---

co to jest GenStage?
Dokumentacja definiuje jak “silnik reguł obliczeniowych Elixir”
Oznacza to, że GenStage pozwala nam na zdefiniowanie potoków pracy, podzielonych na niezależne kroki (etapy), które będą obsługiwane przez różne procesy.

Mamy przykład, zwializualizowaśmy potok danych producent - konsument:
Producent dostarcza jakieś dane, dalej te dane przekształca producenci-konsumenci i na końcu przekształcone dane są dostarczane do konsumenta

Rola wyznaczona dla każdego etapu jest bardzo ważna. Specyfikacja GenStage wyznacza 3 roli:

- producer - źródło danych.Producenta czeka na wymóg od konsumentów i reaguje wykonując wymagane działania
- producer-consumer - i źródło i konsument. Może jak reagować na wymogę konsumenta tak i żądać działania od producenta
- consumer - konsument. Żąda i otrzymuje dane od producenta

Co znaczy, że producenci oczekują na żądania? Konsumenci GenStage wysyłają strumień żądań i następnie przetwarzają dane otrzymane od producentów. Mechanizm ten znany jest jako ciśnienie wsteczne (ang. back-pressure). Ciśnienie wsteczne przenosi ciężar na producentów, chroniąc aplikację przed przeciążeniem, gdy konsumenci są zajęci.

Nasza przykładowa aplikacja używająca GenStage będzie generować strumień liczb, sortować numery parzyste i w końcu wypisywać je.

Użyjemy wszystkich ról GenStage. Producent będzie odliczać i emitować kolejne numery. Użyjemy producenta-konsumenta do odfiltrowania numerów parzystych oraz późniejszego ich odesłania do konsumenta, który będzie wyświetlać je na ekranie.

- Naszą pracę zaczniemy od stworzenia projektu z wykorzystaniem drzewa nadzorców:
- Następnie zmieńmy **mix.exs** dodając zależność do **gen_stage**:
- Zanim przejdziemy dalej musimy pobrać i skompilować zależności:

- tworzenie producenta. Stworzymy zatem plik modułu:
- W funkcji **init/1** ustawiamy stan początkowy, określamy moduł jako producenta Odpowiedź z init/1 pozwala GenStage na klasyfikację procesu
- W funkcji **handle_demand**/2 jest centrum logiki producenta i musi być zaimplementowana we wszystkich producentach GenStage. To tutaj zwracamy zbiór numerów żądanych przez konsumentów i zwiększamy licznik. Żądanie ze strony konsumentów, parametr demand, odpowiada liczbie całkowitej określającej maksymalną ilość zdarzeń, której mogą oni podołać.

- Teraz możemy zająć się producentem-konsumentem. Będzie on żądać od producenta zbioru liczb, wybierać tylko parzyste i odsyłać tak odfiltrowany zbiór na żądanie konsumenta.
- Jak łatwo zauważyć w funkcji **init**/1 dodaliśmy nową opcję opcja **subscribe_to** pomaga nam w komunikacji z określonym producentem.
- Funkcja **handle_events**/3 obsługuje przychodzące odpowiedzi i transformuje zbiór danych.
  Jak widać, konsumenci są implementowani w podobny sposób, ale ważną różnicą jest to, co zwraca funkcja handle_events/3 i to, jak jest używana.

- Jako że producent-konsument i konsument są bardzo podobne, to kod nie będzie się różnił w znaczący sposób
- Jeżeli oznaczymy nasz proces jako producenta-konsumenta, drugi argument w krotce, tu numbers, zostanie wykorzystany do odpowiadania na żądania. W przypadku konsumentów będzie on zignorowany.

- Potoki transformacji danych — producenci nie są tu prostymi generatorami liczb. Możemy generować dane, wykorzystując bazy danych albo rozwiązania w rodzaju Apache Kafka. Łącząc wielu producentów-konsumentów i konsumentów możemy przetwarzać, sortować, katalogować różne dane.
- Kolejki — zdarzenia mogą być różnorodne, a naszym zadaniem jest ich obsługa za pomocą serii konsumentów.
- Obsługa zdarzeń — zbliżona do obsługi danych, lecz tym razem przetwarzamy, sortujemy i obsługujemy zdarzenia generowane w czasie rzeczywistym.