Minusem natomiast to, że Mnesia posiada dosyć zawiłą bazę danych i nie jest używana poza społecznością Elixira/Erlanga. Oznacza to, że trudniej o wsparcie dla narzędzi, a społeczność jest mniejsza, w porównaniu do popularnych rozwiązań typu DBMS. Sprawienie, żeby Mnesia działała na większą skalę, również nie jest łatwe. Jeden z problemów np. polega na tym, że tabele znajdujące się na dysku nie mogą przekroczyć 4 GB (jest to ograniczenie leżące u podstaw przechowywania terminów przez DETS), co oznacza, że większe tabele wymagają fragmentacji.

### 10.3.4. Ćwiczenie: rejestr procesów

To dobry moment na ćwiczenia. Rejestr procesów jest książkowym przykładem praktycznego zastosowania ETS. Moduł `Registry` stosuje inteligentne połączenie `GenServer` i ETS dla osiągnięcia maksymalnej wydajności. W tym ćwiczeniu zaimplementujesz podstawową wersję rejestru `:unique`.

Oto przykład stosowania takiego rejestru:

```
iex(1)> SimpleRegistry.start_link()
{:ok, #PID<0.89.0>}

iex(2)> SimpleRegistry.register(:some_name)
:ok

iex(3)> SimpleRegistry.register(:some_name)
:error

iex(4)> SimpleRegistry.whereis(:some_name)
#PID<0.87.0>

iex(5)> SimpleRegistry.whereis(:unregistered_name)
nil
```

**Pomyślna rejestracja**

**Błąd podczas duplikacji rejestracji**

**Udane wyszukiwanie**

**Nieudane wyszukiwanie**

Interfejs `SimpleRegistry` jest interfejsem podstawowym. Serwer jest lokalnie uruchamiany i rejestrowany. Od tego momentu proces może zarejestrować się poprzez wywołanie `SimpleRegistry.reg- ister/1`, przekazując dowolny termin jako klucz procesu. Funkcja ta zwraca `:ok`, jeżeli rejestracja się powiedzie, lub `:error`, jeżeli nazwa jest już zajęta. Wyszukanie jest wykonywane za pomocą wywołania `SimpleRegistry.whereis/1`, które zwraca pid danego klucza lub `nil`, jeżeli proces nie jest zarejestrowany pod daną nazwą.

Dodatkowo rejestr procesów może wykryć zakończenie każdego zarejestrowanego procesu i usunąć wszystkie wpisy jego rejestracji.

`SimpleRegistry` nie posiada wymyślnych funkcji modułu `Registry`, takich jak wsparcie poprzez krotki, podwójna rejestracja czy wiele instancji rejestracji.

Oto, jak możesz stworzyć taki rejestr:

1. Zaimplementuj pierwszą wersję `SimpleRegistry` jako `GenServer`. Zarówno `register`, jak i `whereis` zostaną zarejestrowane jako wywołania.
2. Stanem `GenServer` powinna być mapa, której kluczami są zarejestrowane nazwy, a wartościami pid.
3. Obsługując wywołanie `:register`, proces rejestru powinien łączyć się z wywołującym, aby móc wykryć zakończenie procesu i wyrejestrować go. Serwer rejestru