PADŁEŚ? POWSTAŃ! TEMPORAL.IO

Marcin Baranowski

WJUG 2024

Agenda

- Przeprawy przez przelewy
- Padłeś? Powstań! Io, Io, Io,Temporal.io
- Coś śmiesznego (nieobowiązkowo)
- Niezręczna cisza po nieudanym żarcie
- Kompulsywne stukanie w klawiaturę
- Blok reklamowy
- 

JAK UNIKAĆ STRESU?

CZYM JEST STRES?

stres [ang.], *psychol. stan obciążenia systemu regulacji psychicznej powstający w sytuacji zagrożenia, utrudnienia lub niemożności realizacji ważnych dla jednostki celów, zadań, wartości.*

Źródło: https://encyklopedia.pwn.pl/

CZYM JEST STRES?

JAK UNIKAĆ STRESU?

MAKSYMALIZACJA SPOKOJU

PISZMY KULOODPORNE APLIKACJE

PRZELEW BANKOWY

Przelew bankowy

Przelew

```
transferWorkflow.transfer("acc1", "acc2", "id", 1000)
```

Retry

```
while failure:
    transferWorkflow.transfer("acc1", "acc2", "id", 1000)
```

Retry

```
while failure && retryNo <= maxRetries:
    transferWorkflow.transfer("acc1", "acc2", "id", 1000)
```

Wait

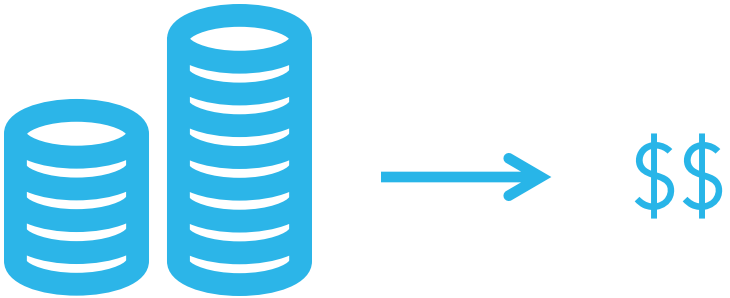
```
while failure && retryNo <= maxRetries:
    transferWorkflow.transfer("acc1", "acc2", "id", 1000)
    wait()
```

Debounce

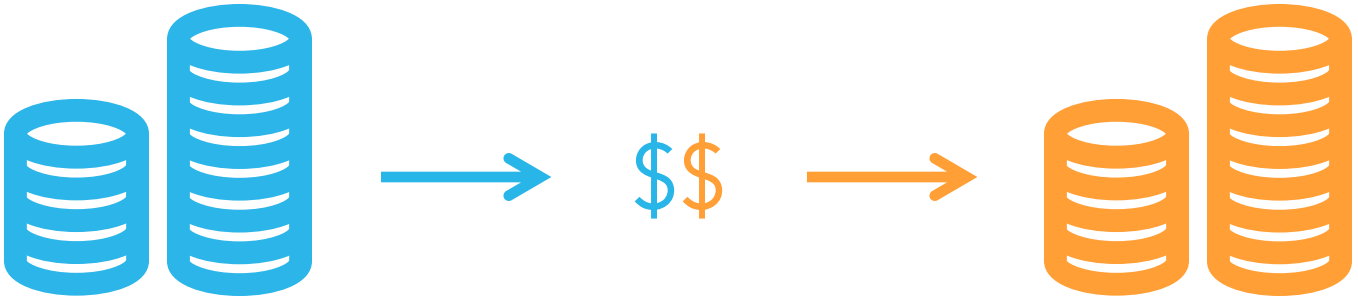
```
while failure && retryNo <= maxRetries:
    transferWorkflow.transfer("acc1", "acc2", "id", 1000)
    debounce()
```

Przelew – 2 kroki

Przelew – 2 kroki



Przelew – 2 kroki



Przelew – 2 kroki

```
transferWorkflow.transfer("acc1", "acc2", "id", 1000)
```

Przelew – 2 kroki

```
transfer(String fromAcc, String toAcc, String id, int amount):
  accountService.withdraw(fromAcc, id, amount)
  accountService.deposit(toAcc, id, amount)
```

Przelew – zwrot

```
transfer(String fromAcc, String toAcc, String id, int amount):

  accountService.withdraw(fromAcc, id, amount)
  try:
    accountService.deposit(toAcc, id, amount)
  catch:
    accountService.deposit(fromAcc, id, amount)    // zwrot
```

Happy path vs Non happy path

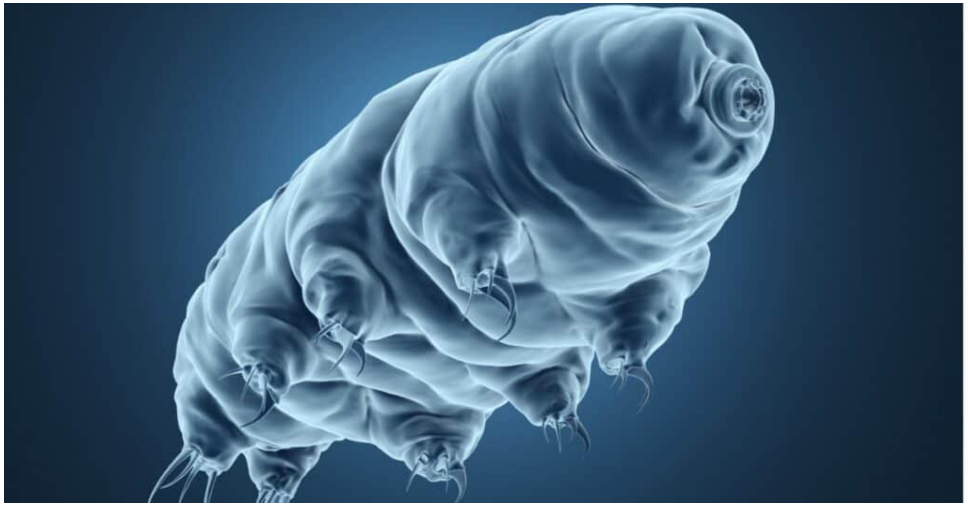
```
accountService.withdraw(fromAcc, id, amount);
accountService.deposit(toAcc, id, amount);
```

```
int tryNo = 1;
boolean success = false;
int debounce = initialDebounce * 1000;  // milliseconds

while (!success && tryNo <= maxTries) {
  try {
    accountService.withdraw(fromAcc, id, amount);
    success = true;
  } catch (Exception e) {
    Thread.sleep(debounce);
    debounce *= debounceFactor;
    ++tryNo;
  }
}

if (!success) {
  throw new RuntimeException("Aborting due to unsuccessful withdraw");
}

tryNo = 1;
success = false;
debounce = initialDebounce * 1000;  // milliseconds

while (!success && tryNo <= maxTries) {
  try {
    accountService.deposit(toAcc, id, amount);
    success = true;
  } catch (Exception e) {
    Thread.sleep(debounce);
    debounce *= debounceFactor;
    ++tryNo;
  }
}

if (!success) {
 accountService.deposit(fromAcc, id, amount);
}
```

TEMPORAL.IO

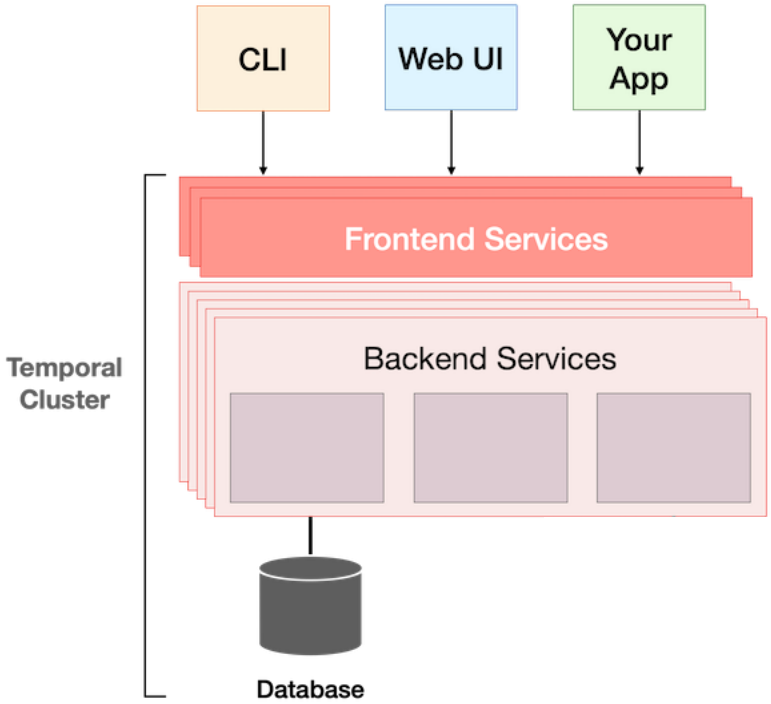
Build invincible applications

Co znaczy “invincible”?



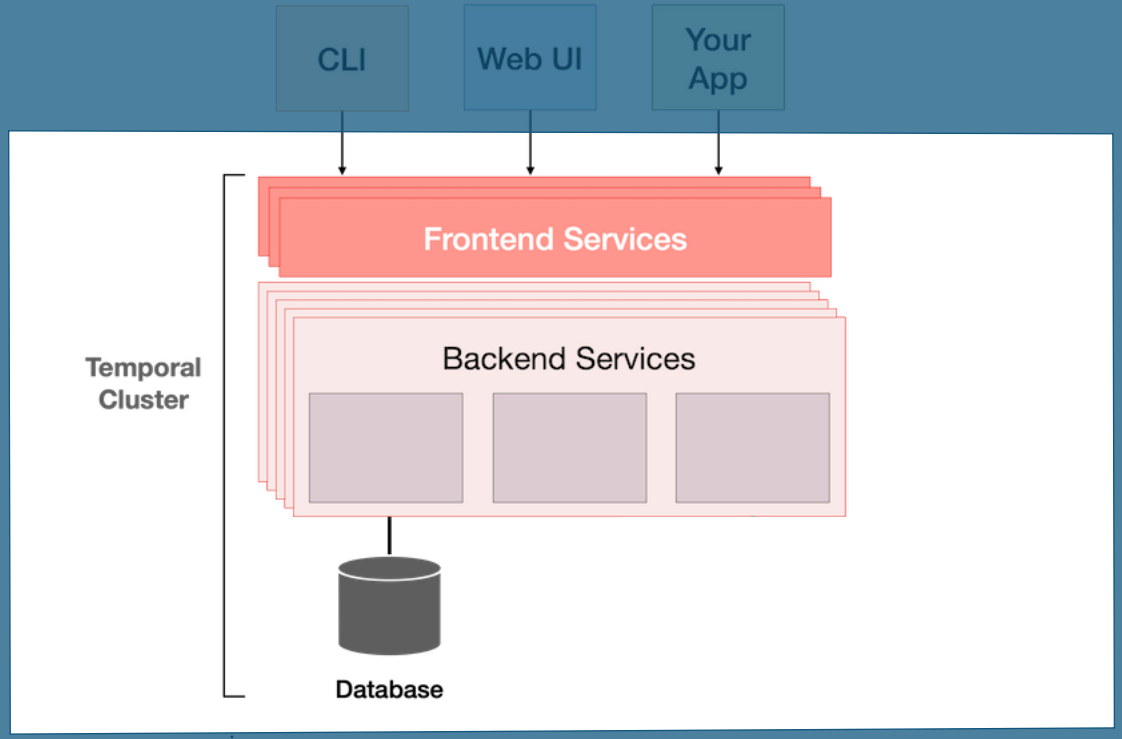
TEMPORAL.IO

Temporal.io – architektura



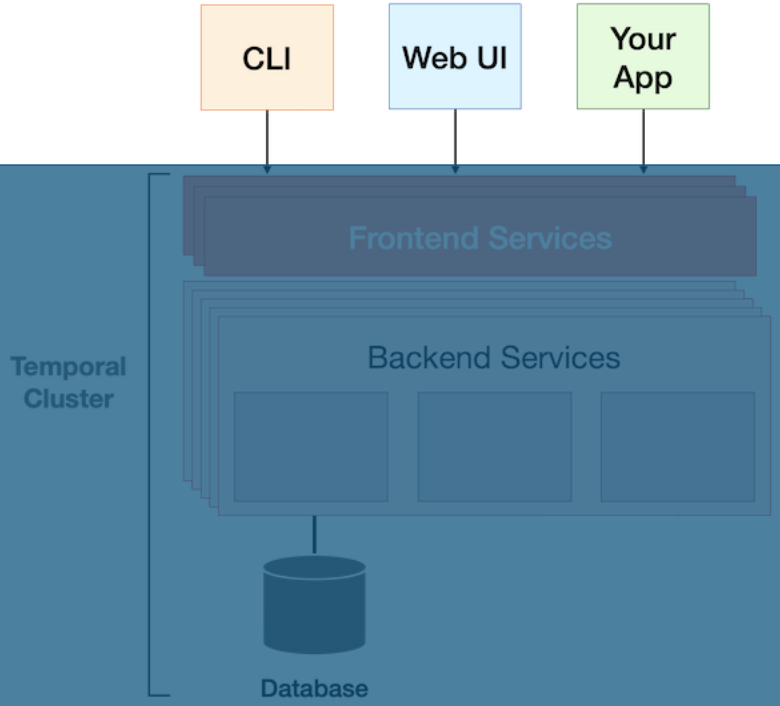
Źródło: temporal.io

Temporal.io – architektura



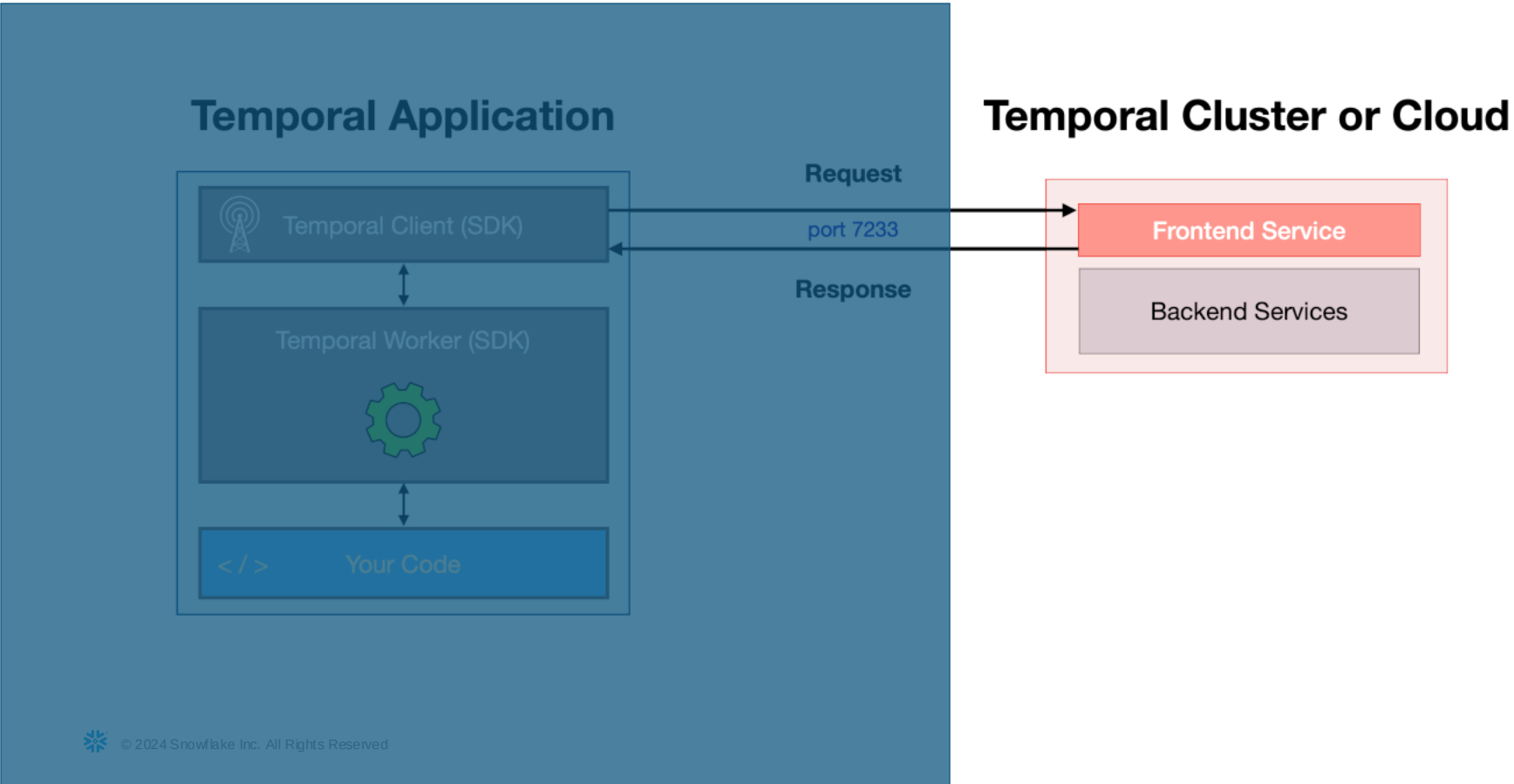
Źródło: temporal.io

Temporal.io – architektura

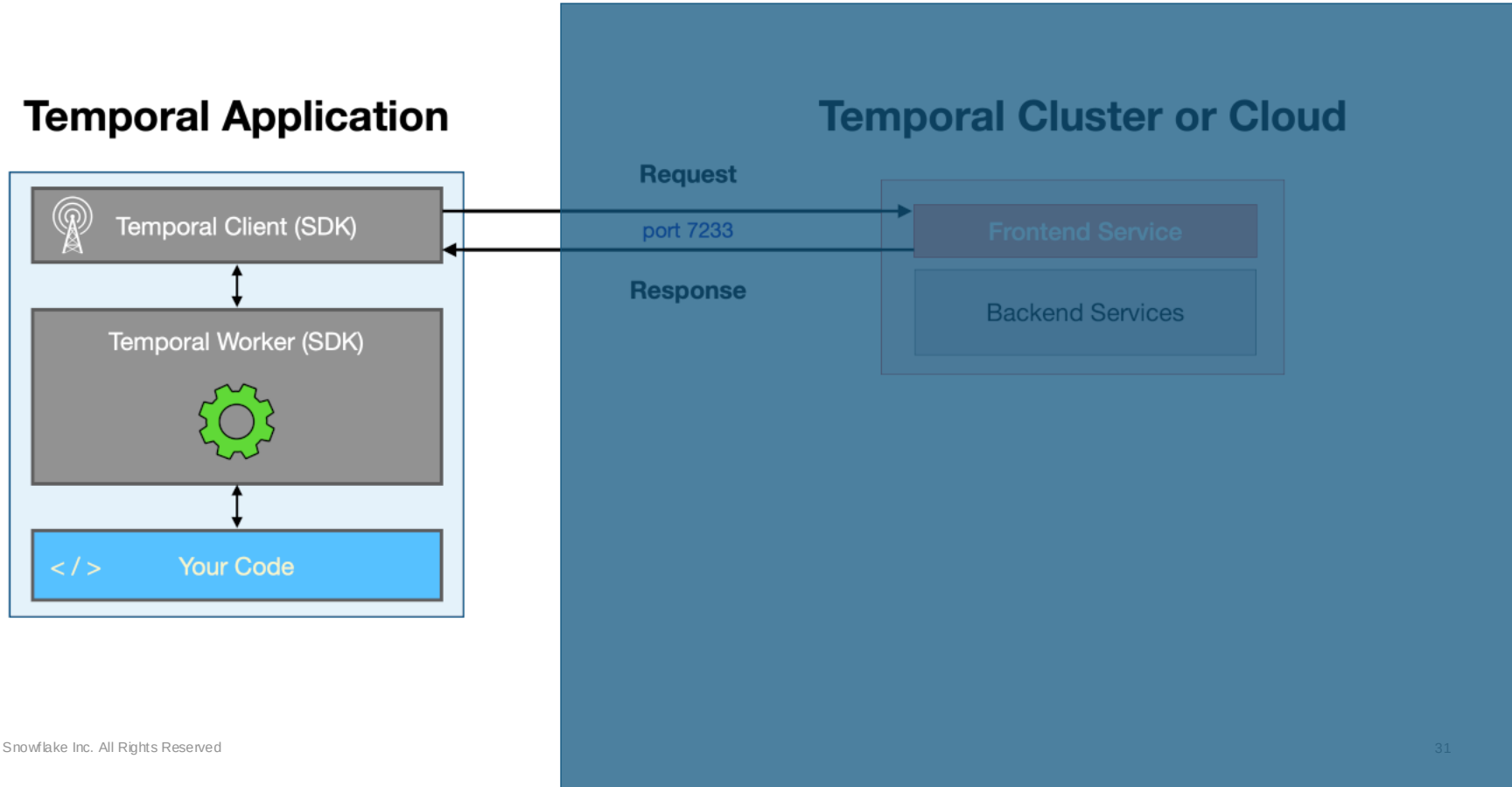


Źródło: temporal.io

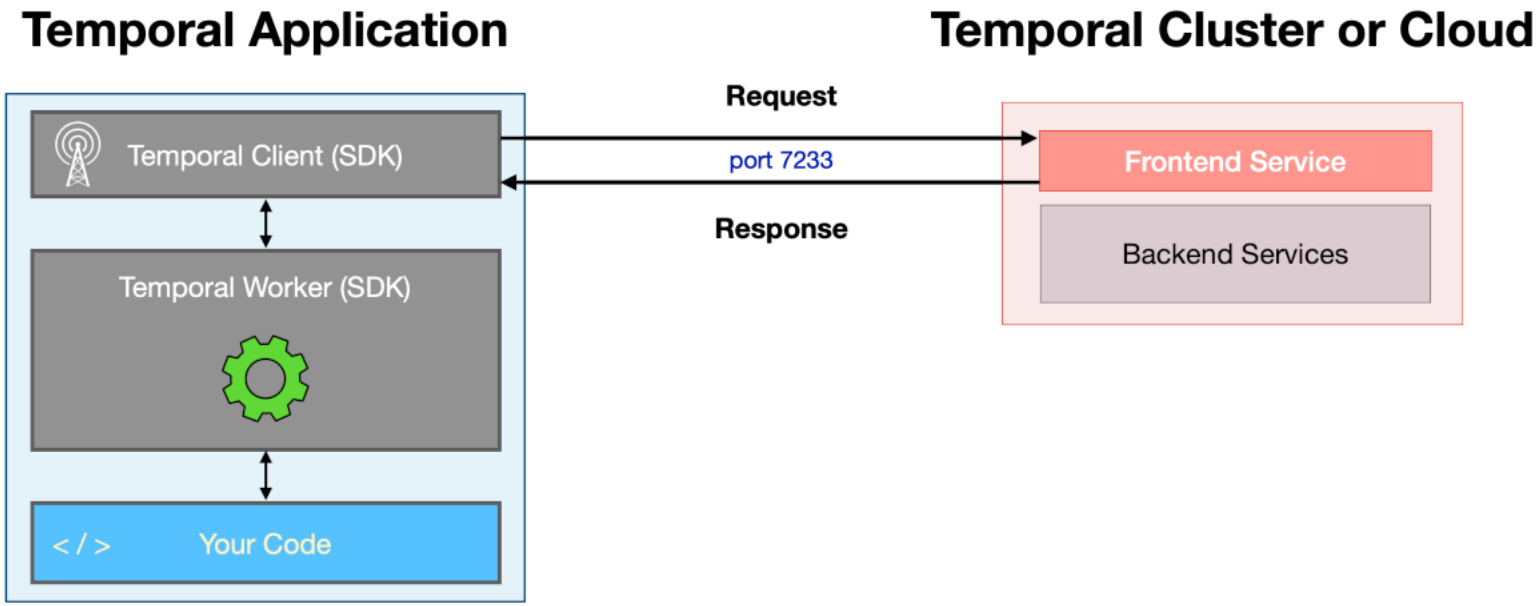
Temporal.io – komunikacja



Temporal.io – komunikacja



Temporal.io – komunikacja



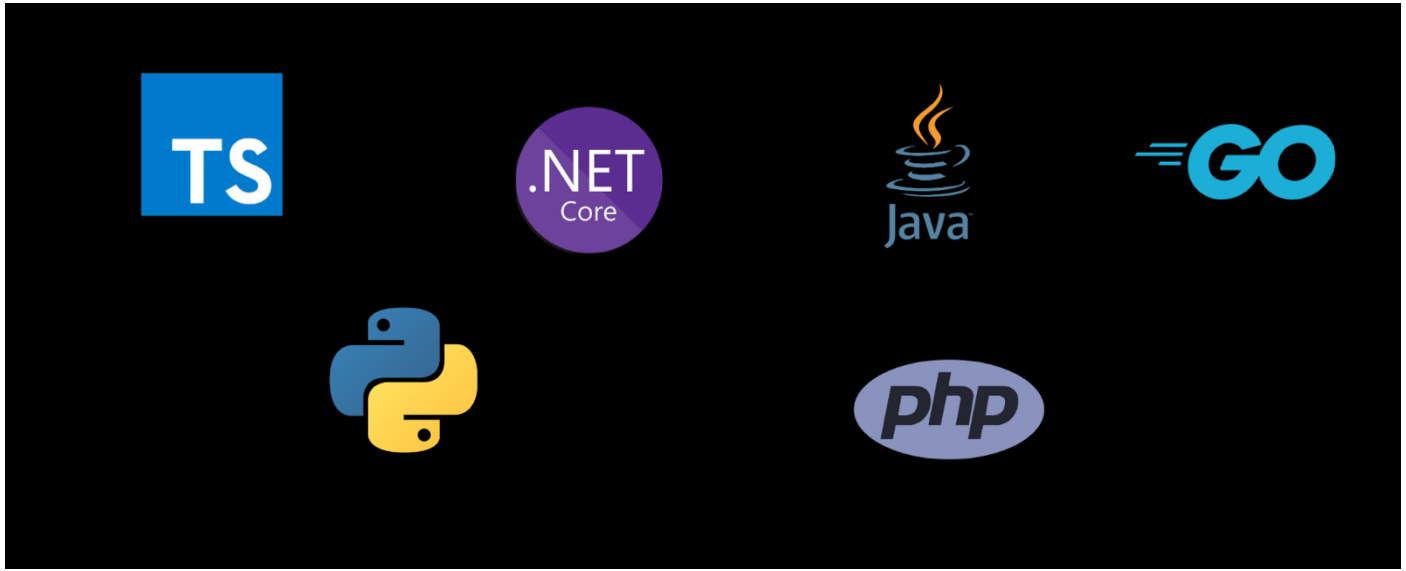
Frontend

Serwer gRPC

Port: 7233

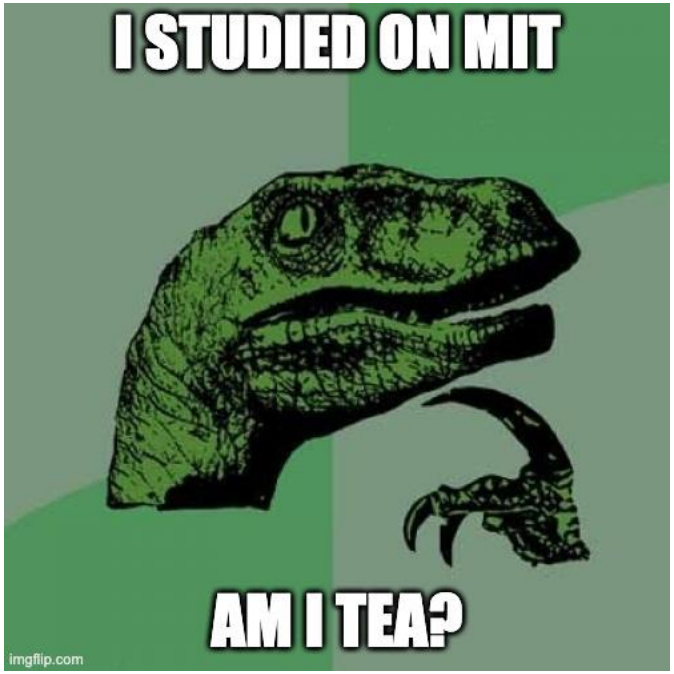
Baza danych

Client - SDK



Podstawowe info o temporalu

- Open source – licencja “Czy jestem herbatą?”



Podstawowe info o temporalu

- Open source – licencja MIT
- Napisany w Go
- Bazuje na event sourcingu, kolejkach, gRPC
- Self-hosted lub Cloud
- Rozwijany od 2009 (wcześniej jako Uber Cadence)
- 11 800 gwiazdek na githubie
- Solidne community (forum + slack)
- Ostatnie wydanie: 18. października

Workflow

- Kod
- Trwały (durable)
- Odporny na awarie (resilient)

Workflow - interfejs

```
@WorkflowInterface
public interface AccountTransferWorkflow {
  @WorkflowMethod
  String transfer(TransferRequestPOJO tr);
}
```

Workflow – implementacja

```
@Override
public String transfer(TransferRequestPOJO tr) {

  account.withdraw(tr.getFrom(), tr.getTransferId(), tr.getAmount());

  account.deposit(tr.getTo(), tr.getTransferId(), tr.getAmount());

  return "Successfully transferred money from: " + tr.getFrom() + " to " + tr.getTo();
}
```

Workflow - uruchomienie

```
var transferRequest = new TransferRequestPOJO("fromAccount", "toAccount", 1000);

var result = transferWorkflow.transfer(transferRequest);
```

Workflow - uruchomienie

```
var transferWorkflow =
        workflowClient.newWorkflowStub(AccountTransferWorkflow.class, options);

var transferRequest = new TransferRequestPOJO("fromAccount", "toAccount", 1000);

var result = transferWorkflow.transfer(transferRequest);
```

Workflow - uruchomienie

```
var workflowClient = WorkflowClient.newInstance(service);


var transferWorkflow =
        workflowClient.newWorkflowStub(AccountTransferWorkflow.class, options);

var transferRequest = new TransferRequestPOJO("fromAccount", "toAccount", 1000);

var result = transferWorkflow.transfer(transferRequest);
```

Workflow - uruchomienie

```
var service = WorkflowServiceStubs.newLocalServiceStubs();
var workflowClient = WorkflowClient.newInstance(service);


var transferWorkflow =
        workflowClient.newWorkflowStub(AccountTransferWorkflow.class, options);

var transferRequest = new TransferRequestPOJO("fromAccount", "toAccount", 1000);

var result = transferWorkflow.transfer(transferRequest);
```

Workflow - uruchomienie

```
var service = WorkflowServiceStubs.newLocalServiceStubs();
var workflowClient = WorkflowClient.newInstance(service);
var options = WorkflowOptions.newBuilder().setTaskQueue(TASK_QUEUE).build();

var transferWorkflow =
        workflowClient.newWorkflowStub(AccountTransferWorkflow.class, options);

var transferRequest = new TransferRequestPOJO("fromAccount", "toAccount", 1000);

var result = transferWorkflow.transfer(transferRequest);
```

Workflow - uruchomienie

```
var service = WorkflowServiceStubs.newLocalServiceStubs();
var workflowClient = WorkflowClient.newInstance(service);
var options = WorkflowOptions.newBuilder().setTaskQueue(TASK_QUEUE).build();

var transferWorkflow =
        workflowClient.newWorkflowStub(AccountTransferWorkflow.class, options);

var transferRequest = new TransferRequestPOJO("fromAccount", "toAccount", 1000);

var result = transferWorkflow.transfer(transferRequest);
```

Worker

- Obsługuje kod workflowu
- Pobiera zadania z przypisanej kolejki
- Po zmianie w kodzie zazwyczaj trzeba go restartować 

Workflow – wymagania

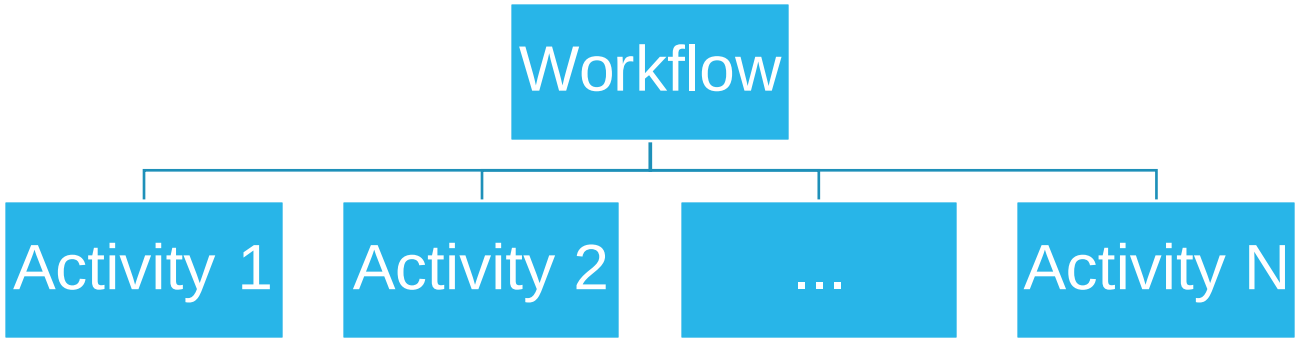
Nie wolno używać:

- Sleep
- Logger.log()
- Random
- System.currentTimeMillis()
- Interakcji z ”zewnętrznym światem”:
  - Operacje na plikach
  - Baza danych
  - Wywołanie serwisu zewnętrznego

Activity

- Kod
- (Może być) niedeterministyczna
- W razie awarii powtarzana

Workflow vs Activity



Workflow vs Activity

Workflow

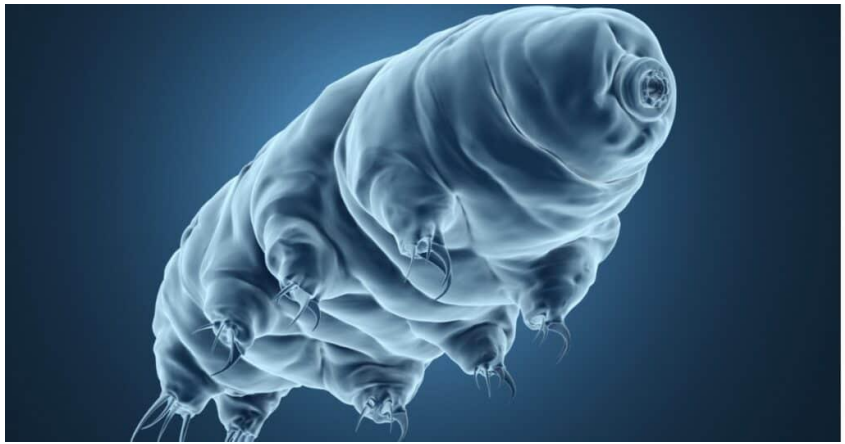
1. Musi być deterministyczny
2. Koordynuje operacjami (high lvl)
3. Odtwarzany (przy awarii)

Activity

1. Nie musi być deterministyczna
2. Wykonuje konkretną pojedynczą operację
3. Powtarzana (przy awarii)

Workflow vs Activity

Workflow



Activity

Workflow - odtwarzalność

```
public String transfer(TransferRequestPOJO tr) {
  accountActivities.withdraw(tr.getFrom(), tr.getTransferId(), tr.getAmount());
  accountActivities.deposit(tr.getTo(), tr.getTransferId(), tr.getAmount());

  return "Success";
}
```

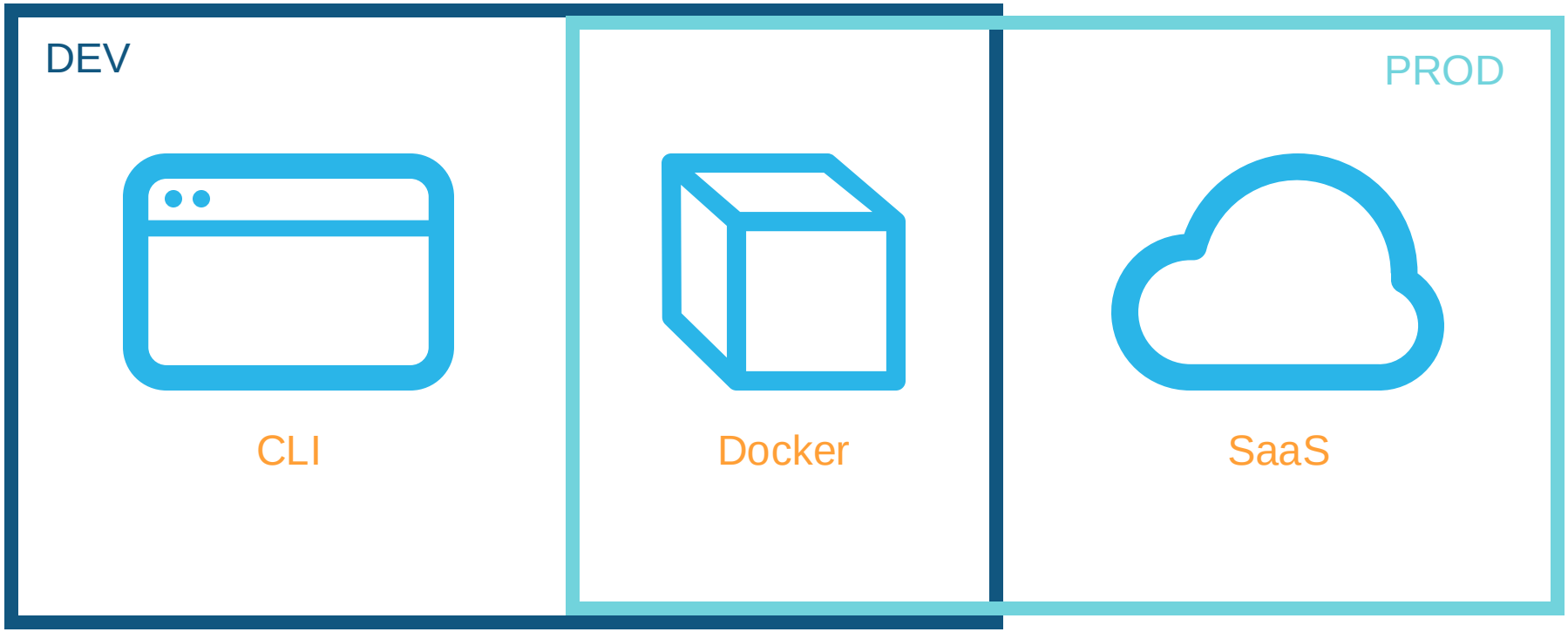
Workflow - odtwarzalność

```
public String transfer(TransferRequestPOJO tr) {
  accountActivities.withdraw(tr.getFrom(), tr.getTransferId(), tr.getAmount());
  accountActivities.deposit(tr.getTo(), tr.getTransferId(), tr.getAmount());

  return "Success";
}
```

Workflow - odtwarzalność

```
public String transfer(TransferRequestPOJO tr) {
  accountActivities.withdraw(tr.getFrom(), tr.getTransferId(), tr.getAmount());
  accountActivities.deposit(tr.getTo(), tr.getTransferId(), tr.getAmount());

  return "Success";
}
```

Jak uruchomić Temporal Cluster?



DEMO

Kto używa Temporal.io?

Do czego używamy?

Wady i zalety

Wady

- Dodatkowy element do utrzymania
- Stroma ścieżka nauki
- Trudniejsze debugowanie
- Trudniejsze testowanie
- Pilnowanie detereminizmu worfklowów
- Początkowa konfiguracja
- Brak GUI do projektowania workflowów

Zalety

- Podstawowa obsługa błędów out of the box
- Pomocny interfejs webowy
- Polyglot
- Zarządzanie skomplikowanymi procesami
- Niski narzut na wydajność
- Skalowalny

Gdzie ja jestem?

mbaranowski.pl

PS. CO POMINĄŁEM?

Jaki istotny aspekt pominąłem?

```
accountService.withdraw(fromAcc, id, amount)
accountService.deposit(toAcc, id, amount)
```

Jaki istotny aspekt pominąłem?

```
accountService.withdraw(fromAcc, id, amount)
accountService.deposit(toAcc, id, amount)
```

Jaki istotny aspekt pominąłem?

```
accountService.withdraw(fromAcc, id, amount)
try:
   accountService.deposit(toAcc, id, amount)
catch:
   accountService.deposit(fromAcc, id, amount)    // zwrot
```

Kompensacja

```
accountService.withdraw(fromAcc, id, amount)

saga.addCompensation(returnMoney, id)
try:
   accountService.deposit(toAcc, id, amount)
catch:
   saga.compensate()
```

Summary

- Problem przelewu
- Czym jest temporal.io
- Jaka jest architektura:
  - Workflow
  - Activity
  - Worker
- Jak używać temporal.io