Lab Test

UAIM

Marcin Dadura
Maciej Włodarczyk

Politechnika Warszawska

9 kwietnia 2021

Spis treści

1. Wstęp

Celem laboratorium było rozbudowanie zaimplementowanej przez nas wcześniej mikrousługi przypisywania lekarzy do gabinetów o testy jednostkowe weryfikujące poprawność implementacji metody przydziału.

2. Użyte Technologie

Mikrousługa napisana została w języku C i wykorzystuje framework .NET 5. Do testowania użyliśmy narzędzia **xUnit**[1], które jest darmowym open-source narzędziem do testowania frameworka .Net. Uzyliśmy również rozszerzenia **Fluent Assertion**[2], co pozwoliło tworzyć asercje w przyjemniejszy i łatwiejszy sposób.

3. Metody testujące mikrousługę aplikacyjną

Testowana przez nas metoda przyjmuje 2 listy: List<DoctorDto> oraz List<ExaminationRoomDto>.Po przejściu algorytmu metoda zwraca listę obiektów DoctorRoomDto, który to jest obiektem agregującym DoctroDto i ExaminationRoomDto.

Zgodnie z wytyczbymi podanymi w instrukcji stworzyliśmy metody testujące odpowiednie aspekty działania funkcji przypisującej gabinety do lekarzy.

3.1. Sprawdzania braku/niepoprawnych danych zwracanych przez usługi danych

W celu sprawdzenia zachowania się metody stworzyliśmy 4 testy, do których przekazujemy argumenty uzywając stworzonej przez nas klasy **DataGenerator** dziedziczącej po **IEnumerable<object[]>**. Argumenty przekazywane są poprzez specjalną adnotację ([MemberData]) dostępną w narzędziu xUnit.

Metoda *ShouldThrowArgumentNullExceptionWhenPassingNullArguments*

Jest to metoda mająca na celu sprawdzenie, czy aplikacja wzniesie wyjątek *ArgumentNullException* po przekazaniu nieodpowiednich danych w postaci:

— null - null
— List<DoctorDto> - null
— null - List<ExaminationRoomDto>

Metoda *ShouldThrowArgumentNullExceptionWhenPassingNullExaminationRooms*

Jest to metoda mająca na celu sprawdzenie, czy aplikacja wzniesie odpowiedni wyjątek po przekazaniu danych w postaci : List<DoctorDto> - null. Wzniesiony wyjątek powinien być *ArgumentNullException* i zawierać w sobie stringa "examinationRoomsDto".

Metoda *ShouldThrowArgumentNullExceptionWhenPassingNullDoctors*

Jest to metoda mająca na celu sprawdzenie, czy aplikacja wzniesie odpowiedni wyjątek po przekazaniu danych w postaci : null - List<ExaminationRoomDto>. Wzniesiony wyjątek powinien być *ArgumentNullException* i zawierać w sobie stringa "doctorsDto".

Metoda *ShouldReturnEmptyListWhenPassingEmptyDoctorsAndEmptyRooms*

Jest to metoda mająca na celu sprawdzenie, czy aplikacja zwróci zerową ilość par po przekazaniu argumentów w postaci : List<DoctorDto> - List<ExaminationRoomDto>, w których to nie będą znajdować się żadne elementy - listy będą puste.

3.2. Sprawdzania poprawności wyników dla trzech zestawów przykładowych danych

W celu sprawdzenia zachowania się metody stworzyliśmy test, do którego przekazujemy argumenty w identyczny sposób, jak w poprzednim punkcie. W tym przypadku przekazujemy jednak 3 zestawy danych.

Metoda *ShouldMatchDoctorWithExaminationRoomWhenPassingOneDoctorAndOneExaminationRoom*

Jest to metoda mająca na celu sprawdzenie, czy metoda zwraca dokładnie jedną parę po przekazaniu argumentów w postaci : List<DoctorDto> - List<ExaminationRoomDto>, w kóryc to znajduje sie po jednym elemencie.

3.3. Sprawdzanie sensowności wyników dla trzech zestawów przykłądowych danych

W celu sprawdzenia zachowania się metody stworzyliśmy test, do którego przekazujemy argumenty w identyczny sposób, jak w poprzednich punktach. Jednakże tym przypadku przekazujemy również liczby całkowitej, będąca oczekiwanym przez nas rezultatem. Tutaj również przekazujemy 3 zestawy danych.

Metoda *ShouldMatchAllDoctorsWithExaminationRoomsAndCountShouldEqualsTheSpecifiedNumber*

Jest to metoda mająca na celu sprawdzenie, czy metoda zwraca zdeklarowaną przez nas ilość par po przekazaniu argumentów w postaci : List<DoctorDto> - List<ExaminationRoomDto> - int.

3.4. Sprawdzanie czasu działania metpdy dla duzego zestawu danych

W celu sprawdzenia zachowania się metody stworzyliśmy test, do którego przekazujemy argumenty z pliku Json. Uzywamy do tego klasy *JsonFileDataAttribute*, co pozwoliło nam uzyć adnotacji [JsonFileData]. Zdecydowaliśmy się na przekazywanie argumentów w ten sposób ze względu na prostotę implementacji oraz łatwość generowania danych w postaci Json.

Metoda *ShouldMatchManyDoctorsWithManyExaminationRoomsInRelevantTime*

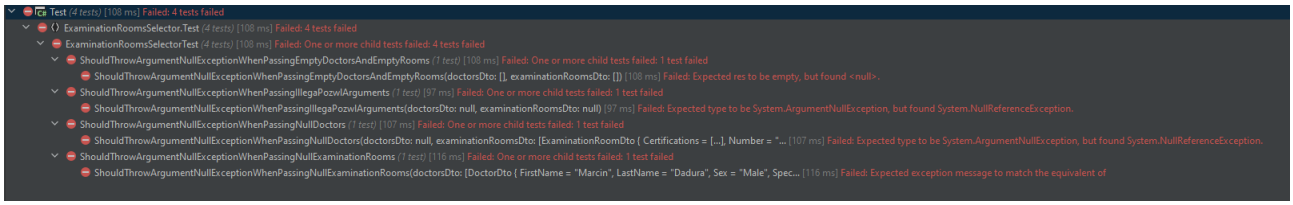
Jest to metoda mająca na celu sprawdzenie, czy algorytm w metodzie którą testujemy jest optymalny. Dla duzych danych, w postaci 10 tyś. Doctors oraz 10 tyś ExaminationRooms sprawdzamy, czy w którym wykona się metoda jest mniejszy niz 30 sekund.

4. Testy zakończone niepowodzeniem

Poniżej znajdują się screeny z porażką dla każdego z zestawów testów.

Test braku/niepoprawnych danych zwracanych przez usługi danych

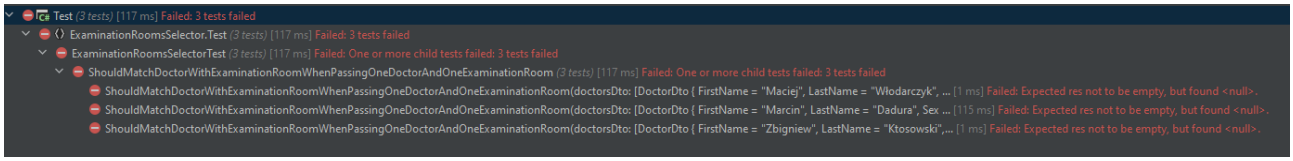
Widać, że testy który mają rzucić wyjątek niepowodzą się. Test przyjmujący puste listy również nie powodzi się (Rys. 1.).



Rys. 1. Sprawdzania braku/niepoprawnych danych zwracanych przez usługi danych

Test poprawności wyników dla trzech zestawów przykładowych danych

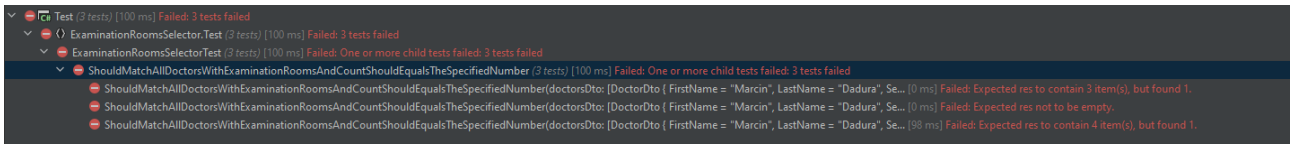
Widać, że testy nie powiosły się. Spowodowane jest to faktem, że zwraca wartość przez funkcję to null, zamiast spodziewanej 1 (Rys. 2.).



Rys. 2. Sprawdzania poprawności wyników dla trzech zestawów przykładowych danych

Test sensowności wyników dla trzech zestawów przykłądowych danych

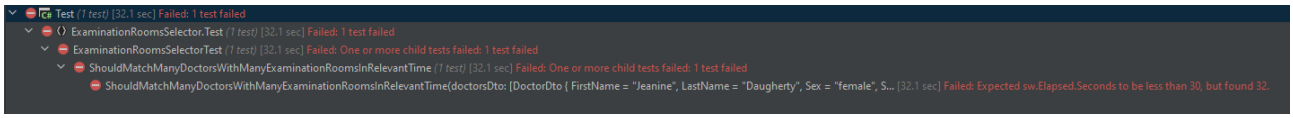
Widać, że testy nie powiosły się. Powodem jest błędna ilość elementów w zwracanej liście (Rys. 3.).



Rys. 3. Sprawdzanie sensowności wyników dla trzech zestawów przykłądowych danych

Test czasu działania metpdy dla duzego zestawu danych

Widać, że testy nie powiosły się. Powodem jest zbyt długi czas działania algorytmu (Rys. 4.).



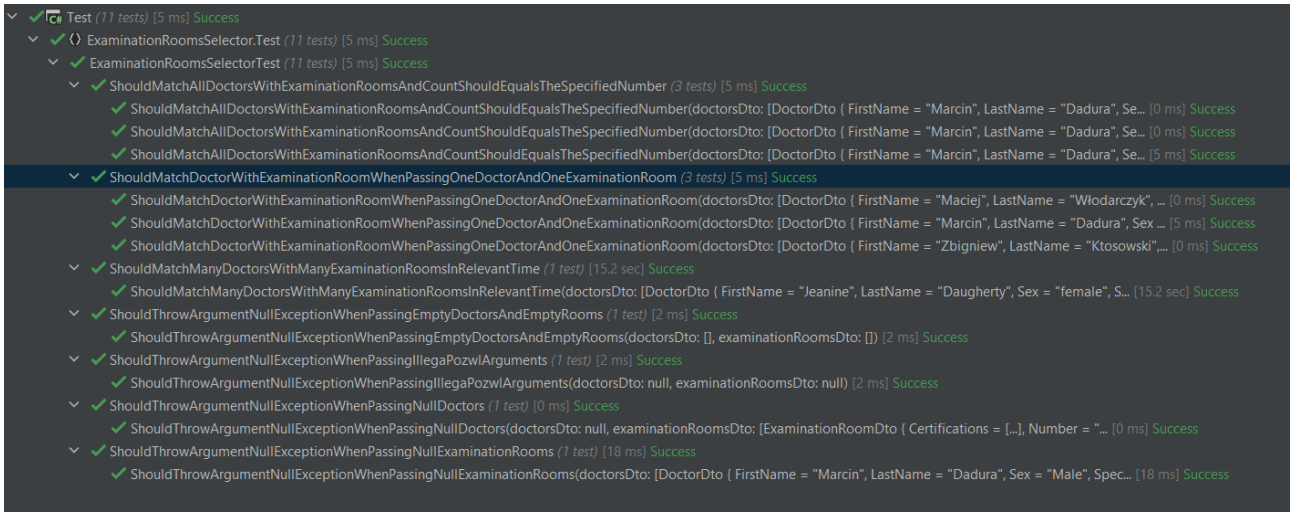
Rys. 4. Sprawdzanie czasu działania metpdy dla duzego zestawu danych

5. Testy zakończone sukcesem

Aby móc osiądnąc cel, czyli aby wsyztskie testy zakończyły się pomyślnie musieliśmy przebudować metodę dobierającą pary lekarz-pokoj. Usprawnienia wprowadzone przez nas:

— sprawdzanie przekazywanych argumentów do funkcji, jeżeli dane sa niepoprawne wznoszony jest wyjątek
— naprawilismy błąd, przez który zwracana była błędna lista List<DoctorRoomDto>
— po naprawie dziury w metodzie zoptymalizowaliśmy kod np. zamiast używania metody *Intersect* iterujemy się po listach uzywając foreach.

Po usprawnieniach wsyztskie meotdy przechodzą pomyślnie (Rys. 5.).



Rys. 5. Poprawnie zakończone wszytskie testy jednostkowe

6. Wnioski

Dzięki laboratoriom zapoznaliśmy się z testami jednostkowymi, uzywając przy tym *xUnit* oraz *Fluent Assertions*. Pisanie testów jest proste i pozwala sprawdzić aplikację na potencjalne błędy.

Literatura

[1] xUnit, repozytorium GitHub `https://github.com/xunit/xunit`
[2] Fluent Assertions, repozytorium GitHub `https://github.com/fluentassertions/fluentassertions`