# Strategia Testów w Tribe: Ujednolicenie i Automatyzacj

## Streszczenie dla Kierownictwa

**Problem:** Obecne procesy testowe w Tribe są zdecentralizowane, niespójne i w dużej mierze manualne. Prowadzi to do problemów ze stabilnością środowisk, wydłuża czas dostarczania oprogramowania, podnosi koszty utrzymania i utrudnia skalowanie operacji.

**Propozycja:** Wdrożenie centralnej, ujednoliconej platformy testowej w modelu "Testing as a Service". Rozwiązanie oparte na technologii **Docker** dostarczy wszystkim zespołom deweloperskim powtarzalne, izolowane środowiska testowe na żądanie, zautomatyzuje proces uruchamiania testów i scentralizuje raportowanie.

**Kluczowe Korzyści:**

* **Przyspieszenie Time-to-Market:** Drastyczne skrócenie cyklu testowego z dni/godzin do minut.
* **Redukcja Kosztów:** Zmniejszenie liczby błędów na produkcji i ograniczenie manualnej pracy testerów.
* **Wzrost Jakości i Niezawodności:** Eliminacja problemu "działa na mojej maszynie" dzięki spójnym, skonteneryzowanym środowiskom.
* **Zwiększenie Produktywności Deweloperów:** Uwolnienie inżynierów od problemów z infrastrukturą testową.

**Inwestycja:** Projekt wymaga alokacji dedykowanego zespołu (Platform Engineering/DevOps) do budowy i utrzymania platformy oraz zaangażowania zespołów w proces migracji. To strategiczna inwestycja w zdolność ING do szybszego i bardziej niezawodnego dostarczania innowacji.

Spis treści

[Strategia Testów w Tribe: Ujednolicenie i Automatyzacj 1](#_Toc202247969)

[Streszczenie dla Kierownictwa 1](#_Toc202247970)

[1. Wprowadzenie i Cele Strategiczne 2](#_Toc202247971)

[1.1. Kontekst Biznesowy 2](#_Toc202247972)

[1.2. Myśl Przewodnia: Jeden Tribe, Jeden Standard Testowania 2](#_Toc202247973)

[1.3. Główne Cele Inicjatywy 3](#_Toc202247974)

[2. Stan Obecny: Analiza Procesów Testowych 3](#_Toc202247975)

[2.1. Ogólne Wyzwania i Obserwacje 3](#_Toc202247976)

[2.2. Szczegółowa Analiza Zespołów 3](#_Toc202247977)

[2.3. Źródła Danych dla Analizy 6](#_Toc202247978)

[3. Uzasadnienie Strategii i Analiza Korzyści 8](#_Toc202247979)

[3.1. Dlaczego Standaryzacja? Dowody z Rynku 8](#_Toc202247980)

[3.2. Porównanie: Przed i Po Wdrożeniu Platformy 9](#_Toc202247981)

[3.3. Pozytywny Wpływ na Kulturę i Biznes 10](#_Toc202247982)

[4. Wizja Docelowa: Ujednolicona Platforma Testowa 10](#_Toc202247983)

[5. Szczegółowy Proces Orkiestracji Testów 11](#_Toc202247984)

[5.1. Wizualizacja Procesu na Przykładzie Zespołu Mariusza 11](#_Toc202247985)

[5.2. Wizualizacja Cyklu Życia Testu (End-to-End) 11](#_Toc202247986)

[6. Implementacja: Wymagania i Wyzwania 12](#_Toc202247987)

[6.1. Wymagania Technologiczne 12](#_Toc202247988)

[6.2. Wymagane Kompetencje i Rozwój Zespołów 12](#_Toc202247989)

[6.3. Potencjalne Wyzwania i Ryzyka 12](#_Toc202247990)

[6.4. Proponowany Harmonogram Działań (Roadmapa) 12](#_Toc202247991)

[7. Słownik Pojęć 13](#_Toc202247992)

## 1. Wprowadzenie i Cele Strategiczne

### 1.1. Kontekst Biznesowy

W dynamicznie rozwijającym się środowisku technologicznym szybkość i jakość dostarczanego oprogramowania są kluczowymi czynnikami sukcesu. Obecne zróżnicowanie procesów i narzędzi testowych w poszczególnych zespołach Tribe, mimo że pozwala na realizację bieżących zadań, prowadzi do powstawania silosów wiedzy, problemów ze spójnością środowisk oraz utrudnia efektywne zarządzanie jakością na poziomie całej organizacji.

### 1.2. Myśl Przewodnia: Jeden Tribe, Jeden Standard Testowania

Ten dokument przedstawia strategię transformacji podejścia do testowania. Jej myślą przewodnią jest ujednolicenie procesów i stworzenie centralnego, wysoce zautomatyzowanego narzędzia do zarządzania i uruchamiania testów. Celem jest przejście od rozproszonych, często manualnych praktyk do modelu **"Testing as a Service"**, gdzie każdy zespół dysponuje spójnym, niezawodnym i skalowalnym środowiskiem do weryfikacji jakości swoich produktów.

### 1.3. Główne Cele Inicjatywy

* **Usystematyzowanie wiedzy** o testach w całym Tribe.
* **Standaryzacja narzędzi i procesów** w celu zwiększenia efektywności i redukcji kosztów.
* **Stworzenie ujednoliconej, skalowalnej platformy testowej** opartej o konteneryzację.
* **Zwiększenie niezawodności i powtarzalności** testów poprzez eliminację problemów środowiskowych.
* **Skrócenie pętli feedbacku** (czasu od zmiany w kodzie do uzyskania wyniku testu).
* **Stworzenie jednego, centralnego miejsca** do wymiany wiedzy i analizy wyników testów.

## 2. Stan Obecny: Analiza Procesów Testowych

### 2.1. Ogólne Wyzwania i Obserwacje

Na podstawie wstępnego zebrania informacji zidentyfikowano powtarzające się wyzwania:

* **Dokumentacja:** Często nieformalna, nieaktualna lub jej brak.
* **Środowiska:** Problemy z dostępnością i konfiguracją.
* **Automatyzacja:** Zróżnicowany poziom, duże poleganie na testach manualnych.

### 2.2. Szczegółowa Analiza Zespołów

Poniżej znajduje się opisowa analiza strategii testowych w zdiagnozowanych zespołach.

**Zespół Płatności (Agnieszka)**

* **Testy UI:** Nightwatch
* **Testy API (manualne):** Postman
* **Testy Jednostkowe:** Istnieją
* **Wnioski:** Strategia niestabilna, zależna od wolnych testów UI.

**Zespół Backend/PMF (Tomasz)**

* **Testy E2E (BE):** Nightwatch
* **Testy Integracyjne:** Cucumber
* **Testy Jednostkowe:** Istnieją
* **Wnioski:** Lepszy balans, ale ryzyko związane z mockami i współdzielonym środowiskiem.

**Zespół Mariusza**

* **Testy Funkcjonalne:** Cypress
* **Testy Wydajnościowe:** Gatling-ts
* **Testy Jednostkowe:** Istnieją
* **Wnioski:** Dobra automatyzacja.

**Zespół DB9 Sprzedaż (Marta)**

* **Backend:** SPE backend (Java/Python)
* **Frontend:** Front MOSING
* **Procesy testowe:** End to end manualne. Testy automatyczne (frontowe - dodatkowo manualne, backendowe - głównie manualne). Testy manualne na środowisku UAT, uruchamiane lokalnie, testy regresyjne.
* **Narzędzia:** Brak szczegółowych informacji o narzędziach do testów jednostkowych/integracyjnych.
* **Liczba testerów:** 4 osoby (2 osoby bankowe, 2 osoby zewnętrzne). 3 osoby zajmują się tworzeniem testów automatycznych, 1 osoba dedykowana WCAG.
* **Wnioski:** Wysokie poleganie na testach manualnych E2E, co spowalnia proces i ogranicza skalowalność. Istnieje potrzeba automatyzacji i usystematyzowania testów, zwłaszcza API.

**Zespół BLK (Maciej)**

* **Aplikacje:** BLK.
* **Backend:** TAK MIG.
* **Frontend:** Moje Ing.
* **Testy frontowe:** Testy pisane są na Nightwatch JS, procesy Blikkowe.
* **Testy backendowe:** Testy pisane są w Cypress procesy backendowe BLK.
* **Testy wydajnościowe:** Nie ma.
* **Testy manualne:** Wszystkie procesy blikkowe backend i front na mobilne przeglądarki.
* **Uruchamianie testów:** Lokalnie i na środowisku UAT. Jest pipeline do testów automatycznych.
* **Liczba testerów:** 3 osoby.
* **Brak dokumentacji do testów:** W Cypressie komentarze rozpraszane poszczególnych źródeł informacji. Plan testów w googlowych plikach.
* **Wnioski:** Istnieje automatyzacja frontowa i backendowa, ale brakuje testów wydajnościowych i jasnej dokumentacji, co utrudnia utrzymanie i rozwijanie.

**Zespół DB12 (Mariusz K.)**

* **Aplikacje:** Front - MING (desktop, android, iOS, fizyczny), powołanie CAM, Back - Genesis / SPE / SPE2.
* **Testy manualne:** Wszystko: decrypt/prep. Testy funkcjonalne Nightwatch (MING 624) na UAT, testy filmowe.
* **Azure Pipelines:** Testy frontowe regresyjne na dev.
* **Środowiska:** Pilot + 2 zewnętrzne.
* **Testowanie z zewnątrz:** Częściowo wdrożone, automatyzujący, brak wiedzy/umiejętności WCAG.
* **Problemy:** Dysk wspólny, orange sharing (nowe rzeczy są odsłonięte - niekoniecznie aktualne). Tabele z przypadkami do regresji na dysku.
* **Backend:** Hyzenizowanie, pojedyncze regresje.
* **Dostęp do przygotowywania przypadków:** Zależność od innych komponentów WCAG.
* **Rozproszona liczba odpowiedzialnych:** Tak.
* **Brak:** Brak uprawnień fizycznych do testów, ING Fupny, nie przygotowane.
* **Narzędzia:** Jmeter, SoapUI, Postman, SauceLabs, Nightwatch, ICB, Xenodm, Vision.
* **Wnioski:** Duża zależność od testów manualnych, problemy z dostępem do środowisk i zarządzaniem przypadkami testowymi. Brak standaryzacji narzędzi i wiedzy.

**Zespół Marzena (Marzena Kolbik, Tomasz Burzyk)**

* **Aplikacje:** Back (Planner, Agama, Ewi, Korelacja), Front (Nadchodzące i cykliczne płatności, stałe zlecenia, polecenia zapłaty, subskrypcje, Agregator - dodaj konto, przedłuż dostęp, odłącz, odmień).
* **Testy frontowe:** Testy automatyczne w tworzeniu w **Moje ING** oraz testy manualne.
* **Testy backendowe:** Testy automatyczne w tworzeniu w **Moje ING** oraz testy manualne (PS02 tj. Agama, Ewi wykorzystujemy z Thunder Client).
* **Testy wydajnościowe:** Tak, Agama, Ewi. Korzystamy z Gatlinga.
* **Testy manualne/akceptacyjne/regresyjne:** Dla nadchodzących i cyklicznych płatności, czasem dla PS02.
* **Uruchamianie testów:** Lokalnie, na środowisku UAT.
* **Testy wdrożeniowe WCAG:** Czytają pisane automaty od frontu.
* **Procesy testów z frontu / wywoływane z zewnątrz:** Obie opcje.
* **Wnioski:** Duża różnorodność aplikacji i procesów testowych. Istnieją automatyzacje, ale nadal duża część testów jest manualna lub oparta na niestandardowych rozwiązaniach. Brak czasu na rozwój automatyzacji to główne wyzwanie.

### 2.3. Źródła Danych dla Analizy

Powyższe analizy piramid testowych powstały na bazie informacji zebranych podczas spotkań i burzy mózgów z przedstawicielami zespołów. Poniżej znajduje się zestawienie kluczowych punktów.

**Dane wejściowe od Agnieszki (Zespół Płatności)**

* "Główne oparcie na testach frontowych, realizowanych przez automaty Front Nightwatch."
* "Dodatkowe testy API są wykonywane w Postmanie, ich zakres i częstotliwość zależą od potrzeb."
* "Testy Wydajnościowe: Realizowane przy użyciu Gatlinga. Odpalane lokalnie w zależności od potrzeby, nie są uruchamiane w Azure."
* "Pozostałe testy są głównie manualne. Bieżąca liczba testów manualnych uniemożliwia pozyskiwanie nowej wiedzy."
* "Testy jednostkowe istnieją."

**Dane wejściowe od Tomasza (Zespół Backend/PMF)**

* "Główne testy do Nightwatch pokryte są w większości przez niego."
* "Wykorzystuje częściowo Nightwatcha dla testów BE, które działają na UAT i nie na mockach."
* "Wykorzystywany jest framework Cucumber, który jest mockowany."
* "Testy integracyjne aplikacji PMF są wykonywane z wykorzystaniem VDI."
* "Testy Wydajnościowe: Realizowane przy użyciu Gatlinga, związane z przejściem na chmurę."
* "Testy jednostkowe istnieją."

**Dane wejściowe od Marty (Zespół DB9 Sprzedaż)**

* **Aplikacje:** Backend: SPE backend (Java/Python), Frontend: Front MOSING.
* **Procesy testowe:** Proces end to end manualny. Frontowe - dodatkowo manualne testy automatyczne. Backendowe - głównie manualne testy automatyczne. Testy manualne na środowisku UAT, uruchamiane lokalnie, testy regresyjne.
* **Testerzy:** 4 osoby (2 osoby bankowe, 2 osoby zewnętrzne). 3 osoby zajmują się tworzeniem testów automatycznych, 1 osoba dedykowana WCAG.
* **Dokumentacja:** Nie ma dokumentacji do testów. Istnieje tylko lista przypadków. Brak formalnej dokumentacji.
* **Wdrożenie testów w siebie:** Jeśli chodzi o testy manualne to każdy tester wspiera drugiego, natomiast przy pisaniu testów automatycznych każdy robi to sam.
* **Testy wdrożeniowe WCAG:** Testy backendu działają na środowisku UAT.
* **Procesy z frontu / wywoływane z zew:** Nie używają frontu (wywoływane z zewnątrz).
* **Problemy/trudności:** Brak wsparcia dla testów automatycznych, brak czasu na ich rozwój, brak dokumentacji.

**Dane wejściowe od Macieja (Zespół BLK)**

* **Aplikacje:** BLK (Backend: MIG, Frontend: Moje Ing).
* **Testy frontowe:** Pisane na Nightwatch JS, procesy Blikkowe.
* **Testy backendowe:** Pisane w Cypress procesy backendowe BLK.
* **Testy wydajnościowe:** Brak.
* **Testy manualne:** Wszystkie procesy blikkowe backend i front na mobilne przeglądarki.
* **Uruchamianie testów:** Lokalnie i na środowisku UAT, jest pipeline do testów automatycznych.
* **Liczba testerów:** 3 osoby.
* **Dokumentacja:** W Cypressie komentarze rozpraszane poszczególnych źródeł informacji. Plan testów w googlowych plikach.
* **Wnioski:** Największym problemem jest brak czasu na testy automatyczne.

**Dane wejściowe od Mariusza K. (Zespół DB12)**

* **Aplikacje:** Front - MING (desktop, android, iOS, fizyczny), powołanie CAM, Back - Genesis / SPE / SPE2.
* **Testy:** Testy manualne (wszystko: decrypt/prep. Testy funkcjonalne Nightwatch (MING 624) na UAT, testy filmowe. Testy frontowe regresyjne na dev (Azure Pipelines). Manualne UAT, pilot + 2 zew.
* **Testowanie z zew:** Częściowo wdrożone, automatyzujący, brak wiedzy/umiejętności WCAG. Dysk wspólny + orange sharing (nowe rzeczy są odsłonięte - niekoniecznie aktualne). Tabele z przypadkami do regresji na dysku.
* **Backend:** Hyzenizowanie, pojedyncze regresje.
* **Dostęp do przygotowywania przypadków:** Zależność od innych komponentów WCAG.
* **Rozproszona liczba odpowiedzialnych:** Tak.
* **Brak:** Uprawnień fizycznych do testów, ING Fupny, nie przygotowane.
* **Narzędzia:** Jmeter, SoapUI, Postman, SauceLabs, Nightwatch, ICB, Xenodm, Vision.
* **Wnioski:** Brak. (Z treści zdjęcia wynika ogólny brak wymienionych problemów/trudności w obszarze wdrożeń/rozwiązań technologicznych, ale kontekst "brak" w odniesieniu do pytań o problemy sugeruje, że to może być również brak odpowiedzi lub brak zidentyfikowanych problemów w tamtej chwili).

**Dane wejściowe od Marzeny Kolbik i Tomasza Burzyka**

* **Aplikacje:** Backend (Planner, Agama, Ewi, Korelacja), Frontend (Nadchodzące i cykliczne płatności, stałe zlecenia, polecenia zapłaty, subskrypcje, Agregator - dodaj konto, przedłuż dostęp, odłącz, odmień).
* **Testy frontowe:** Tak, automatyczne w tworzeniu w **Moje ING** oraz testy manualne.
* **Testy backendowe:** Tak, automatyczne w tworzeniu w **Moje ING** oraz testy manualne (PS02 tj. Agama, Ewi wykorzystujemy z Thunder Client).
* **Testy wydajnościowe:** Tak, Agama, Ewi. Korzystamy z Gatlinga.
* **Testy manualne/akceptacyjne/regresyjne:** Dla nadchodzących i cyklicznych płatności, czasem dla PS02.
* **Uruchamianie testów:** Lokalnie i na środowisku UAT.
* **Testy wdrożeniowe WCAG:** Czytają pisane automaty od frontu.
* **Procesy testów z frontu / wywoływane z zewnątrz:** Obie opcje.
* **Problemy/trudności:** Brak czasu na rozwój automatyzacji.

## 3. Uzasadnienie Strategii i Analiza Korzyści

### 3.1. Dlaczego Standaryzacja? Dowody z Rynku

Proponowane ujednolicenie procesów nie jest celem samym w sobie, lecz środkiem do osiągnięcia wymiernych korzyści biznesowych. Badania rynkowe i praktyki liderów technologicznych jednoznacznie wskazują na pozytywną korelację między standaryzacją narzędzi a efektywnością dostarczania oprogramowania. To podejście, znane jako **Platform Engineering**, jest obecnie jednym z kluczowych trendów w branży.

#### Przykłady Liderów Technologicznych

Wiele wiodących firm technologicznych zainwestowało w budowę wewnętrznych platform deweloperskich, których kluczowym elementem są zunifikowane systemy testowania:

* **Spotify:** Opracowało wewnętrzną platformę ["Golden Paths"](https://engineering.atspotify.com/2020/08/17/how-we-use-golden-paths-to-solve-fragmentation-in-our-software-ecosystem/), aby ujednolicić proces tworzenia oprogramowania. Jak sami piszą: *"Zapewnienie złotych ścieżek dla typowych zadań, takich jak tworzenie nowego serwisu, uwalnia naszych inżynierów od podejmowania powtarzalnych decyzji i pozwala im skupić się na problemie, który próbują rozwiązać."* Nasza platforma testowa jest właśnie taką "złotą ścieżką" dla zapewnienia jakości.
* **Netflix:** Jest pionierem w dziedzinie inżynierii niezawodności i testowania w środowisku produkcyjnym. Ich narzędzia do "Chaos Engineering" (np. Chaos Monkey) są częścią większej platformy, która pozwala deweloperom na bezpieczne eksperymentowanie i testowanie odporności systemów. To pokazuje, jak dojrzałe organizacje inwestują w zaawansowane, scentralizowane narzędzia do weryfikacji jakości.
* **Uber:** Zbudował platformę do testowania aplikacji mobilnych na dużą skalę, która pozwala na równoległe uruchamianie tysięcy testów na setkach urządzeń. W artykule na swoim blogu inżynierskim wyjaśniają, że standaryzacja była kluczem do skalowania: *"Stworzenie jednej, spójnej platformy pozwoliło nam na redukcję duplikacji wysiłków i zapewnienie spójnych wyników dla wszystkich zespołów mobilnych."*

#### Dowody Naukowe i Branżowe

Raport **DORA (DevOps Research and Assessment)**, corocznie publikowany przez Google, wykazuje, że elitarne organizacje (tzw. "Elite performers") charakteryzują się wysokim stopniem automatyzacji i wykorzystaniem wewnętrznych platform. Jak czytamy w raporcie z 2021 roku: *"Wewnętrzne platformy, które przyspieszają rozwój i dostarczanie, są silnym wskaźnikiem najwyższej wydajności"*. Takie platformy redukują obciążenie kognitywne deweloperów, pozwalając im skupić się na tworzeniu wartości biznesowej, a nie na walce z infrastrukturą.

"Organizacje z wewnętrznymi platformami mają 1.6 raza większe prawdopodobieństwo osiągnięcia, a nawet przekroczenia, swoich celów biznesowych." - Puppet, State of DevOps Report

Martin Fowler, jeden z sygnatariuszy Manifestu Agile, w swoim artykule na temat [Piramida Testów](https://martinfowler.com/bliki/TestPyramid.html), podkreśla znaczenie solidnych fundamentów w postaci szybkich testów jednostkowych. Nasza propozycja bezpośrednio adresuje ten problem, promując zrównoważoną strategię, w przeciwieństwie do obecnych, często niestabilnych "odwróconych piramid".

### 3.2. Porównanie: Przed i Po Wdrożeniu Platformy

Wprowadzenie ujednoliconej platformy testowej przyniesie fundamentalną zmianę w codziennej pracy zespołów. Poniższa tabela ilustruje kluczowe różnice.

| Aspekt | Stan Obecny (Bez Platformy) | Stan Docelowy (Z Platformą) |
| --- | --- | --- |
| **Środowiska Testowe** | Niespójne, często konfigurowane manualnie, zawodne ("u mnie działa"). | W pełni zautomatyzowane, spójne, izolowane i powtarzalne (Docker). |
| **Uruchamianie Testów** | Często manualne, lokalne, zależne od dostępności konkretnej osoby. | Na żądanie przez każdego członka zespołu (CLI/API), zintegrowane z CI/CD. |
| **Pętla Feedbacku** | Długa, wyniki testów dostępne po wielu godzinach lub dniach. | Krótka, wyniki dostępne w ciągu minut dzięki równoległemu wykonaniu. |
| **Analiza Wyników** | Rozproszona, brak centralnego miejsca, trudna analiza trendów. | Scentralizowany portal z raportami, historia wykonań, analiza trendów. |
| **Wdrożenie Nowego Testera** | Długotrwałe, wymaga przekazywania nieudokumentowanej wiedzy. | Szybkie, oparte na jasnej dokumentacji i jednym, standardowym procesie. |

### 3.3. Pozytywny Wpływ na Kulturę i Biznes

Inwestycja w platformę testową to nie tylko zmiana technologiczna, ale przede wszystkim kulturowa. Jak dowodzą autorzy książki **"Accelerate: The Science of Lean Software and DevOps"**, organizacje o wysokiej wydajności traktują jakość jako wspólną odpowiedzialność całego zespołu, a nie zadanie jednego działu. Nasza platforma wspiera tę filozofię, dając deweloperom proste narzędzia do samodzielnego uruchamiania testów i weryfikacji swoich zmian. Prowadzi to do wcześniejszego wykrywania błędów, co drastycznie obniża koszt ich naprawy i podnosi ogólną jakość produktu. Wprowadzenie takiego rozwiązania to strategiczna inwestycja w zdolność całej organizacji do szybszego i bezpieczniejszego dostarczania wartości klientom.

## 4. Wizja Docelowa: Ujednolicona Platforma Testowa

Celem jest stworzenie wewnętrznej platformy, która abstrahuje od testera złożoność konfiguracji środowiska. Tester skupia się na pisaniu wartościowych scenariuszy testowych, a platforma dostarcza na żądanie w pełni skonfigurowane, izolowane i powtarzalne środowisko do ich uruchomienia.

**Ekosystem oparty o kontenery Docker**

* Pojedynczy Kontener: Zespół A - Testy UI (Docker: Cypress)
* Pojedynczy Kontener: Zespół D - Testy UI (Docker: Nightwatch)
* Pojedynczy Kontener: Zespół B - Testy BE (Docker: Cucumber)
* Pojedynczy Kontener: Zespół C - Funkcjonalne (Docker: Cypress)
* Pojedynczy Kontener: Zespół A - API (Docker: Gatling)
* Kolejne Kontenery...

## 5. Szczegółowy Proces Orkiestracji Testów

### 5.1. Wizualizacja Procesu na Przykładzie Zespołu Mariusza

Poniższy opis ilustruje, jak piramida testów konkretnego zespołu wpisuje się w zautomatyzowany proces orkiestracji, od wywołania polecenia do uzyskania artefaktów.

**Przykład: Proces Orkiestracji dla Zespołu Mariusza**

Piramida Testów Zespołu Mariusza:

* Funkcjonalne (Cypress)
* Wydajnościowe (Gatling-ts)
* Jednostkowe (Istnieją)

**Przepływ Procesu:**

1. Polecenie uruchomienia

$ ing test mariusz-suite --all

1. **Platforma Testowa tworzy zadania Dockerowe**
   * Zadanie 1: Uruchomienie Kontenera z testami funkcjonalnymi (Docker: cypress:latest)
   * Zadanie 2: Uruchomienie Kontenera z testami wydajnościowymi (Docker: gatling-ts:latest)
2. Kontenery wysyłają artefakty do Storage

(Raporty HTML, filmy .mp4, zrzuty .png, logi .txt)

### 5.2. Wizualizacja Cyklu Życia Testu (End-to-End)

Poniżej przedstawiono pełny, zautomatyzowany przepływ pracy - od napisania kodu przez dewelopera, aż po wizualizację wyników.

1. **Developer & Git**: Deweloper wysyła kod do repozytorium Azure
2. **CI/CD Pipeline**: Azure Pipelines automatycznie buduje **obraz Docker**
3. **Container Registry**: Nowy **obraz Docker** jest umieszczany w Rejestrze Kontenerów
4. **Platforma Testowa**: Test jest wywoływany przez CLI lub automatycznie, platforma uruchamia **kontener Docker**
5. **Test Execution**: Testy są wykonywane w nowym, izolowanym **kontenerze Docker**
6. **Artefakty do Storage**: Wyniki i raporty są zapisywane w Azure Blob Storage
7. **Portal z Raportami**: Portal z raportami wizualizuje wyniki testów

## 6. Implementacja: Wymagania i Wyzwania

### 6.1. Wymagania Technologiczne

* **Platforma Konteneryzacji (Docker):** Dostęp i konfiguracja środowiska Docker z odpowiednimi uprawnieniami.
* **Rejestr Kontenerów:** Skonfigurowany Azure Container Registry (ACR) do przechowywania **obrazów Docker**.
* **Storage:** Utworzenie konta Azure Blob Storage na artefakty.
* **Narzędzia CI/CD:** Zaawansowana konfiguracja Azure DevOps Pipelines do budowania i publikowania obrazów oraz wywoływania testów.
* **Narzędzie CLI:** Stworzenie i dystrybucja własnego narzędzia CLI.
* **Portal z Raportami:** Wdrożenie i konfiguracja aplikacji do raportowania.

### 6.2. Wymagane Kompetencje i Rozwój Zespołów

* **Docker:** Wszyscy testerzy i deweloperzy muszą posiadać podstawową wiedzę na temat konteneryzacji i tworzenia plików Dockerfile.
* **Automatyzacja Infrastruktury:** Wymagana jest grupa specjalistów (np. Platform/DevOps Team) odpowiedzialna za utrzymanie i rozwój infrastruktury do uruchamiania kontenerów.
* **Kultura Jakości:** Przesunięcie odpowiedzialności za jakość w lewo ("shift-left"). Konieczne będzie promowanie i egzekwowanie pisania testów jednostkowych przez deweloperów.

### 6.3. Potencjalne Wyzwania i Ryzyka

* **Krzywa uczenia się:** Adaptacja do nowych narzędzi (**Docker**) może zająć czas.
* **Złożoność początkowa:** Skonfigurowanie całej platformy od zera jest złożonym zadaniem inżynierskim.
* **Opór przed zmianą:** Konieczność przekonania zespołów do zmiany sprawdzonych, choć nieefektywnych, nawyków.
* **Koszty:** Chociaż docelowo platforma ma optymalizować koszty, jej budowa i utrzymanie generują początkowe wydatki.

### 6.4. Proponowany Harmonogram Działań (Roadmapa)

* **Q3 2025:** Sfinalizowanie zbierania informacji ze wszystkich zespołów. Powołanie zespołu odpowiedzialnego za budowę platformy (Proof of Concept). Rozpoczęcie prac nad PoC: konteneryzacja testów jednego zespołu.
* **Q4 2025:** Prezentacja działającego PoC. Decyzja o pełnym wdrożeniu. Rozpoczęcie cyklu szkoleń z **Dockera** dla zespołów.
* **Q1 2026:** Wdrożenie produkcyjnej wersji platformy. Migracja pierwszych 2-3 zespołów na nowe rozwiązanie.
* **Q2 2026 i dalej:** Stopniowa migracja kolejnych zespołów. Ciągły rozwój platformy i dodawanie nowych funkcjonalności.

## 7. Słownik Pojęć

* **Docker:** Platforma do tworzenia, wdrażania i uruchamiania aplikacji w kontenerach.
* **Kontener:** Lekka, samodzielna, wykonywalna paczka oprogramowania, która zawiera wszystko, co jest potrzebne do jej uruchomienia.
* **CLI (Command-Line Interface):** Interfejs wiersza poleceń.
* **Artefakty:** Pliki wynikowe generowane podczas procesu budowania lub testowania (np. raporty, logi).