Event Storming

Event Storming wspiera

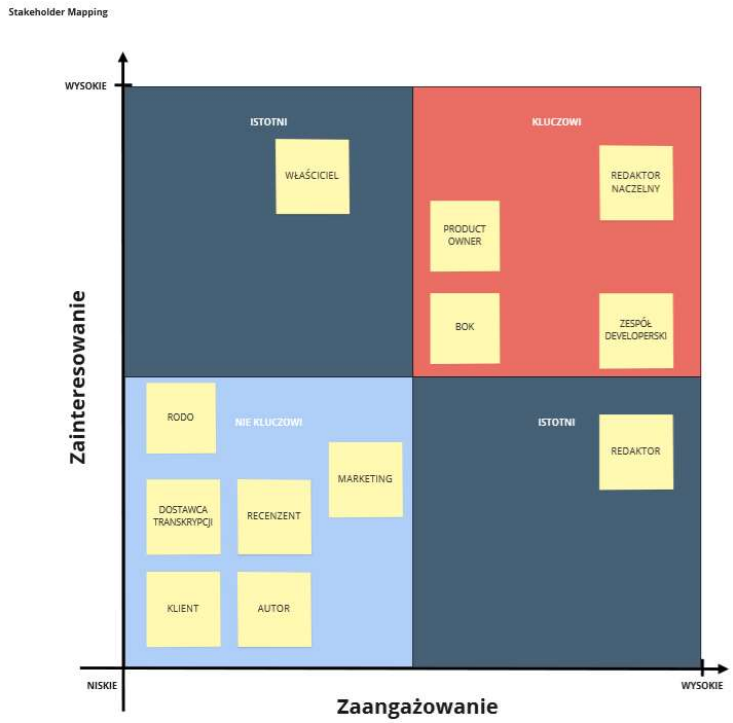
1. **Eksploracje przestrzeni problemu (domeny biznesu, do której tworzymy oprogramowanie) poprzez dokumentacje procesów biznesowych w zidentyfikowanych subdomenach**
2. **Identyfikacje reguł biznesowych, ulepszenie procesów biznesowych, identyfikacje bounded contextów tworzonego systemu**
3. **Zaprojektowanie (w celu docelowego oprogramowania):**
    a. **modeli pojedynczych komponentów w bounded Context-cie**
    b. **architektury Bounded Contextu i jego modularyzacje**
    c. **sposobów komunikacji między bounded Contextami i wewnątrz bounded Contextu**

Te 3 elementy stanowią podstawę do świadomej budowy nietrywialnego systemu informatycznego i są idealną bazą do zdefiniowania Epic-ów, Featuer-ów, Product Backlog Item-ów / User Storiesów i Tasków w SCRUM-ie. Gdyż stanowią jednolity obraz ich współistnienia tj.: widać biznes systemu i przyjęte modele architektoniczne tworzonego oprogramowania.

Alternatywą do realizacji tych 3-ech punktów jest Event Modeling – www.eventmodeling.org

Warsztat Event Storming Big Picture

Kogo zaprosić na warsztat Event Storming Big Picture?



Rysunek 1 Źródło: DNA

Przygotowania do warsztatu Event Storming Big Picture

Przyda się

- Przestrzeń
- Ściany - papier plakatowy, taśma malarska
- Karteczki i Flamastry
- Flipchart
- Minutnik
- Narzędzie do dokumentacji miro board – można na nim przeprowadzić Event Storming Zdalny, czasem jedyne wyjście, choć wskazany jest Event Storming Big Picture wspólnie, nie zdalnie.

Fazy warsztatu Event Storming Big Picture

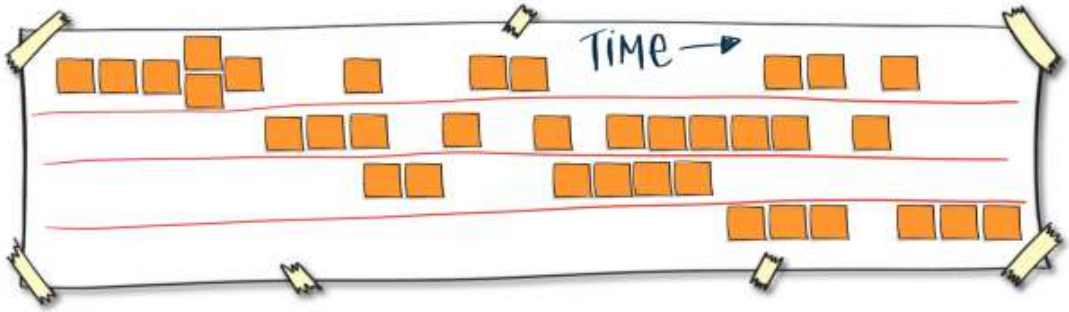
Faza 0 - Rozpoczęcie (przedstaw cel spotkania)

Faza 1 - Chaotyczna eksploracja

- Pomarańczowa karteczka - **event biznesowy**
- Istotne zdarzenie biznesowe w procesie biznesowym (np. Wystawienie faktury)
- Czas przeszły (tryb temporalny, coś się stało więc ktoś to zrobił, coś ma jakiś stan i ma poprzednika / następnika i swoje konsekwencje w czasie)
- Każdy osobno klei eventy - opcjonalne, ale się sprawdza
- Kończymy chaotyczną eksplorację - kiedy pierwsza osoba skończy wypluwać eventy (karteczki)

Faza 2 - Wprowadzenie osi czasu - uporządkowanie eventów zgodnie z linią czasu

- Wprowadzenie struktury
- Usunięcie duplikatów
- **Hotspot** - czerwona karteczka - jest jakiś problem ogarniemy go później
- SwimLine - bardzo dużo zdarzeń i może pojawić się sporo równoległych procesów



Faza 3 - Opowieść od końca

- Angażuje Facylitatora
- Debugowanie procesu biznesowego
- Znajdowanie brakujących eventów - biznes ma się tu skupić (wejść w stan skupienia pomaga im przejście od końca)
- Co się dzieje
    - Nowe zdarzenia
    - Poprawki
    - Rozgałęzienia
- Pytania Facylitatora by pomoc debugować proces biznesowy

    - Wcześniej
        - Co jeszcze musiało się stać, żeby X
    - Pomiędzy
        - Czy między X a Y dzieje się coś jeszcze?
    - Alternatywa
        - Co, jeśli X by się nie powiodło
- Można już to znaleźć i oznaczyć **reguły biznesowe** (np. nie można zarezerwować mentora kiedy jest niedostępny, kleimy żółtą kartkę Jeśli jest dostępny" przed Eventem „Zarezerwowano Mentora")
- Strukturyzowanie układu eventów przez
    - Klejenie w dół (eventy mogą zadziać się wszystkie, bądź minimum jeden w dowolnej kolejności)
    - Alternatywa krótka
    - Alternatywa długa
    - Po skosie w dół (automatyczne następstwa event-ów)

Faza 4 - Ludzie (Nie zawsze role) i systemy - kto wchodzi w interakcje z naszym systemem

- Nie każde zdarzenie musi mieć aktora
- **Żółte** ludzie, **Fioletowe** systemy zewnętrzne
- Strukturyzowanie układu eventów przez
    - Klejenie w dół
    - Alternatywa krótka
    - Alternatywa długa
    - Po skosie w dół

Faza 5 - Problemy i okazje

- **Problem** - fioletowa karteczka
- **Okazja** - zielone karteczka

Faza 6

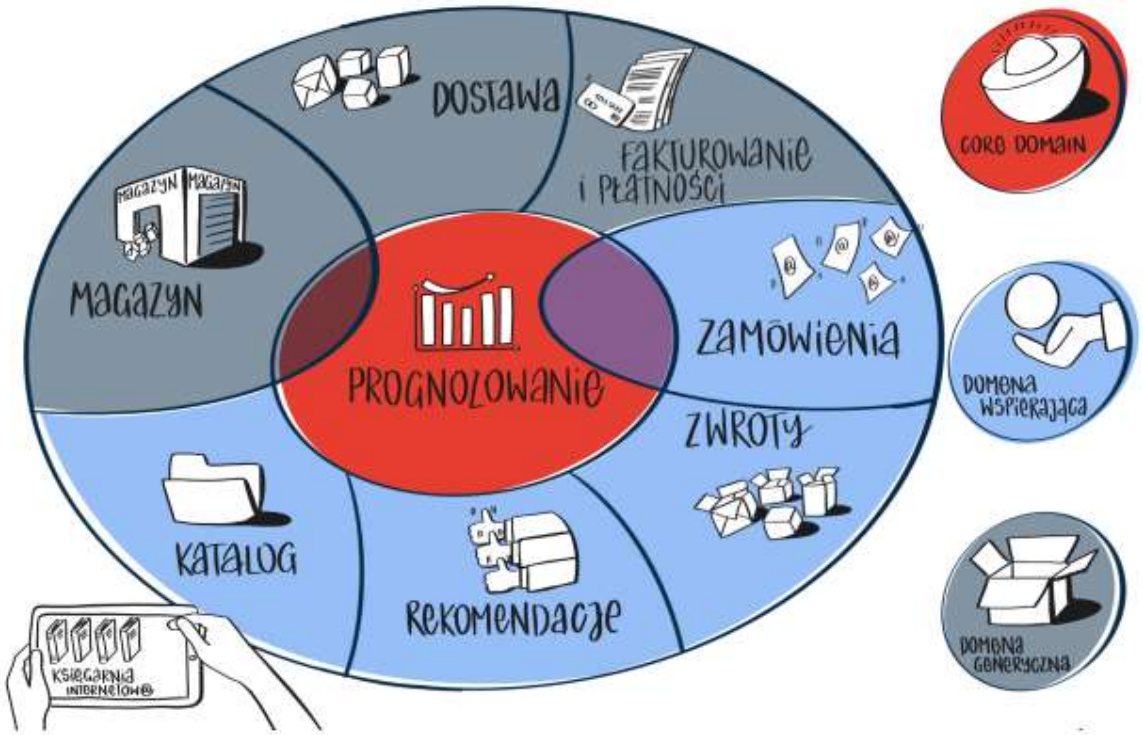
- **Czarnymi strzałkami** pokazujemy najważniejsze **problemy** / **okazje** / **informacje** i tam chcemy usprawnić lub utworzyć proces (na design level es), możliwe, że tam jest Core Domain

Wskazówki

Podczas sesji Event Storming staramy się

- wyłuskać czasowniki - kandydaci na **zdarzenie biznesowe**. Zdarzanie zmienia stan, istotne jest by było to zdarzenie biznesowe z domeny klienta, niekoniecznie myślimy, że ma to być odwzorowane zdarzeniem w systemie komputerowym.
- znaleźć kluczowe **procesy biznesowe** (pomogą w identyfikacji subdomen systemu),
- można już odnajdować **reguły biznesowe** (np. nie można zarezerwować mentora, kiedy jest niedostępny, kleimy żółtą kartkę „Jeśli jest dostępny" przed eventem „Zarezerwowano Mentora")
- znaleźć niewiadome – **HotSpot**-y. Musimy do nich wrócić
- Znaleźć **szanse**, **ryzyka** – kartki zielone oszacowane przez drivery i metryki

Odkrywanie Core Domain i Subdomain



Rysunek 2Źródło: DNA

Rodzaje Subdomain

- **Core Domain** (Najważniejsza, Przewaga rynkowa, najlepszy kod - Taktyczne DDD)
- **Supportive Subdomain** (Nie istnieje gotowy produkt je dostarczający)
- **Generic Subdomain** (Istnieje gotowy produkt je dostarczający)

Odkrywanie Subdomen według poniższych heurystyk (wskazówek)

- Struktura organizacji - podział wg struktury dobra heurystyka na początek
- Różni eksperci domenowi
- Język ekspertów domenowych - np. Książka znaczy co innego w różnych obszarach, więc różne subdomeny
- Patrz na wartość biznesową - coś co daje dużą wartość biznesową jest kandydatem na Subdomain a nawet Core Subdomain
- Patrz na kroki procesu biznesowego
- Pamiętaj, że subdomeny się zmieniają - patrz czy nasz model mentalny podąża, za modele operacyjnym

Event Storming Process Level // rozwinięcie w kursie o architekturze

Po Event Storming na poziomie **Big Picture** możemy kontynuować Event Storming na poziomie **Process Level**. W celu:

- doprecyzowania lub / i utworzenia **Reguł Biznesowych**
- zaadresowania **Hot Spot-ów** poprzez refactoring eventów
- nazwania **Procesów Biznesowych**
- zaadresowania **Szans** i **Ryzyk** – określamy dla nich **Drivery** i **Metryki** refactorujemy pod nie procesy biznesowe
- Akceptacji klienta – pokazujemy mu np. wyeksportowaną do pdf-a tablicę w miro lub udostępniamy link
- wprowadzenia **Polityk** (Polityka- obsługa długo trwającego asynchronicznego procesu)
- Utworzenia **Bounded Context**-ów i ich granic

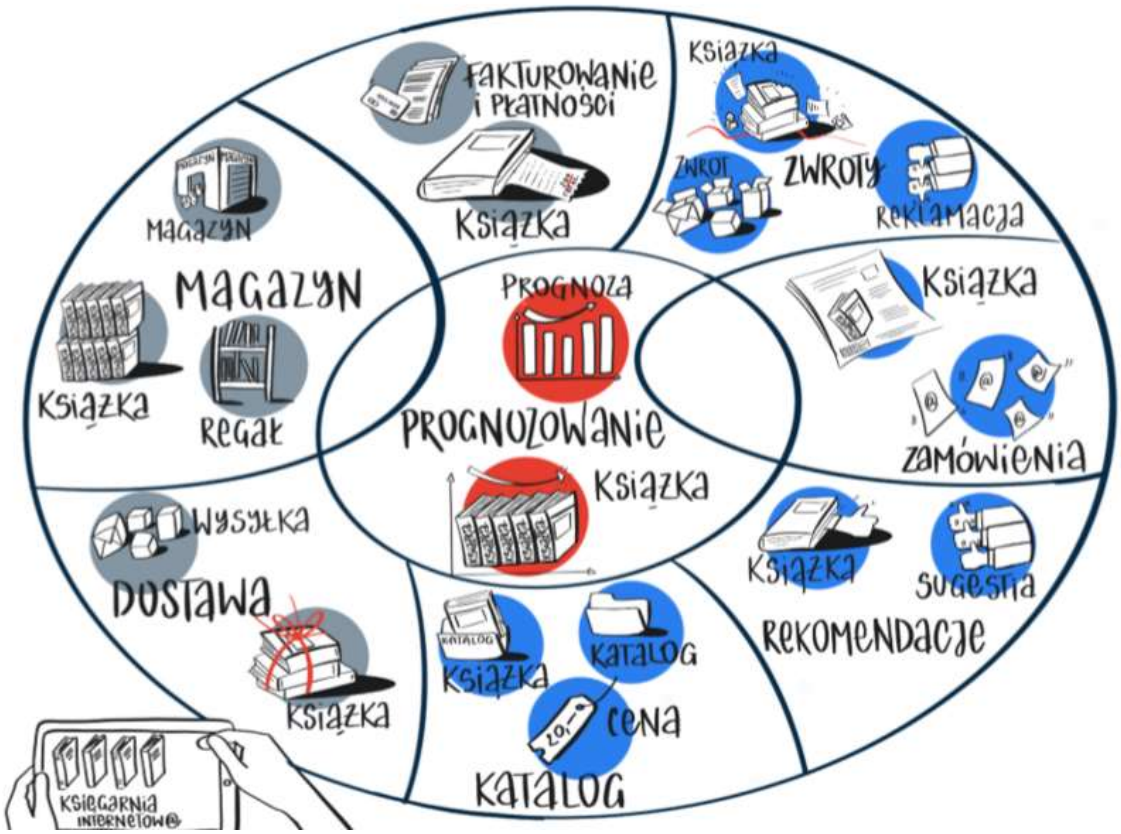
W ramach ES Process Level jesteśmy już w **przestrzeni rozwiązania** (wcześniej była to **przestrzeń problemu / domeny**).

Utworzenie Bounded Context-ów i ich granic// rozwinięcie w kursie o architekturze

W tym momencie przenosimy się z **przestrzeni problemu** (domeny biznesu) **do przestrzeni rozwiązania**

Bounded Context znajduje się już w świecie oprogramowania. Subdomeny są w świecie domeny biznesu. Bounded Contexty są często po prostu odzwierciedleniem subdomen, ale w funkcjonują w przestrzeni oprogramowania. Wydzielamy je by zdefiniować dla nich **Architektury**, **Modele Domenowe, Sposoby komunikacji, API, Use Case-y, Readmodele** – pomocne są do tego modele zaprojektowane w **Event Storming Design Level**. Komunikujemy się z Bounded Contextami za pomocą ich **API**. Używamy w nich tzw. **Ubiquitous Language (Wszechobecny / Jednolity Język)**. Granice Bounded Context-ów identyfikujemy za pomocą heurystyk.

Metafora różnicy między Subdomenami i bounded Contextami to: Do pokoju do podłogi (subdomena) dodajemy dywan (bounded Context)



Rysunek 3:Źródło DNA

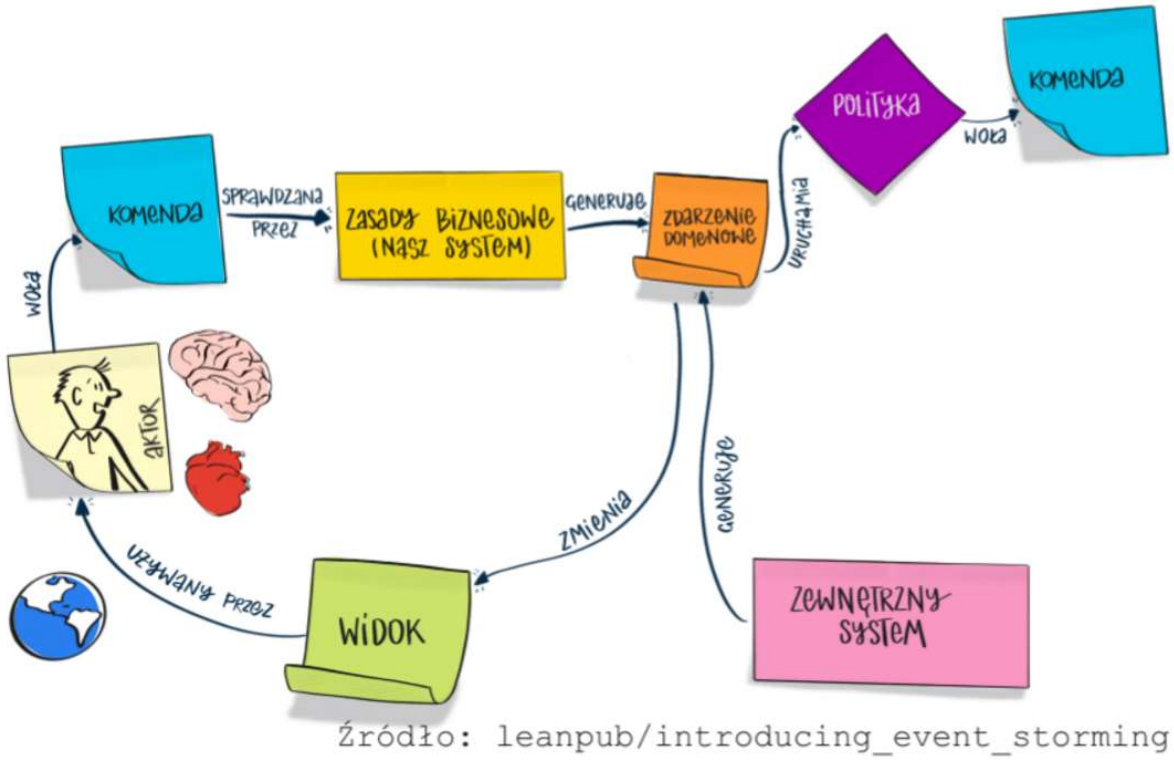
Heurystyki walidowania granic kontekstów i tym samym odkrywania bounded contextów

1. **Autonomia kontekstu** – Bounded Context ma być jak najbardziej autonomiczny
    - Czy nasz Bounded Context może sam podejmować decyzje?
    - Czy musi pytać o zgodę kilka innych Bounded Context?
2. Liczba kontekstów w procesie biznesowym - im **mniej** Bounded Context uczestniczy w danym procesie biznesowym (proces biznesowy jest oznaczony na boardzie Event Storming-owym) tym pewnie **lepiej**
    - Jak wiele kontekstów uczestniczy w danym procesie biznesowym?
3. Szukaj informacji zmieniających się razem
4. Szukaj informacji używanych razem
5. Zadaj sobie pytania
    - Jaka jest odpowiedzialność kontekstu? - jeśli kontekst robi zbyt dużo to może trzeba go rozbić na mniejsze
    - Ile integracji ma kontekst?
    - Czy jest jedno źródło prawdy każdej informacji? - czy każda informacji ma właściciela - kontekst
    - Czy istnieje schizofreniczny kontekst?
        - Np. sprawdza czy klient indywidulany / korporacyjny -> najprawdopodobniej wydzielić 2 konteksty
6. Szukaj anty-wymagań
    - Anty-wymagania to biznesowe reguły nie mające sensu
    - Pomagają przy walidacji granic

Zaprojektowanie modeli pojedynczych komponentów per Bounded Context w Event Storming Design Level //rozwinięcie w kursie o architekturze

Elementy omówione szczegółowo na miro

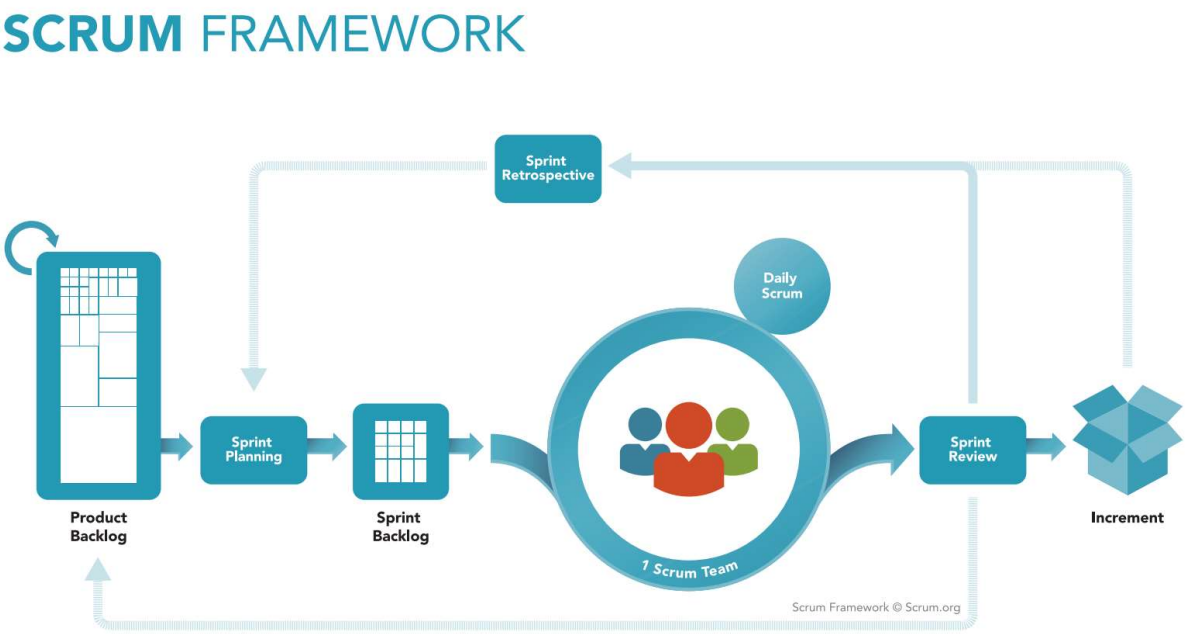
Elementy, z których budujemy modele komponentów w **Event Storming Design Level** – mają bezpośrednie przełożenie na wzorce taktyczne w kodzie – **Agregaty (zbudowane z reguł)**, **Komendy**, **ReadModele**, **Sagi / Process Manager-y**



Źródło: leanpub/introducing_event_storming

SCRUM

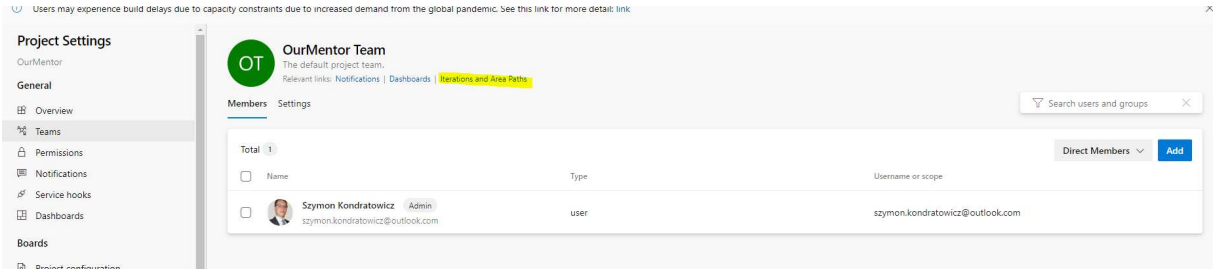
Wsadem do prowadzenia projektu w metodyce **SCRUM** są: Boardy z sesji Event Storming Big Picture, Process Level, Design Level. A w szczególności odkryte Subdomain-y, stworzone Bounded Contexty, nazwane procesy biznesowe, modele poszczególnych komponentów w bounded contextach, modele komunikacji



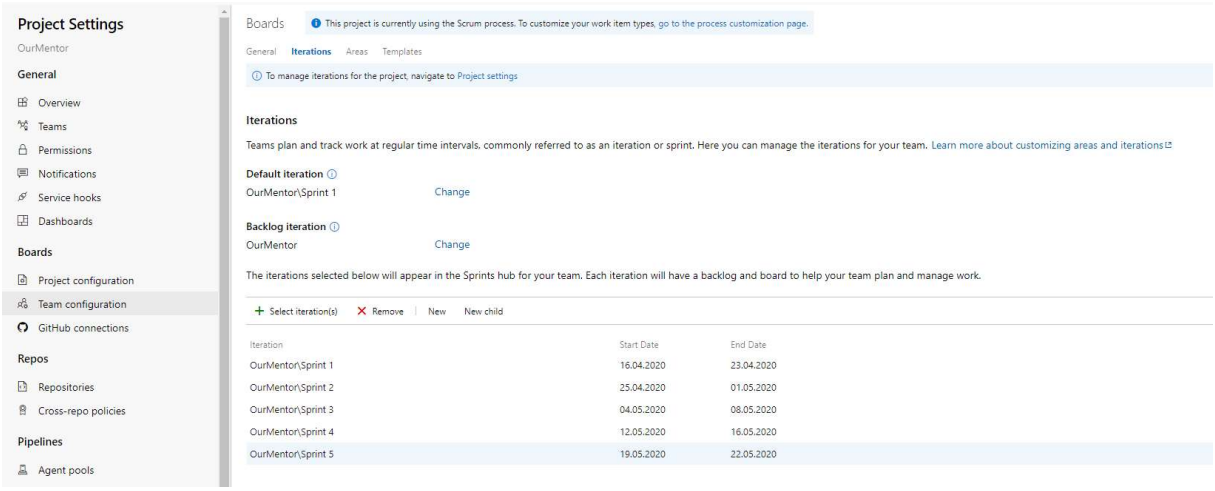
Azure DevOps Boards

- Planowanie, Zarządzanie i Monitorowanie pracy całego Zespołu
- Product Backlog, Sprint Backlog, Task Boards

Praca z Zespołami (Teams), Obszarami (Areas) i Iteracjami (Iterations)



Rysunek 4 zdefiniowanie zespoóu i areas-ów



Rysunek 5 Zdefiniowanie sprintów

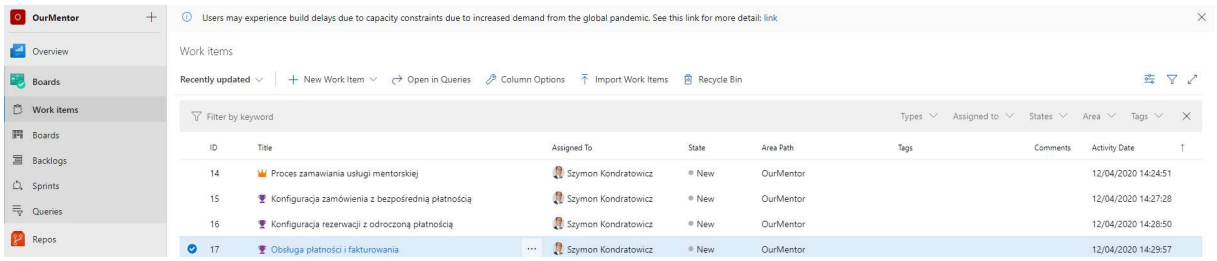
Praca z Work Item-ami na podstawie wyniku event Storming

Na podstawie Event Storming mamy wsad do tworzenia poszczególnych Work Item-ów

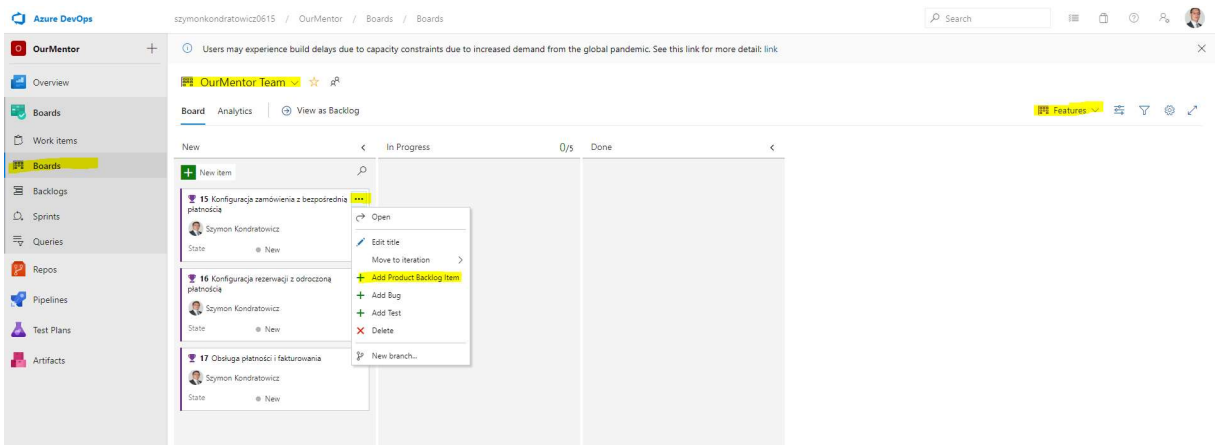
- **Epic** – nazwa procesu
- **Feature** – nazwa podprocesu lub fragmentu procesu
- **Product Backlog Item**
    - Wynika z komendy / polityka + event + integracja / readmodel,
    - Opis sprzed wykonania eventu, jeśli można to z aktorem. Np. Jako klient mogę wybrać jedną lub wiele usług.

- Jeśli eventy event jest automatycznym następstwem, to robimy scalamy taki ciąg eventów w jeden Product Backlog Item

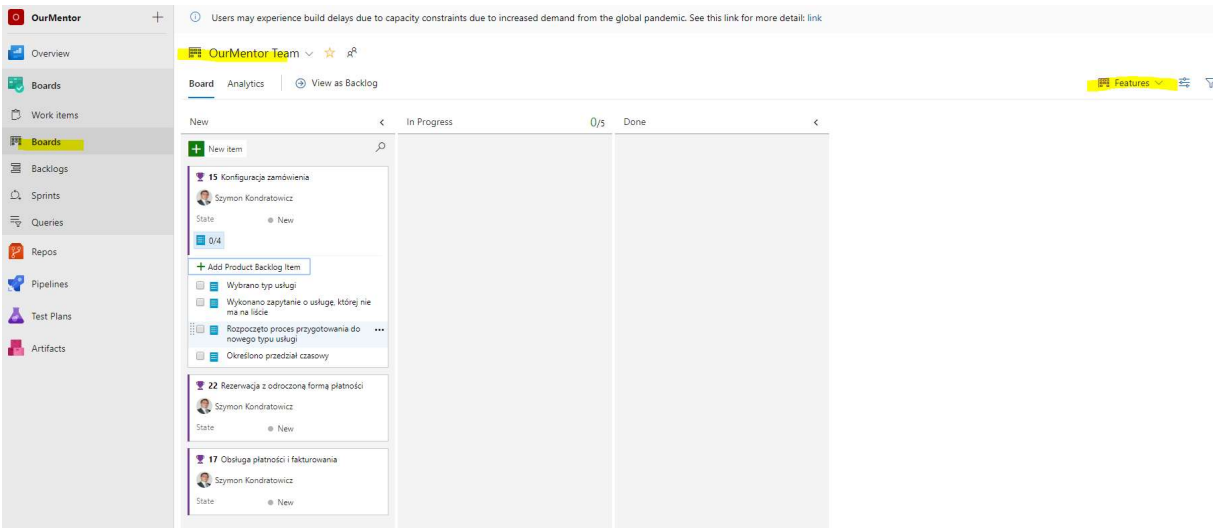
- Product Backlog Item – dzielimy na **Taski** techniczne
  - **Frontend**, **Endpoint**, **Logika domenowa**, **Persystencja**, **Integracja synchroniczna** lub **asynchroniczna** przez **Politykę** (którą implementuje wzorzec **Saga** lub **Process Manager**)
  - Taski techniczne najlepiej robić po ES Design Level – widać wtedy elementy modelu technicznego



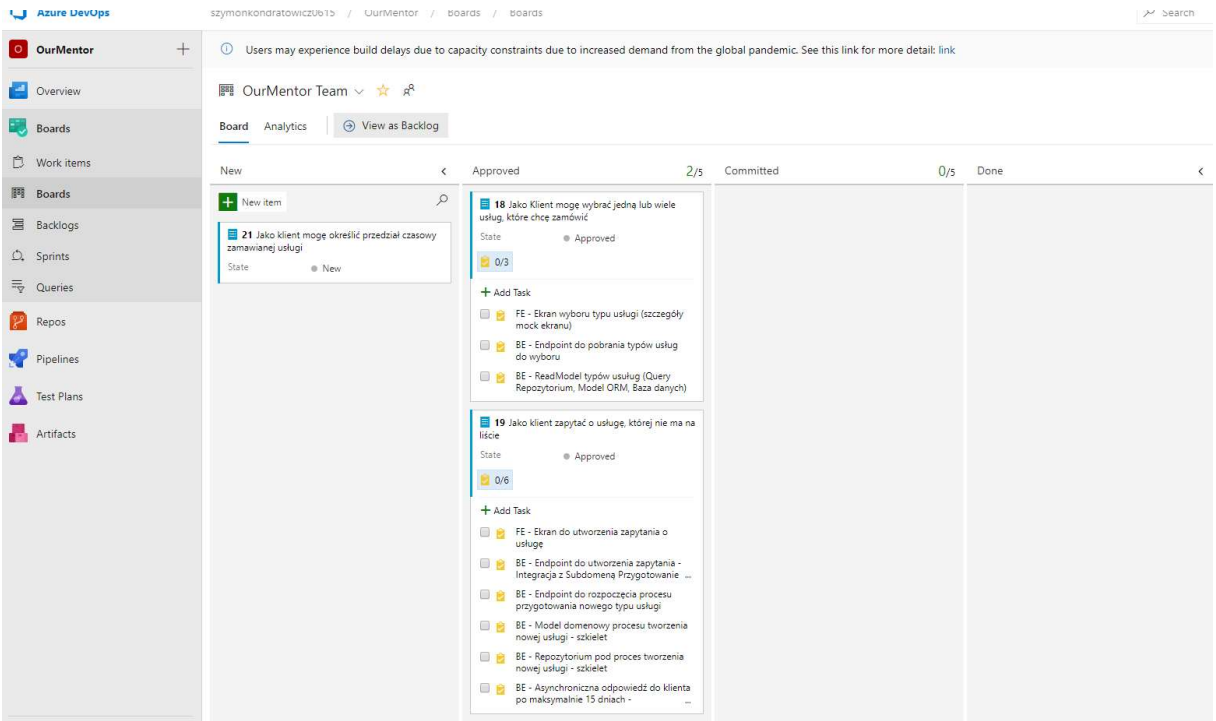
Rysunek 6 Utworzenie Epic-a i zlinkowanych relacją child - parent Featuer-ów



Rysunek 7 Utworzenie Product Backlog Item-ów



Rysunek 8 Utworzenie Product Backlog Item-ów



Rysunek 9 Dodanie tasków do PBI-sów

Zarządzanie Sprintami i Capacity

Planowanie

Tworzenie **Sprint Backlog**-a podczas **Planowania Sprintu** – typowo pierwszy dzień **Sprintu**. Każdy Sprint ma określony czas trwania (z reguły 1 lub 2 tygodnie). Podczas planowania **Product Owner** identyfikuje **Product Backlog Item**-y, które powinny być zrealizowane w trakcie sprintu. Product Backlog Itemy mają pole **Priority**, które stanowi o ich **ważności** i **kolejności** wykonania.

1 faza planowania sprintu

Zidentyfikowanie, które Product Backlog Itemy wejdą do Backlog sprintu, a które przesuwamy na następne sprinty. Product Backlog Item-y mają pole Effort, które można wypełnić np. elementem ciągu Fibonacciego. Pozwala wycenić czasochłonność. Pole te uzupełniamy na przykład w **SCRUM Pokerze**.

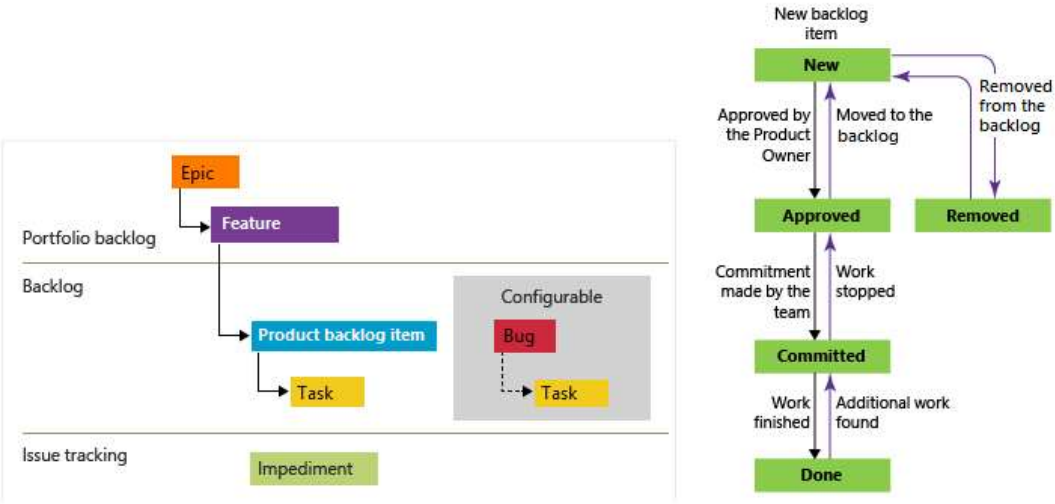
2 faza planowania

Identyfikacja i wycena tasków, wycena już może nastąpić w godzinach, robi ją zespół developerski, można również użyć SCRUM Poker-a.

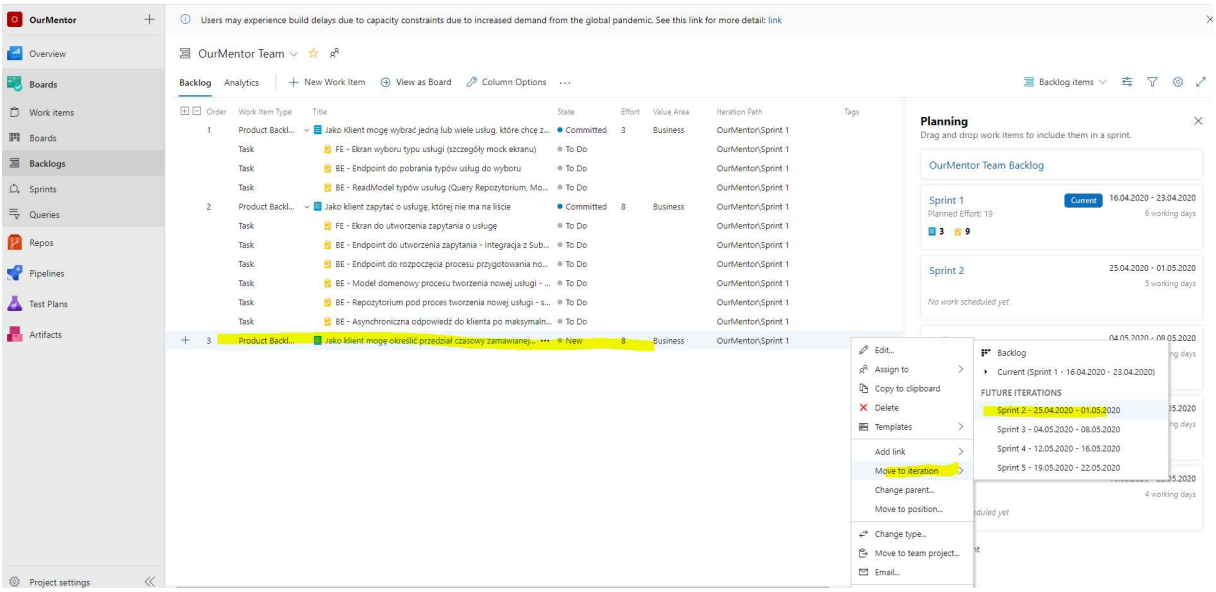
3 faza planowania sprintu

Sprawdzenie **Capacity** zespołu po wycenie i ewentualne przesunięcie PBI-sów z Task-ami na przyszły sprint, lub dobranie nowych.

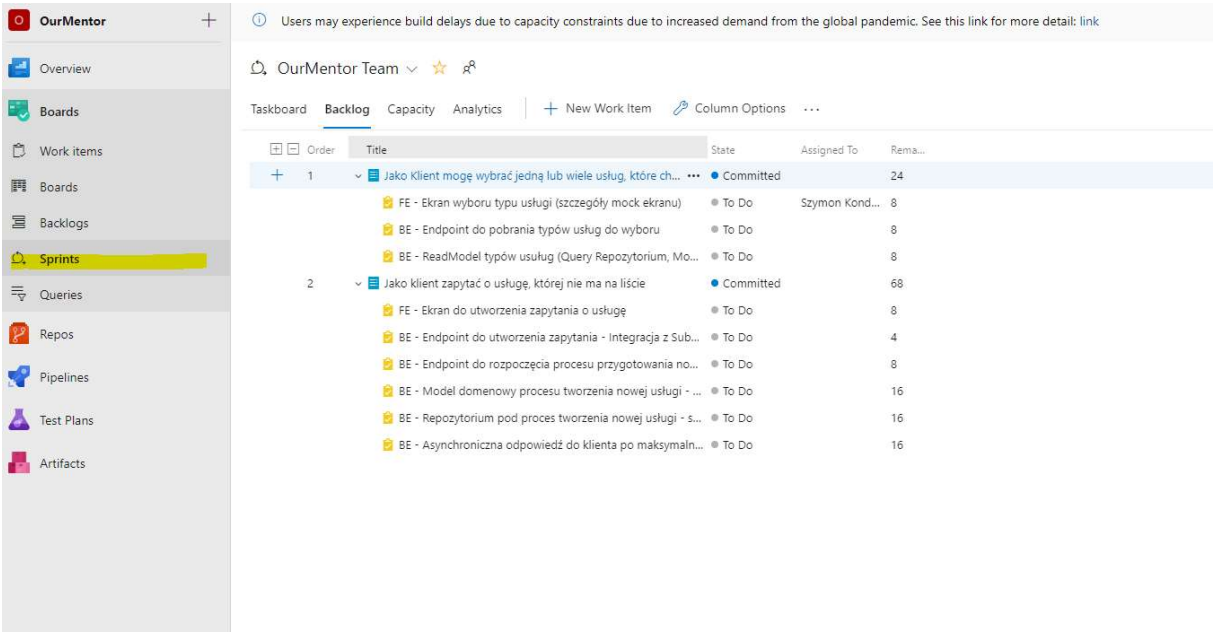
Po planowaniu następuję **praca nad Task-ami**. Według **flow przejść**. Rysunek niżej. Przydatną opcją jest dodanie kolumn: **Code Review i QA Tested**. Po zweryfikowaniu kodu w code review i przetestowaniu zostaje oddany na release branche-a.



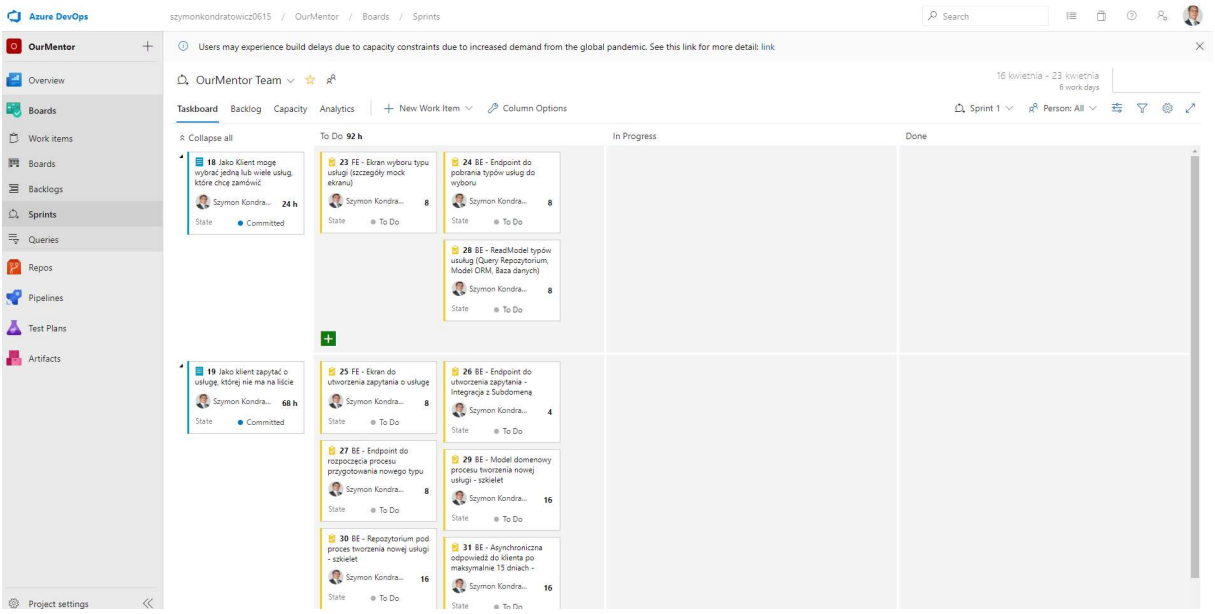
Rysunek 10 SCRUM Proces w AzureDevops



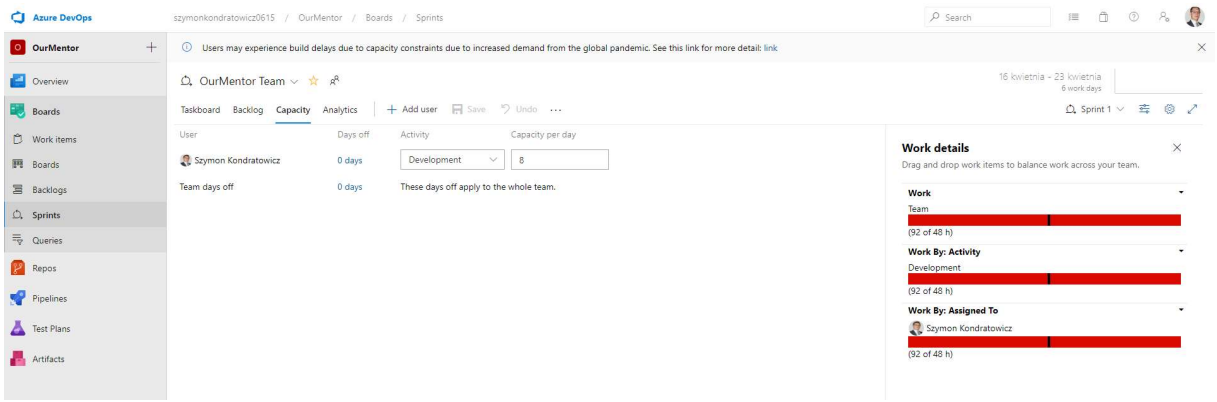
Rysunek 11 Backlog – wykorzystany do planowania sprint - Przesunięcie na przyszły sprint



Rysunek 12 Sprint Backlog



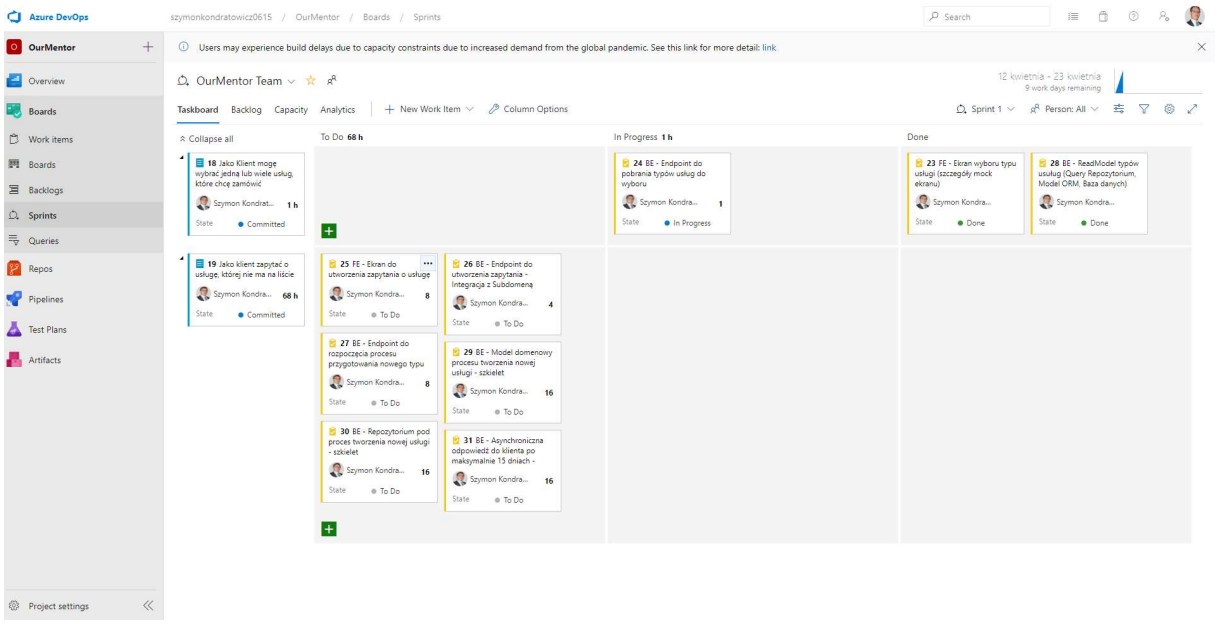
Rysunek 13 Sprint Task Board



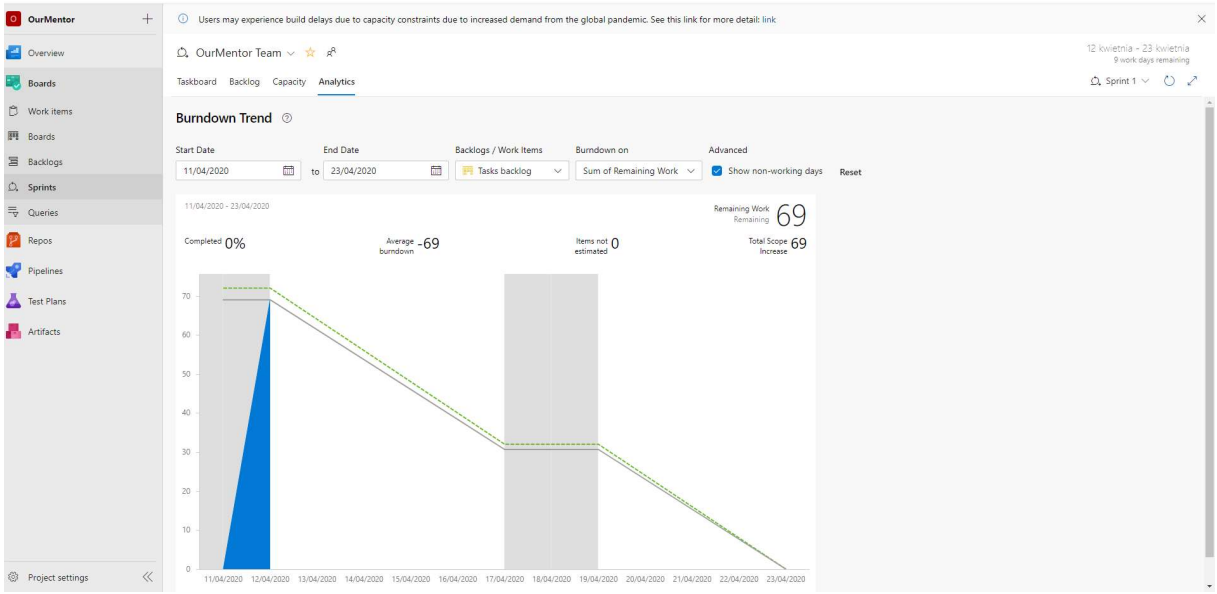
Rysunek 14 Sprint Capacity - analiza obłożenia pracą zespołu

Daily Scrum

Co zrobiłem? Co robię? Co będę robił? Jakie mam problemy? Na każde pytanie odpowiada każdy na widoku Sprint TaskBoard-a, ewentualnie analiza Burndown chart-a



Rysunek 15 Sprint Task Board, na którym realizujemy Daily



Rysunek 16 Burndown chart do oceny postępów pracy

Sprint review

https://www.scrum.org/resources/what-is-a-sprint-review

Srint Retrospective

https://www.scrum.org/resources/what-is-a-sprint-retrospective

Podsumowanie - Event Storming jako mapa Epic-ów, Featuer-ów, Product Backlog Item-ów i Task-ów

Model (board w miro) komunikacji między dwoma Bounded Context-ami poprzez Politykę, gdzie event-y są asynchronicznymi komunikatem z jednej do drugiej subdomeny a Polityka wie, kiedy wysłać e-mail. Polityka modeluje taki asynchroniczny (rozciągnięty w czasie) handler eventów. Widać na tym przykładzie, że model w Event Storming dostarcza dużo więcej informacji niż zestaw opisowych work itemów w AzureDevops (lub innym narzędziu do prowadzenia SCRUM-a).

Źródła

- szkolenie DNA
- https://docs.microsoft.com/pl-pl/azure/devops/?view=azure-devops
- https://www.azuredevopslabs.com/labs/azuredevops/agile/
- scrum.org