ci klient.
- **Mechanizm realizacji** - kolumna jest opcjonalna do uzupełnienia. Często podczas rozmowy z klientem będą Ci przychodziły pomysły na realizację (lub sposób realizacji jako wymóg od klienta gdy wymagane jest użycie konkretnego protokołu). Takie elementy wpisuj w tę kolumnę.

- **Ograniczenie** - wszelkie ograniczenia biznesowe lub systemowe, które przychodzą Ci do głowy.

Jak więc widzisz rejestr wymagań jest bardziej formalnym narzędziem ze sztywną strukturą gromadzenia wymagań (względem mapy myśli). Jest na pewno „wolniejszy” do uzupełnienia niż mapa myśli lub notatki odręczne. Ma natomiast dużą przewagę - notatki zrobione w ten sposób zawierają więcej informacji kontekstowych do każdego wymagania. Pomaga to zrozumieć „co mieliśmy na myśli” nawet rok później. To dzięki takim dodatkowym elementom jak: geneza wymagania, priorytet, mechanizm realizacji i ograniczenia.

### 3.3. Kartka Papieru

---

Tak jak wcześniej wspomniałem, że przedstawię Ci dwa główne narzędzia do gromadzenia notatek i jedno pomocnicze. Czas właśnie na narzędzie pomocnicze - notatki odręczne, które wspomagają cały proces, ale nie polecałbym Ci na nim bazować.

Notatki odręczne mają dużą zaletę - nie ograniczają Cię w żaden sposób (jak na przykład ogranicza wzór rejestru wymagań lub mapa myśli). Możesz na nich kreślić w dowolną stronę i rysować symbole.

Ja głównie używam notatek odręcznych jako narzędzie wspomagające do mapy myśli lub rejestru wymagań.

Kreślę odręcznie najczęściej różnego rodzaju makiety poglądowe (stosując notatkę do którego wymagania się tyczy). Dzięki temu udaje mi się na spotkaniu z klientem uzyskać wstępne potwierdzenie „czy rozmawiamy o tym samym”.

### 3.4. Dobre praktyki wywiadu z klientem

---

Umów się z klientem na rozmowę (wywiad), w celu zebrania wymagań. Odpytuj klienta według Framework’u 4 pytań aby poznać cały zakres projektu.

Przed spotkaniem wybierz formę notatek - mapa myśli lub wzór wymagań i przyjdź na spotkanie z gotowym szablonem do uzupełnienia. Pomocniczo weź ze sobą kartkę i długopis aby wykonać poglądowe makiety i zapisywać rzeczy nie nadające się do np. rejestru wymagań.

Tak jak wspomniałem na początku tego ebooka - chcę aby był jak najkrótszy więc starałem się przebrnąć przez tę sekcję w żołnierskich słowach. Temat prowadzenia wywiadu z klientem i gromadzenia wymagań jest bardzo szeroki. O samych „najczęstszych błędach” w notowaniu można napisać odrębny ebook.

Mam nadzieję, że wszelkie informacje z tego rozdziału były dla Ciebie jasne. Jednak gdybyś potrzebował dodatkowego omówienia to zapraszam Cię do mojego mini kursu: Jak zbierać wymagania od klienta aby poznać PRAWDZIWE OCZEKIWANIA do projektu?

Omawiam w nim w formie wideo dużo dokładniej Framework 4 pytań + sposoby notowania. Jeśli chcesz wejść głębiej w tajniki - zapraszam do kursu, czekają tam na Ciebie również załączniki takie jak - wzór rejestru wymagań, wzór mapy myśli + informację jakich narzędzi używam.

Chcę być fair wobec Ciebie i nie chcę żebyś płacić drugi raz płacił za tę samą wiedzę (jednak bardziej detalicznie omówioną) - dlatego mam dla Ciebie kod rabatowy na ten mini kurs, który obniży jego cenę o wartość ebooka. Dzięki temu wyjdzie Ci on „gratis”.

Użyj kodu „**mam\_ebook\_dokumentacji**” - obniży on cenę kursu żebyś wyszedł „na zero”.

**TIP!**

*Zapytaj klienta o zgodę na nagrywanie audio waszego spotkania. Możesz przesłuchać go później ponownie - gwarantuję Ci, że wyłapiesz kilka wymagań, które umknęły Ci podczas spotkania.*

# 04



## Zastanów się co chcesz opisać

Za nami etap, z którego mieliśmy się dowiedzieć co ma być w projekcie. Kolejnym krokiem jest przetworzenie tych wymagań w naszej głowie, aby „wypluć” na kartkę przemyślany tekst.

Nie ma nic gorszego niż sięść do pisania dokumentacji, kiedy nie masz przemyślanego projektu.

### 4.1. Nie bój się diagramów UML

---

Podejrzewam, że obyłeś się o diagramy - widziałeś je lub słyszałeś. Są one narysowane za pomocą prostokątów, kółek i strzałek. Te wszystkie rzeczy nazywają się „notacją UML”. Można uogólnić, że UML to język do rysowania diagramów.

Największą obawą jaką słyszę od ludzi, którzy bronią się przed nauką diagramów jest to, że nie znają UML i nie mają czasu uczyć się całej notacji.

Uspokoję Cię - możesz śmiało zacząć rysować diagramy, nie znając całego UML. W sytuacji, gdy nie wiesz jak coś zrobić według notacji UML - wstaw swój wymyślony symbol lub element. Oznacz go legendą, żeby był zrozumiały dla czytających. Pewnie za to zdanie zostanę zlinchowany przez Strażników UML, ale tak uważam - **komunikacja poprzez diagramem ponad jego formę.**

Diagramy służą do komunikacji. Chodzi w nich o to aby przekazać za ich pomocą pewną koncepcję - np proces. Jeśli jest narysowany z błędną notacją, ale druga osoba ją rozumie tak jak tego oczekiwaliśmy to według mnie jest wszystko w porządku.

Najgorsze co możesz zrobić to nie zaczynać rysować diagramów bo najpierw chcesz poznać 100% UML.

Tak jak pisałem wcześniej - tekst zupełnie nie sprawdza się w tłumaczeniu diagramów. Dlatego odeślę Cię do mojego drugiego odcinka podcastu <https://wedlugplanu.pl/002> - gdzie opowiadam pobieżnie o diagramach oraz do kursu Dokumentacja.Pro, gdzie omawiam je kompleksowo.

# 05



## Jak pisać aby być rozumianym?

Wiele osób zapomina o tym, że dokumentacje będą czytały konkretne osoby. Błędem jest pisanie dokumentacji jakby miał ją czytać „każdy” zamiast nasz konkretny Paweł, Marzena czy Lucyna. Nam zależy na tym, żeby te konkretne osoby zrozumiały to co im chcemy przekazać za pomocą dokumentacji.

Inaczej więc będziemy pisać jeśli odbiorcą będzie **Techniczny Kierownik Projektu** po stronie klienta.

A w inny sposób będziemy pisać gdy dokumentacje “odbiera” **Prezes** firmy, który ma osobowość sprzedawcy i bardziej zwraca uwagę na aspekty wizualne niż techniczne.

**Przed napisaniem pierwszego słowa, musisz zastanowić - kto konkretnie będzie czytał Twoją dokumentację.**

Prawdziwa trudność jest jednak gdy tekst musi być pisany pod osobowość z kompetencjami “biznesowymi” i “technicznymi”.



Wtedy ilość detali, którą trzeba zawrzeć jest większa. Nie będę cię czarował - taki "case" występuje najczęściej.

Wiele osób piszących dokumentację w takim przypadku zastanawia się - **jak opisać rzeczy funkcjonalne i techniczne w dokumentacji aby było zrozumiałe?** Czy pisać to w jednym akapicie? Czy rozdzielić na dwa? A może zrobić dwa dokumenty - jeden techniczny, drugi "biznesowo-funkcjonalny".

Temat nie jest trywialny bo jest strategiczny. Musisz uwzględnić również elementy utrzymania swojej dokumentacji podejmując taką decyzję. Odpowiem na niego za kilka akapitów.

Najważniejszą rzeczą będzie na tym etapie **"wczucie" się w osobę czytającą i próba pisania pod konkretnego Pawła, Marka czy Kasię.**

## 5.1. Jaki styl pisania wybrać?

---

Zastanów się kto będzie czytał Twoją dokumentację i spróbuj określić go w trzech wymiarach:

### 1. Wiedza biznesowa

Każdy biznes jest oparty o procesy i procedury. Często są one niejawne - czyli wszyscy w firmie wiedzą, że „pewien raport robimy tak: 1) ..., 2) ..., 3) ...”, ale nie ma tego nigdzie spisane. Nikt nie zdaje sobie sprawy, że taka procedura jest nieformalna - czyli jest w głowach pracowników. Tutaj ważne zaznaczenia jest, że chodzi również o procesy, które dzieją się „offline” - czyli „na piechotę” lub „na gębę”. Bez automatyzacji.

Wiedzę o tych elementach nazywamy potocznie „biznesową”. Pomyśl na ile osoba ma pojęcie o tych rzeczach. Być może zna wszystkie procesy biznesowe na wskroś. Jeśli masz trochę szczęścia być może będziesz rozmawiał z osobą, która zaprojektowała te procesy.

Być może trafisz na osobę nowo zatrudnioną, która wszystko widzi jak przez mgłę.

Jeśli określisz sobie stopień wiedzy odbiorcy - będziesz mógł dobrać poziom szczegółów, które będziesz opisywał.

Oczywistą sprawą jest, że dla osoby z mniejszą wiedzą biznesową będziesz musiał określić szerszy kontekst biznesowy. Dla osoby, która „zjadła zęby” w danej organizacji będziesz mógł pisać bardziej lakonicznie i skrótowo bo wiele rzeczy nienapisanych będzie oczywiste.

## **2. Wiedza na temat branży i trendów**

Kategoria zbliżona do wiedzy biznesowej, ale skupiona na informacji, spoza organizacji klienta. Czyli jeśli robicie aplikacje dla branży np farmaceutycznej - to czy osoba zna ten rynek (ogólnie).

Czy rozumie trendy rynku? Taka dodatkowa wiedza „eksperska” ułatwia wiele rzeczy.

Zwróć szczególną uwagę bo często Twój odbiorca może mieć wiedzę ekspercką z innej dziedziny niż jego stanowisko. Częstym przykładem takiej sytuacji jest np wiedza z zakresu RODO lub prawnego.

Możesz wykorzystać te dodatkowe kompetencje do weryfikacji swoich założeń.

Np. jeśli ktoś zna się dobrze na prawie, to możesz w dokumencie dać szablony tekstów, które zobaczy użytkownik. W ten sposób poddasz je od razu weryfikacji.

Nie ma tutaj jednej słusznej drogi - musisz myśleć kreatywnie.

### 3. Wiedza techniczna

Ważna kwestia - szczególnie jeśli tworzycie software. Postaraj się ocenić na ile dana osoba ma wiedzę techniczną. Nikt nie oczekuje, że odbiorca będzie programistą lub architektem, ale inaczej będziesz pisał do kogoś kto ma problem z obsługą komputera, a inaczej dla kogoś kto sprawnie porusza się w Excelu.

Taka informacja pomoże Ci dobrać odpowiedni poziom detali technicznych, które odbiorca może ocenić.

Najczęściej Twój odbiorca będzie mixem tych trzech kategorii - poświęć kilka minut na zastanowienie się kim jest i co najlepiej do niego trafi aby uzyskać maksymalne zrozumienie.

Dokumentacja ma przede wszystkim zakomunikować Twoją koncepcję odbiorcy aby uzyskać feedback - czy chcecie tego samego. Wczucie się w buty odbiorcy niezwykle zwiększa zrozumienie Twoich dokumentów.

#### 5.2. Jeden dokument czy rozbić na kilka?

---

W kontekście określenia odbiorcy często dostaję pytanie - czy rzeczy techniczne i biznesowe opisywać w jednym dokumencie, czy rozbić na dwa?

Często pytają o to osoby, które mają do czynienia z odbiorcą typowo biznesowym z małą wiedzą techniczną. Wtedy korci aby przygotować dokumentację z opisem funkcjonalnym dla klienta, a drugą z opisem technicznym dla programistów.

Jednak z takiego rozdzielenia zazwyczaj jest więcej szkód niż pożytku.

Rekomenduję Ci trzymać wszystko w jednym dokumencie. Rzeczy funkcjonalne (biznesowe), są potrzebne osobom z biznesu, ale również osobom technicznym.

Bez znajomości kontekstu biznesowego, nie będą w stanie wykonać danej funkcjonalności. Sam opis techniczny często nie wystarcza. Część funkcjonalna jest często obszerniejsza względem technicznej. Strzelam, że proporcje to 70% dla części funkcjonalnej i 30% dla technicznej.

Dlatego rekomenduję Ci podejście trzymania wszystkiego w jednym dokumencie i opisywać to następująco.

Każdy akapit zaczyna się od wiedzy biznesowej/funkcjonalnej, która jest niezbędna do zrozumienia danego elementu. Na dole tego akapitu dajemy detale techniczne (opcjonalnie) - jeśli pomagają zrozumieć mechanizmy realizacji.

#### 10. Faktury

[opis części biznesowo - funkcjonalnej]

[opis części technicznej]

#### 11. Płatności

[opis części biznesowo - funkcjonalnej]

[opis części technicznej]

Dzięki temu osoba, która jest techniczna czyta dokumentację od góry do dołu i po przeczytaniu opisu funkcjonalnego od razu zbuduje szerszy obraz przez część techniczną.

Natomiast osoba stricte biznesowa przeczyta część funkcjonalną, a pominie aspekt techniczny - bo prawdopodobnie i tak go nie zrozumie.

Kluczem dobrego podziału jest odpowiednia konstrukcja spisu treści, o której opowiem Ci w kolejnym rozdziale.

### 5.3. Od czego zacząć pisanie?

---

We wcześniejszym rozdziale zrobiłem zajawkę o spisie treści. Według mnie to jeden z ważniejszych aspektów dokumentacji.

Zanim zaczniesz pisać - spróbuj ułożyć spis treści. Przeglądając go staraj się zastanowić „co opiszesz w tym akapicie”.

Staraj się uzyskać taki efekt, że Twoją dokumentację czyta się od góry, do dołu. Czyli nie trzeba skakać po dokumencie.

Prawdopodobnie na górze spisu treści powinny się znaleźć **elementy trzonowe** - czyli takie bez których nie zrozumiemy innych akapitów (bo są one ich integralną częścią).

Przykładem rzeczy „trzonowej” jest np opis „Użytkowników”, a dopiero w późniejszych akapitach opis modułów, które korzystają z użytkowników. To tylko przykład, ale mam nadzieję, że wiesz co mam na myśli.

Oprócz tego, żeby spis treści był rozumiany „od góry do dołu” to jeszcze drugim kryterium dobierania kolejności powinno być - od najważniejszych do najmniej ważnych.

Staraj się na górze opisać rzeczy kluczowe z punktu widzenia klienta - czyli np. moduł na którym mu najbardziej zależy.

Na końcu możesz opisać rzeczy mniejszej wagi - jak np integrację z zewnętrznymi systemami.

Tak więc układając spis treści skup się na tym aby Twoją dokumentację można było przeczytać od góry do dołu (bez skakania) i od rzeczy najważniejszych, do mniej ważnych.

O układaniu spisu treści można by napisać osobny ebook, nie chcę rozwlekać tego na 300 stron. Szczegółowo do wyjaśnienia jak ułożyć spis treści podszedłem w kursie Dokumentacja.Pro - gdzie poświęciłem na to dwie mięsne lekcje, omawiając przykłady.

Jeśli zdecydujesz się na dołączenie do Dokumentacja.Pro koniecznie użyj kodu „mam\_ebook\_dokumentacji” - obniży on cenę kursu o koszt ebooka! Dzięki temu ebook wyjdzie Ci na zero :)

Z darmowych treści o spisie treści polecam Ci zapoznanie się z moim wideo - [dlaczego klient nie czyta Twojej dokumentacji?](#)

## 5.4. Prefiksy i nazwy nagłówków

---

Zdarza się, że aplikacja (lub system), który projektujemy składa się z wielu elementów, komponentów. Często występuje sytuacja, że niektóre akapity nazywają się bliźniaczo np:

1. Moduł użytkowników

1. **Filtry**

2. Moduł artykułów

1. **Filtry**

Bazując później na takim dokumencie dostając zadanie: 2.1 Filtry i 1.1 Filtry – można się łatwo pomylić. Widząc taki akapit trzeba zawsze sprawdzić jakiego modułu dotyczy. Można też wprowadzać bardziej rozbudowane nazwy takie jak: Filtr modułu użytkowników, Filtr modułu artykułów itp.

Zdecydowanie lepiej w tym przypadku sprawdzi się zastosowanie prefiksów. Załóżmy, że moduł użytkowników będziemy określać skrótowo USR, a moduł artykułów ART. Polecam też dodatkowo numerowanie akapitów w obrębie danego modułu. W zależności od rozbudowania systemu można dawać też znacznik informujący o tym, czy dane zadanie jest funkcjonalne lub niefunkcjonalne.



Całość wygląda mniej więcej tak:

1. Moduł użytkowników
  1. **USR-F-001** Filtry
  2. **USR-F-002** Sortowanie
2. Moduł artykułów
  1. **ART-F-001** Filtry

Po spojrzeniu na sam prefiks **“USR-F-001”** – mamy już sporo informacji. Dopełnić ją może opis *“Filtry”* – wtedy możemy powiedzieć, że sam nagłówek jest jednoznaczny. Łatwiej też rozmawiać na temat USR-002 niż o *„sortowaniu w module użytkowników”*.

“Moduł użytkowników” i “Moduł artykułów” nie posiadają prefiksów ponieważ te elementy służą jedynie do grupowania pozostałych.

Nadawanie prefiksów ma jedną wadę - w razie zmiany struktury dokumentu (spisu treści) ich aktualizacja jest pracochłonna. Dlatego polecam Ci nadawać je na samym końcu gdy jesteś przekonany, że masz odpowiedni spis treści i jego struktura nie ulegnie zmianie.

## 5.5. Jeden akapit = jedno zadanie

---

Projektując dokumentację musisz rozpatrzyć wiele aspektów, aby ułatwić sobie życie. Jednym z takich elementów jest automatyczne tworzenie zadań dla programistów z dokumentacji.

Większość oprogramowania do zarządzania projektami - Jira, Redmine, Trello mają możliwość importu zadań z Excela. Najprościej to zrobić kopiując spis treści z dokumentacji do Excela, który importujemy - wtedy nasze zadania utworzą się automatycznie.

Oczywiście trzeba pamiętać o ustawieniu kolumn, które Twój software wymaga, ale na pewno będzie to szybsze niż Tworzenie 200 zadań manualnie.

Dlatego właśnie pisząc dokumentację powinniśmy rozpatrywać to, aby łatwo było ją przekuć na zadania do realizacji. Ma to ogromny związek z logicznie ułożonym spisem treści - bez niego ciężko będzie nam osiągnąć taki efekt.

Źle ułożona struktura pracy (zadania do realizacji) wpływa negatywnie na jakość pracy, a często na terminy. Powiem Ci kilka przykładów błędnie stworzonych zadań, abyś wiedział czego unikać.

✓ Zadania „almost done”

✓ Zadania „worek”

### **Zadania „almost done”**

Pewnie spotkałeś się z zadaniami, które są otwierane na początku projektu... i kończone na końcu. Często programiści na pytanie „czy zrobiłeś to zadanie?”, odpowiadają „Prawie, tam trzeba jeszcze tylko XYZ”.

Często nie da się ukończyć tego zadania bo np. czekamy na efekt innego zadania. Wtedy dochodzimy do sytuacji zadań widmo, które są otwarte cały czas i bardzo trudno zmierzyć jest postęp projektu.

Zadania powinny być zamykalne, jeśli więc w Twoim projekcie występują zadania „almost done” - to często wynikają z nieprzemyślanego spisu treści (struktury dokumentu).

### **Zadania „worek”**

Zadania worki to bardzo obszerne zbiory zakresu, które mogą zająć programiście np 4-5 dni, albo i więcej. Zazwyczaj jeśli PM nie chce się tworzyć bardziej detalicznej struktury pracy to tworzy zadanie typu „Moduł Użytkowników” - gdzie jest duża część zakresu. Pozornie wygląda to na oszczędność czasu organizacyjnego.

Jednak tworzenie tego typu zadań ma dużą wadę - w większości przypadków takie zadania są niedoszacowane czasowo bo programista

nie zdaje sobie sprawy z ogromu prac, który go czeka. Z mojego doświadczenia po dekompozycji takiego zadania na mniejsze - wycena różniła się nawet o 40% (bliższa faktycznemu czasowi realizacji po ukończeniu).

Pomijając aspekt niedoszacowania czasu przez programistów, który generuje opóźnienia kolejnym problemem jest, że PM dowiaduje się zbyt późno o problemach w danym zadaniu.

Jeśli zadanie worek jest wyznaczone np. na 6 dni pracy (ponad tydzień) to PM dowie się o jego opóźnieniu prawdopodobnie ostatniego dnia (pomimo, że było to prawdopodobne już po np. 2 dniach). Programista pracujący nad zadaniem rzadko kiedy da znać w połowie, że zadanie dąży do opóźnienia. Raczej będzie miał nadzieję, że jeszcze nadgoni pracę i ukończy w terminie. Niestety praca nie jest nadrabiana bo nie ma na to szans ze względu na niedoszacowany czas zadania.

Zadania typu worek w kontekście dokumentacji, często są objawem zbyt dużej generalizacji akapitów. Jeśli Twoje zadania z dokumentacji są wyceniane na np. 4-5 dni to prawdopodobnie powinieneś rozbić te „worki” na mniejsze akapity.

Będąc przy tym wątku jeszcze raz warto wspomnieć jak wielką rolę w tworzeniu dokumentacji ma właśnie odpowiedni spis treści.

## 5.6. Nazwy akapitów

---

Prawdopodobnie pracując na zadaniach w Jirze, Redmine, Trello nie zwracałeś uwagi na nazwy zadań, z którymi pracujecie. Musisz mi uwierzyć, że **nazwa zadania ma znaczenie**.

Często zdarzało mi się przeglądać zadania gdzie opis wyglądał np tak: *„Po wejściu na zakładkę nowości z użytkownika admina nie wyświetla się poprawnie menu”*.

Takie i wiele innych nazw powoduje to, że zadanie jest niezarządzalne. Ciężko o nim dyskutować - *"czy zrobiłeś to zadanie gdzie po wejściu na zakładkę nowości z użytkownika admina nie wyświetla się poprawnie menu?"* – to absurd.

Po rzucie okiem na nazwę zadania nie jestem w stanie wywnioskować o co w nim chodzi (mimo, że opis jest dokładny) – zawsze dla pewności wchodzę w zadanie i patrzę na szerszy opis, który bardzo często jest taki sam jak tytuł. Tracę czas.

Nazwa zadania powinna być krótka – nie musi w 100% opisywać problemu. Zawsze to porównuję do tytułu książki – jeśli nazwa zadania, którą chcę nadać mogłaby być tytułem książki to jest OK, jeśli nie – wymyślam nową, lepszą.

Zauważyłem też, że gdy nazwy zadań są niepoprawne (szczególnie w przypadku testowania i zakładania błędów typu bug) – praktycznie śledzenie progresu staje się niemożliwe, tworzonych jest wiele duplikatów zadania nazywanych inaczej. Wcześniej wspomniany przykład lepiej nazwać "Błąd w menu" – i szczegółowo sprecyzować warunki występowania w opisie. Należy pamiętać, że tytuł tasku nie jest po to żeby wpisywać tam opis.

## 5.7. Obgryź tekst do kości

---

Termin obgryzania do kości brzmi przerażająco, ale taki nie jest. Dzięki tej technice Twój tekst stanie się bardziej zrozumiały.

Musisz pamiętać, że dokumentacja jest tekstem klasyfikowanym jako techniczny. Używa się w niej zwrotów, których nie użyłbyś podczas "mówienia". I odwrotnie... nie powinieneś używać zwrotów potocznych i "mówionych" w swojej dokumentacji.

Mimo, że wydaje się to oczywiste, często spotykam takie określenia w dokumentacji jak:

*„Tutaj mamy widok aplikacji, a skoro użytkownik ma rangę administratora to widzi również dodatkowy przycisk.”*

Nie muszę pisać, że powyższe zdanie nie jest idealne, zawiera wiele ozdobników zdań np. - „skoro”, „również”. Po obgryzieniu do kości to zdanie brzmiało by:

*„W powyższym widoku aplikacji ADMIN widzi dodatkowy przycisk.”*

Jest krócej i zwięźlej. Napisany przez Ciebie tekst powinien być **jednoznaczny. I obgryziony do kości... czyli taki, że po wyrzuceniu jakiegokolwiek wyrazu zdanie traci sens**. Nie używaj więc proszę ozdobników mowy podczas pisania. “Keep it simple as fuck”, ale nie tracąc jednoznaczności.

## 5.8. Jednoznaczność opisów

---

Po napisaniu akapitu dokumentacji postaraj się sam zweryfikować dany fragment tekstu. Przeczytaj go i postaraj się go zwalidować - **czy napisany fragment da się zrozumieć na więcej niż jeden sposób?**

Wczuj się w buty swojego odbiorcy dokumentacji. Jeśli Ty jesteś w stanie znaleźć kilka interpretacji swojego tekstu - doprecyzuj go na tyle aby był jednoznaczny.

Jeśli tego nie zrobisz to istnieje duża szansa, że Twój odbiorca zrozumie go w inny sposób niż miałeś intencję. Zmniejszy to jego zrozumienie dokumentacji i może powodować wiele problemów podczas realizacji projektu.

Dobra dokumentacja jest jednoznaczna - czyli Twój odbiorca zrozumie ją dokładnie w taki sposób jaki miałeś w intencji.

## 5.9. Nie dubluj opisów

---

Jeśli w kilku akapitach pasuje Ci aby opisać tą samą rzecz - być może jest to znak, że powinieneś ją wydzielić do tak zwanego słownika.

Słownik to miejsce w dokumentacji, gdzie są różnego rodzaju definicje projektowe. Tworząc słownik jesteś w stanie długie i skomplikowane pojęcia ubrać w krótkie akronimy.

Np. zamiast pisać „*Użytkownik z rolą administratora i moderatora*”, możesz to zdefiniować jako np. „Superuser”.

Wtedy zamiast za każdym razem pisać definicję SuperUsera (jakie ma rangi) - możesz użyć pojęcia tłumaczącego (Superuser).

Tę strategię linkowania możesz użyć szerzej. Zdarza się często, że robimy np różnego rodzaju tabele, z której wynika np. jaki użytkownik ma dostęp do jakiego modułu (coś na wzór ACL (access control list)).

W wielu funkcjonalnościach zamiast pisać kto ma dostęp do danego modułu - możesz **linkować do tabeli z ACL**. Dzięki temu w przypadku zmiany uprawnień użytkowników do modułów zmieniasz to w jednym miejscu (tabeli ACL), a reszta dokumentu jest spójna bo linkowałeś w nim do wspomnianej tabeli.

Główną puentą tego akapitu jest to, żeby nie opisywać jednej rzeczy w wielu miejscach, tylko opisać ją w jednym i linkować do niej w ramach potrzeby.

## 5.10. Unikaj rozważań w dokumencie

---

Czytałem wielokrotnie dokumentację, w której autor rozpisywał się na temat wyborów, które mamy.

Mniej lub więcej brzmi to tak „*Jeśli zdecydujemy się na opcję A, to XYZ. Jeśli na opcję B to ZXY*”.

Głównym błędem takich rozważań jest to, że dokumentacja powinna zawierać przemyślany i opracowany zbiór przemyśleń. Czyli w przypadku gdy zastanawiasz się nad opcjami - to decyzję powinieneś podjąć przed napisaniem, a w tekście opisać drogę, którą wybrałeś.

Postępując inaczej, czyli robiąc rozważania w tekście jest to bardzo niejasne - bo rzadko kiedy klient będzie wiedział, że w tym miejscu ma podjąć jakąś decyzję i dać Ci znać.

Dużo sprawniej będzie jeśli przedstawisz mu opcję A i B i finalnie wybraną opiszesz w dokumencie.

## 5.11. Bądź precyzyjny w wartościach

---

Kolejną rzeczą jest żeby unikać liczebników nieokreślonych, takich jak "trochę", "mało".

Zdanie, które brzmi *"Aplikacja będzie działała trochę wolno dla dużego ruchu"* nic tak naprawdę nie znaczy.

Dużo lepiej jest określić dla jakiego ruchu działa optymalnie, np *"Aplikacja będzie działała płynnie dla 100 użytkowników równolegle"*.

Takie naleciałości biorą się z języka mówionego - nawet gdy staramy się mówić precyzyjnie to zdarza nam się wtrącić "około", "mało", "dużo", "trochę" i jeszcze inne.

## 5.12. Używaj odpowiedniej formy gramatycznej

---

Kolejną rzeczą jest forma, której używasz. Polecam pisać bezosobowo czyli np:

*"Zostanie uruchomiony cron"* lub używać osoby trzeciej czyli *"Uruchomimy cron o godzinie 19"*.

Możesz sobie wyobrazić, że piszesz w imieniu organizacji, wtedy bardziej naturalnie przyjdzie Ci osoba trzecia.  
"Zrobimy, wdrożymy" itp itd.

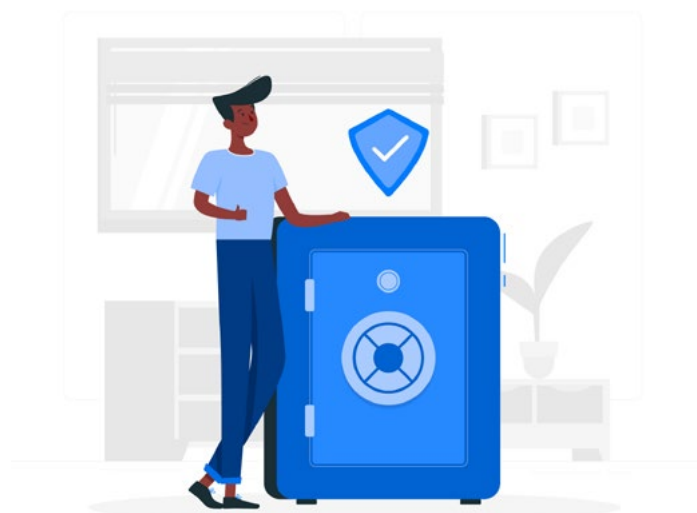
Skłonności do używania osoby pierwszej mają zazwyczaj ludzie, którzy sami będą realizowali to co opisują, wtedy zdarza się im napisać *"zrobię to tak i tak". "Tutaj zastosuję takie rozwiązanie"*.

To brzmi mniej profesjonalnie, zdecydowanie lepiej używać osoby trzeciej, nawet gdy jesteś freelancerem. Musisz zrozumieć, że pisanie dokumentacji to coś innego niż pisanie email.



# 06

## Zabezpiecz się dokumentacją



Często czytając dokumentację mam wrażenie, że projekt, który został w nich opisany jest idealny i zbawi świat. Każda funkcja jest opisana w samych superlatywach. Dzieje się tak dlatego, że autor w dalszym ciągu próbuje „sprzedać” produkt.

W większości przypadków jeśli powstaje dokumentacji to projekt jest już “sprzedany” – czyli został wstępnie oszacowany kosztowo i ta cena została zaakceptowana przez odbiorcę. Cel tworzenia dokumentu analitycznego jest zupełnie inny – po jego stworzeniu ma być wyklarowana finalna wizja projektu – jego efekt końcowy.

Wszystkie funkcje powinny być uzasadnione, a ich realizacja powinna zostać zaplanowana używając jak najmniejszego nakładu pracy. Im szybciej coś wykonamy, tym taniej nas to wyniesie, dokumentacja nie powinna nadmuchiwać “balonika cenowego” – od tego jest dział sprzedaży, którego działanie powinno poprzedzić etap analizy przedwykonawczej.

Moim zdaniem nie ma lepszego marketingu niż rekomendacja zadowolonego klienta. Warunkiem koniecznym w uzyskaniu satysfakcji przez klienta jest to, że finalny produkt będzie wyglądał dokładnie tak jak oczekiwał/wyobrażał go sobie.

Moment, w którym klient uświadamia sobie jak będzie wyglądał projekt po realizacji następuje po przeczytaniu dokumentacji i ostatecznemu zatwierdzeniu jej – że to jest dokładnie to, czego chciał. Właśnie dlatego dokumentacja powinien opisywać projekt w realistycznym świetle. Używając sformułowania “realistyczne światło” mam na myśli to, że dokument powinien zawierać przede wszystkim:

- **ograniczenia funkcjonalne** – czyli to czego nie będziemy mogli robić w systemie. Często to nie jest dla klienta jednoznaczne, zwłaszcza jeśli buńczucznie nazywamy małe funkcjonalności wielkimi słowami. Z mojego doświadczenia często spotykałem się z rozjazdem w “Raportach”, które brzmią ciekawie, a często zrealizowane raporty nie dostarczają tego co klient oczekuje. Chodzi o sytuację gdy jeden wykres nazywany szumnie „modułem raportów”.
- **planowane niedociągnięcia funkcjonalne** – często niektóre z funkcjonalności zawierają planowane niedociągnięcia. Np pominięcie weryfikacji wprowadzania danych przez administratora ponieważ to administratorowi będzie zależało na wprowadzeniu poprawnych danych. Dzięki temu można zredukować roboczno godziny pracy. Kluczowe jest żeby nie tworzyć nieprzemyślanego systemu poprzez “niedociągnięcia funkcjonalne” tylko je zaplanować i zastosować tam gdzie można – oczywiście informując o tym klienta.
- **problemy, których nie rozwiązuje** – tworząc dokument analityczny często próbuję “wcielić się w klienta” – to pozwala mi na wymyślenie alternatywnej interpretacji funkcji projektu. Te alternatywne interpretacje, trzeba zanegować – musimy mieć pewność, że rozmawiamy o tym samym. Dzięki temu unikniemy komentarzy klienta w stylu: *“myślałem, że to również wykona za mnie XYZ, nazwa jest podobna, pewnie byście to zrobili w kilka minut!”*.

- **sytuacje, w których projekt jest awaryjny** – każdy system jest awaryjny, tylko niektóre z nich mają nieokreślone zakresy awaryjności – przez co ich autorzy mylnie uważają je za bezawaryjne. Moim zdaniem należy skupić się na określeniu sytuacji, w których poprawnie zaimplementowane funkcje projektu przestaną działać. Wbrew pozorom może okazać się, że sytuacje, w których system będzie awaryjny są kluczowe dla klienta. Jeśli tego nie odkryjesz na etapie analizy – nie uzyskasz satysfakcji klienta, bo dla niego produkt okaże się nieprzydatny.
- **fragmenty, których rozbudowa będzie trudna** – projekty IT mają to do siebie, że są rozbudowywane i powinny być skalowalne. Określ klientowi fragmenty funkcji, których rozbudowa będzie trudna/droga lub wręcz niemożliwa. Wbrew pozorom dla osoby nie technicznej (jaką jest często klient) nie jest to łatwe do przewidzenia. Taki akapit w analizie może wiele wyjaśnić. Rzuć okiem na mój wpis o tym gdy klient mówi “ale przecież miało być napisane elastycznie!!!!” – [analizie przedwykonawczej](#).

Jeśli opiszesz te “negatywne strony” projektu, możesz uniknąć wielu zmian zakresu podczas realizacji. Prawdopodobnie unikniesz też rozczarowania klienta i uda ci się uzyskać jego satysfakcję. Klient dostanie “to czego chciał”.

# 07



## Organizacja pracy podczas pisania

Musisz wiedzieć, że rozróżniamy dwa główne typy dokumentacji:

1. Dokumentacja całościowa
2. Dokumentacja przyrostowa

**Dokumentacja całościowa** opisuje projekt od A do Z. Dobrym przykładem są różnego rodzaju podręczniki użytkownika/manuale. One opisują każdy aspekt projektu w jednym dokumencie. Jeśli część projektu ulega zmianom wtedy taką dokumentację należy zaktualizować aby zawierała najnowsze opisy.

**Dokumentacja przyrostowa** jest tworzona pod zmiany w projekcie. Jeśli macie projekt w wersji 0.1 i wykonujecie zmiany, które nazwiecie 0.2 to dokumentacja przyrostowa będzie zawierała drogę pomiędzy wersją 0.1, a 0.2.

Głównie w dokumentacji przyrostowej opisujemy:

- ✓ **CO ZMIENISZ** - który element
- ✓ **NA CO ZMIENISZ** - jak będzie wyglądał po ukończeniu prac
- ✓ **JAK ZMIENISZ** - jak przebiegnie proces zmiany

Dokumentacja przyrostowa będzie lepsza jeśli chcesz śledzić zmiany w projekcie, natomiast całościowa gdy chcesz wytłumaczyć komuś funkcje projektowe. Każde wdrożenie i większa zmiana jest osobną dokumentacją przyrostową.

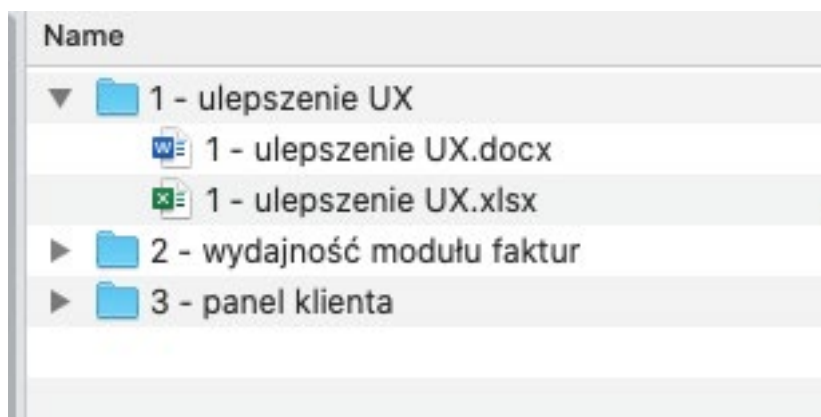
Dzięki temu możesz precyzyjnie śledzić jak projekt się rozwijał, ale trudno z nich wywnioskować jak aktualnie wygląda - musiałbyś przeczytać wszystkie dokumentacje przyrostowe w kolejności takiej, jak rozwijaliście projekt. To trudne do zrozumienia.

Pracując nad dokumentacjami jeśli potrzebujesz i śledzić zmiany w projekcie i jednocześnie mieć dokument, który będzie czymś na wzór „manuala” proponuję Ci podejść do tego tak:

1. Napisz dokumentację całościową projektu
2. W razie zmian twórz dokumentację przyrostową
3. Po każdej dokumentacji przyrostowej aktualizuj dokumentację całościową.

Wydaje się pracochłonne, ale takie nie jest. Dzięki temu będzie i wilk syty i owca cała.

Jeśli chodzi o organizację pracy to najtrudniej jest utrzymać porządek w dokumentacjach przyrostowych, dlatego proponuję Ci pracować na takiej strukturze folderów:



W tym przypadku mamy osobny folder na każdą dokumentację przyrostową, którego nazwa jest według schematu: [numer zmiany] - [skrótowa nazwa].

Dzięki temu będziesz w stanie dość sprawnie znaleźć to czego potrzebujesz. Ważną kwestią jest, żeby pliki nazywać również w tej konwencji - dzięki temu będziesz mógł znaleźć coś po nazwie pliku. Dużo wygodniej niż wyświetlić 50 plików dokumentacja.docx.

Jeśli umowa z klientem Ci na to pozwala dobrze te dokumentacje trzymać w chmurze - np Dropbox / Google Drive. Dzięki temu wszyscy mają dostęp do najnowszych wersji dokumentu (bez przesyłaniu sobie na email każdej wersji).

# 08

## Nie masz czasu na dokumentację?



Chcę powiedzieć, że nie jestem ortodoksem jeśli chodzi o tworzenie dokumentacji. Uważam, że są projekty, których nie ma sensu dokumentować. Twierdzę jednak, że pisanie dokumentacji oszczędza dużo czasu.

Wiem, że napisanie dokumentu potrafi być pracochłonne, ale koniec końców jesteś na plusie dzięki korzyściom z jej pisanie.

Mam wrażenie, że często firmy **nie tworzą dokumentacji bo są zbyt zajęte bieżączką**. Mają tyle pootwieranych aktualnych tematów, że nikt nie wyobraża sobie znalezienia czasu na stworzenie dokumentacji.

Kolejnym najczęściej występującym powodem nie tworzenia dokumentacji jest **brak formalnej roli analityka** w firmie.



Wiem, że stworzenie dobrej dokumentacji nie jest proste. Ale uwierz mi, że **nie musisz być formalnie zatrudniony jako analityk by stworzyć dokument, który przyniesie korzyści dla pracy Twojego zespołu.**

Widziałem wiele świetnych dokumentów napisanych przez kierowników projektów, senior programistów lub testerów. Wszystko jest kwestią praktyki.

Największe korzyści z pisania dokumentacji to sprawy analityczne. Przy wylewaniu z siebie myśli (na klawiaturę) musisz się zastanowić nad tym co piszesz. Lubię używać stwierdzenia, że "sam układasz sobie projekt w głowie".

Podczas wywiadu z klientem prawdopodobnie przeprowadziłeś analizę. [Pisanie dokumentacji jest syntezą.](#)

Tworząc rozwiązanie sam je walidujesz podczas pisania. Możesz zaprojektować mockupy, diagramy – sam zobaczysz co jest potrzebne żeby odpowiednio przekazać swoje myśli.

To co chcę zaznaczyć w kontekście oszczędności czasu to fakt, że przed realizacją pracy **DOKŁADNIEJ JĄ PRZEMYŚLISZ.**

A **uwierz mi – nic nie idzie sprawniej niż przemyślany projekt.** Z mojego doświadczenia wynika, że **najwięcej zmian w projekcie są spowodowane niewłaściwą analizą.**

Albo nie odkryliśmy dokładnie potrzeb klienta, albo źle zaprojektowaliśmy naszą aplikację.

W każdym razie zmiany to pisanie tego samego w inny sposób, więc w pewnym sensie tracisz czas.

Ja podczas pisania dokumentacji milion razy (w przybliżeniu 😊 ) zauważyłem, że proces przekazany od klienta się "nie spina".

Wyłapałem ogromną ilość luk procesowych, które wpadały w ślepy zaułek.

To wszystko dzięki temu, że pisząc dokumentację musiałem "wyobrazić" sobie efekt finalny projektu.



## 8.1. Pisząc RAZ, mówisz to samo WIELE RAZY

---

Wyobraź sobie sytuację, w której nie tworzysz dokumentacji. Tak czy siak w jakiś sposób musisz przekazać programistom co trzeba zrobić. Rozmawiasz z jednym, rozmawiasz z drugim... może nawet zbierzecie się “w kupę” i powiesz to raz. Możemy nawet założyć, że uczestnicy zrobią notatki lub to co opowiadasz jest poparte lakonicznymi mailami.

Tak czy siak będziesz musiał powtórzyć to więcej niż raz. Ludzie nie są idealni i o części tego co mówiłeś zapomną. Wszystko to przy optymistycznym założeniu, że ZROZUMIELIŚCIE się i wszystkie strony tak samo wyobrażają sobie efekt końcowy. To nie jest takie oczywiste.

Dobra dokumentacja jest JEDNOZNACZNA. Czyli nie ważne kto ją czyta – każdy powinien zrozumieć ją tak samo. Nie jest łatwe tworzyć takie dokumenty, ale wszystko jest kwestią praktyki.

Spisując to co ma być zrobione tak naprawdę “mówisz to wiele razy” nie poświęcając na to czasu. Czyli go oszczędzasz.

W razie pytań możesz odesłać do dokumentacji. Gdyby były w niej luki – możesz uzupełnić dokument. To lepsze zamiast zwoływać zebranie lub pisać aktualizujący e-mail dotyczący wymagań.

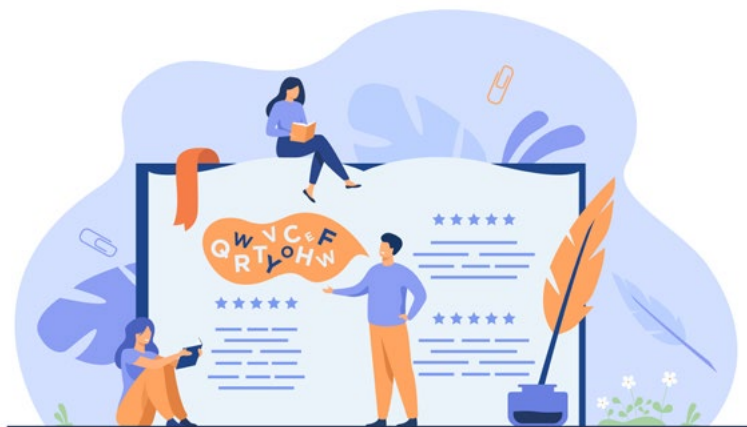
Przechodząc przez to (proces tworzenia dokumentacji) przy kilku projektach staje się to procesem.

Wiesz co jest istotne, od czego zacząć, wiesz kto ma potwierdzić wymagania – twoja organizacja zaczyna mieć procesy analityczne. Dużo łatwiej jest później wyciągać wnioski który klocek procesu należy poprawić.

W sytuacji gdy działacie “losowo”, w każdym przypadku inaczej – bardzo trudno wyciągnąć obiektywne wnioski.

# 09

## Co po przeczytaniu?



Skup się na praktyce. Musisz zdobyć umiejętność komunikowania się w formę pisemną, kolejnie małe fragmenty grupować logicznie w duży obrazek projektu - to w skrócie jest dokumentacja.

Pisz dokumenty - te pierwsze będą słabe, ale nie zrażaj się tym. Każdy kiedyś zaczynał. Nie bój się pisać „do szuflady”. Nie bój się, że pierwsze wersje będą „do wyrzucenia”.

Jeśli pracujesz w pracy nad jakimś projektem - spróbuj go opisać wykorzystując wskazówki z tego ebooka. Jeśli kompletnie nic nie przychodzi Ci do głowy możesz ćwiczyć pisanie na istniejącym projekcie, który znasz. Opisz na przykład swój ulubiony edytor tekstowy - Evernote, Notion, Notes.

Ogranicza Cię wyobraźnia, a najważniejsza jest praktyka. Moje pierwsze dokumentacje były gniotami, ale wytrwałość pozwoliła mi z czasem pisać bardzo dobre dokumentacje.

Poniżej chciałbym Ci przedstawić ciekawe miejsca, gdzie warto być, aby rozwinąć się z zakresu analizy i dokumentacji.

## 9.1. Wiedza na e-mail

---

Na moim blogu możesz zapisać się na newsletter poświęcony pisaniu dokumentacji. Raz w tygodniu przesyłam ciekawe tipy i przemyślenia w tym temacie. Warto być na mojej liście!

<https://wedlugplanu.pl/zacznij-tu/>

## 9.2. Webinar

---

Jakiś czas temu prowadziłem webinar **Jak napisać dokumentację IT (bez bólu)?** Nagranie trwa ponad godzinę i jest bardzo mięsne. Omawiam w nim kwestię:

- ✓ O co pytać klienta aby poznać prawdziwe oczekiwania projektowe?
- ✓ Jak notować aby było czytelne nawet tydzień po spotkaniu?
- ✓ Programiści zrobili inaczej niż miało być – jak zapobiec?
- ✓ Jak zmniejszyć ilość błędów w projekcie za pomocą dokumentacji?
- ✓ Jak zmniejszyć dług technologiczny za pomocą dokumentacji?
- ✓ Jak skrócić czas pisania dokumentacji?
- ✓ Dlaczego klient nie czyta Twoich dokumentacji?
- ✓ Jak pisać dokumentację pod projekty „od zera” a jak do ciągnących się w czasie?
- ✓ Opis techniczny czy biznesowy?
- ✓ Jak przekonać szefa do pisania dokumentacji?
- ✓ Jak opisywać wady projektu?

Jak widzisz część rzeczy dowiedziałeś się z tego ebooka - dlatego mam dla Ciebie kod zniżkowy na webinar. Dzięki temu nie będziesz płacić za tą samą wiedzę dwa razy. Wtedy ebook wyjdzie Ci bezpłatnie. Kod to „**mam\_ebook\_dokumentacji**”.

[Webinar: Jak napisać Dokumentację IT \(bez bólu\)?](#)

### **9.3. Mini kurs: Jak zbierać wymagania od klienta aby poznać prawdziwe oczekiwania?**

---

To wycinek kursu Dokumentacja.Pro, w którym omawiam bardzo dokładnie zbieranie wymagań. Skupiam się na metodzie wywiadu z klientem - framework'u 4 pytań.

Omawiam też szerzej sposoby notowania na spotkaniu + znajdziesz w tych lekcjach wzory mapy myśli, rejestru wymagań gotowe do użycia wraz z komentarzem.

Wiedza ta została poruszona w tym ebooku, dlatego standardowo kod „**mam\_ebook\_dokumentacji**” obniży cenę mini kursu o wartość ebooka.

[Mini kurs: Jak zbierać wymagania od klienta aby poznać prawdziwe oczekiwania?](#)

### **9.4. Trzy darmowe wideo**

---

Jakiś czas temu na [moim YouTube](#) opublikowałem trzy wideo z tematu dokumentacji, możesz rzucić na nie okiem:

- [Co napisać w dokumentacji?](#)
- [Jakie diagramy w dokumentacji?](#)
- [Jak ułożyć spis treści w dokumentacji?](#)

Polecam Ci również przesłuchanie mojego podcastu, w którym poruszam m.in. kwestię [jak pisać dokumentację w Scrum?](#)

## 9.5. Kurs Dokumentacja.Pro

---

To mój najnowszy kurs, w którym uczę jak tworzyć zrozumiałe dokumentacje IT. Przeprowadzę Cię w nim „za rękę” do napisania pierwszej dokumentacji.

Kurs jest rozbudowaną formą nauki - zawiera 24 lekcje wideo, które łącznie trwają prawie 5h. Do kursu otrzymasz wiele załączników jak np wzór dokumentacji + materiały z mini kursu o zbieraniu wymagań.

[Kurs dokumentacja.pro](#)

## 9.6. Pozostała wiedza

---

Zachęcam Cię do rzucenia okiem na pozostałe produkty, które są najczęściej dodawane razem do koszyka:

- [Bezpieczna Umowa Wykonania Projektu IT](#)
- [Wideo: Wycena Projektu IT od A - Z](#)
- [Szablon Excel do Wyceny Projektów IT](#)
- [Kurs Dokładna Wycena Projektów IT](#)

# Zakończenie

Idealna sytuacja po zakończeniu czytania tego ebooka to z jednej strony wiesz od czego zacząć pisanie i jak nie popełniać podstawowych błędów. Z drugiej strony zdajesz sobie sprawę, że analiza i tworzenie dokumentacji to ciągła nauka, która wymaga wytrwałości.

Jestem przekonany, że sobie poradzisz, a wiedza z tego ebooka pomogła Ci chociaż w 1% :)

Bardzo mi zależy na tym aby Cię poznać - uwielbiam mieć bliski kontakt z czytelnikami, dlatego zachęcam Cię do kontaktu [kontakt@wedlugplanu.pl](mailto:kontakt@wedlugplanu.pl)

Podziel się ze mną swoją opinią na temat ebooka, lub pierwszym stworzonym dokumentem. Możesz również oznaczyć mnie w social mediach - nie zdajesz sobie sprawy jak bardzo motywuje mnie to do dalszego tworzenia dla Ciebie treści.

Pamiętaj, że trzymam za Ciebie kciuki! Powodzenia!

A handwritten signature in blue ink, reading "Karol Bzysko". The signature is fluid and cursive, with the first name "Karol" and the last name "Bzysko" clearly distinguishable.

# Bądź na bieżąco

---



*Trzymam za Ciebie kciuki!*