AKADEMIA HANDLOWA NAUK STOSOWANYCH W RADOMIU

RADOM ACADEMY OF ECONOMICS

INSTRUKCJA LABORATORYJNA NUMER 1	
Przedmiot: Zagrożenia bezpieczeństwa systemów	Tok: 2024/2025
Cel praktyczny zajęć: Usunięcie podatności rozpoznanej w aplikacji w ramach testów penetracyjnych.	

1. Scenariusz ćwiczenia

Podczas przeprowadzania testów penetracyjnych w aplikacji WeatherChecker wykryto podatności, poniżej znajdziesz fragment raportu.

Raport z testów bezpieczeństwa

1. Informacje ogólne

- **Data testu:** 20 marca 2025
- **Tester:** Brajan Testujący
- **System testowany:** WeatherChecker API
- **Metoda testowania:** Testy manualne i automatyczne (np. OWASP ZAP, Postman, Burp Suite)

2. Opis wykrytego problemu

- **Identyfikator:** SEC-API-001
- **Typ zagrożenia:** Brak uwierzytelniania JWT w API
- **Opis:** Testy wykazały, że API serwisu WeatherChecker nie wymaga tokena JWT do autoryzacji dostępu do wrażliwych endpointów. Brak takiej ochrony umożliwia nieautoryzowany dostęp do danych.
- **Poziom ryzyka:** Wysoki

3. Kroki do reprodukcji błędu

1. Wysłanie zapytania HTTP do endpointu https://localhost:7160/WeatherForecast bez nagłówka Authorization: Bearer <JWT_TOKEN>.
2. API zwraca poprawne dane bez sprawdzania tokena.
3. Powtórzenie testu dla innych endpointów ujawnia ten sam problem.

4. Skutki potencjalnego ataku

- Możliwość nieautoryzowanego dostępu do zasobów aplikacji.
- Ryzyko wycieku danych wrażliwych.
- Możliwość manipulacji danymi przez nieautoryzowane podmioty.

5. Zalecane działania naprawcze

- Wymuszenie autoryzacji dla wszystkich endpointów, które obsługują dane użytkowników oraz zasoby aplikacji.
- Implementacja weryfikacji nagłówka Authorization i walidacji tokena JWT.
- Użycie middleware do globalnej obsługi uwierzytelniania.
- Regularne testy penetracyjne w celu wykrycia podobnych luk w przyszłości.

6. Podsumowanie

Luka w API serwisu WeatherForecast pozwala na nieautoryzowany dostęp do danych aplikacji. Zaleca się pilne wdrożenie tokenów JWT i ich weryfikację w celu zabezpieczenia API przed nieautoryzowanymi żądaniami.

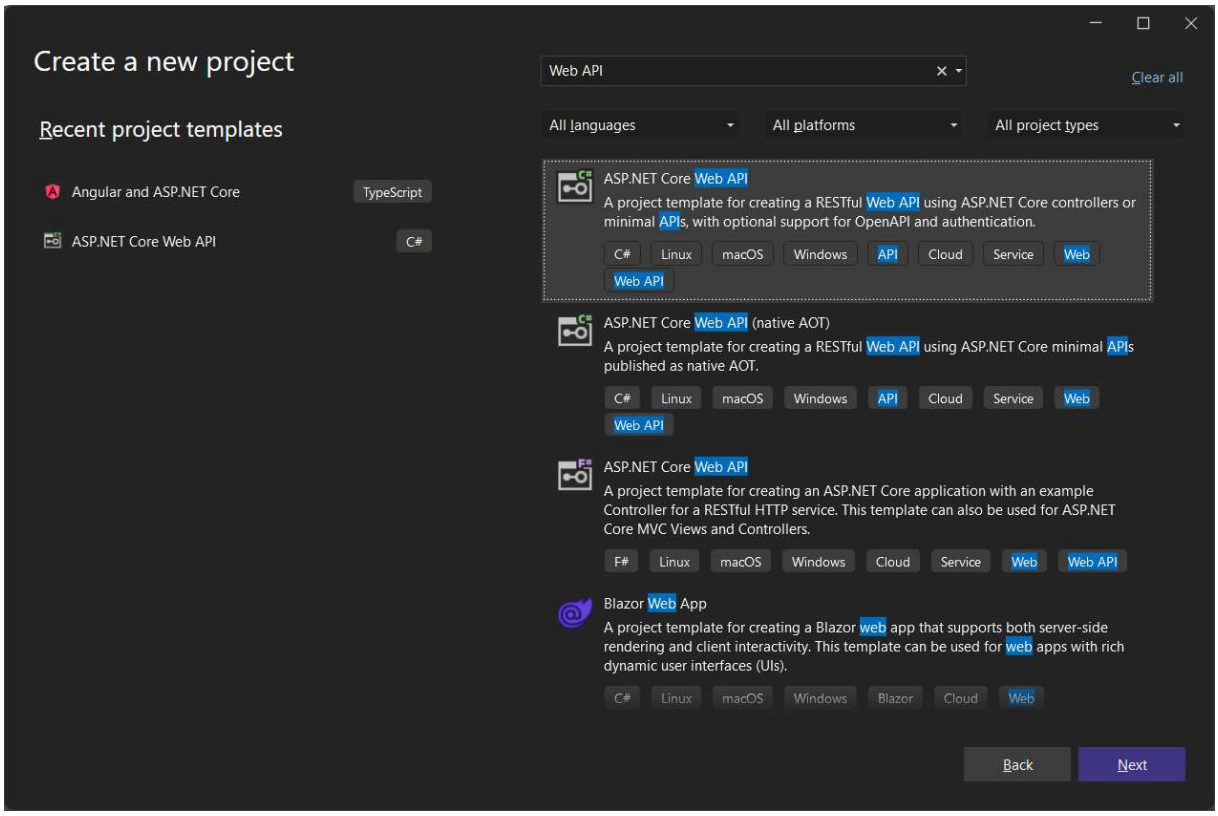
Aplikacja WeatherChecker jest przykładową aplikacją stworzoną w technologii .NET. W ramach zajęć laboratoryjnych Twoim zadaniem jest odtworzenie błędnego działania aplikacji, a następnie usunięcie podatności z powyższego raportu. Realizacja omawianego ćwiczenia jest dowodem potwierdzającym czynny udział w zajęciach. Wykonanie ćwiczenia jest obowiązkowe, podlega zaliczeniu i ocenie, każda z poszczególnych instrukcji laboratoryjnych jest obligatoryjna i warunkuje pozytywną ocenę z przedmiotu. Rezultat prac projektowych/technicznych powinien być umieszczony na platformie github, a odpowiedzi na dodatkowe pytania z instrukcji przesłane w formie tekstowej w pliku pdf pod nazwą: imię_nazwisko_numer_grupy.pdf. Wspominany plik oraz link do projektu na platformie github (należy zadbać o ustawienie publicznego repozytorium) proszę przesłać na adres mailowy: bkicior@ahns.pl, najpóźniej do dnia poprzedzającego termin kolejnych zajęć, w tym przypadku będzie to: 05.04.2024.

2. Utworzenie projektu

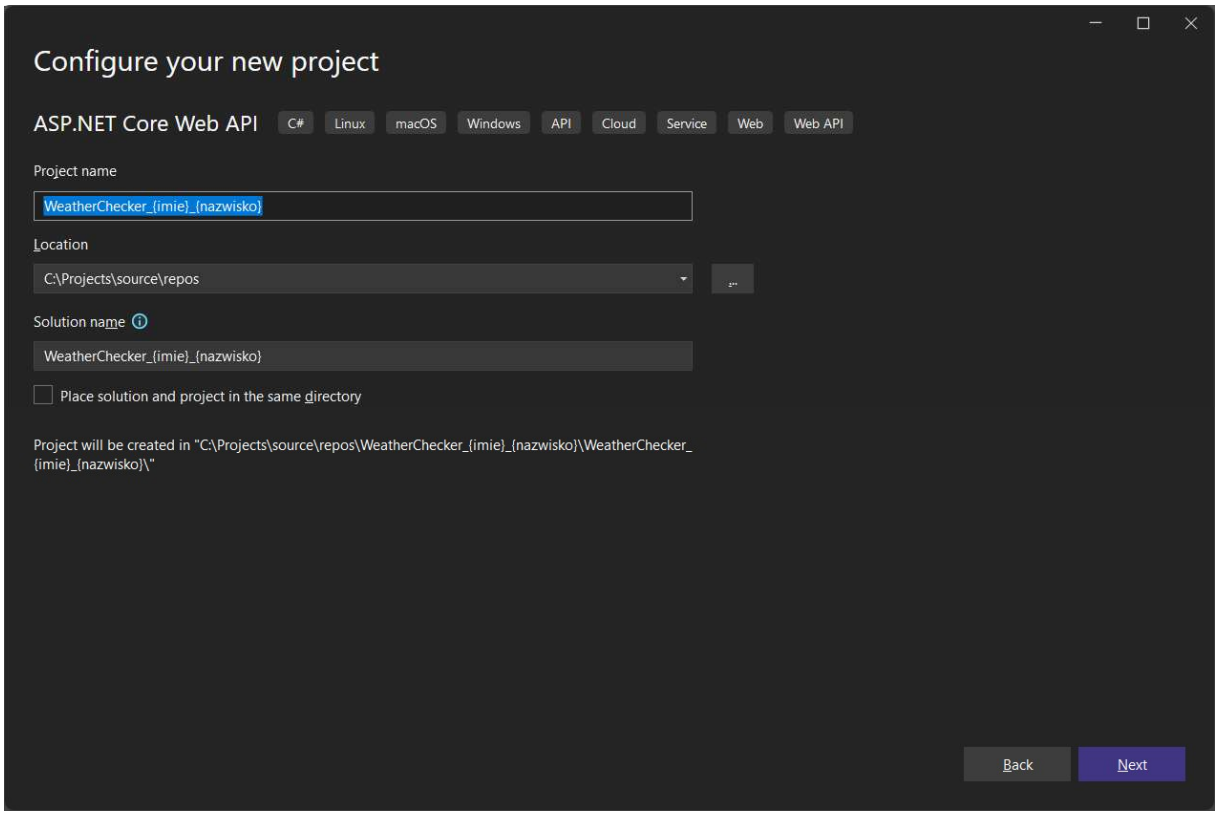
Otwórz Visual Studio lub Visual Studio Code, a następnie utwórz projekt o nazwie WeatherChecker_TwojeInicjały.

Możesz to zrobić za pomocą komendy:
dotnet new webapi -n WeatherChecker_TwojeInicjały --use-controllers

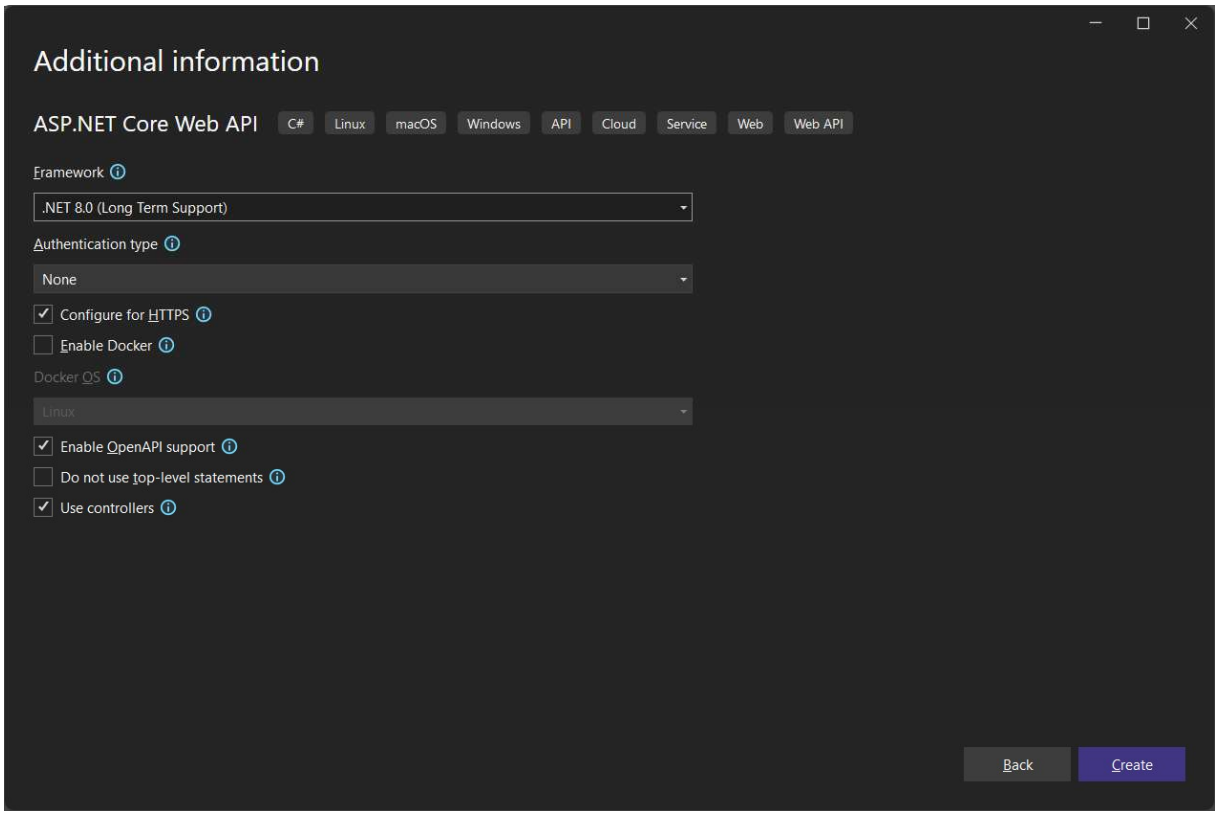
Lub korzystając z GUI:



Wyszukaj szablon projektu o nazwie ASP.NET Core Web API

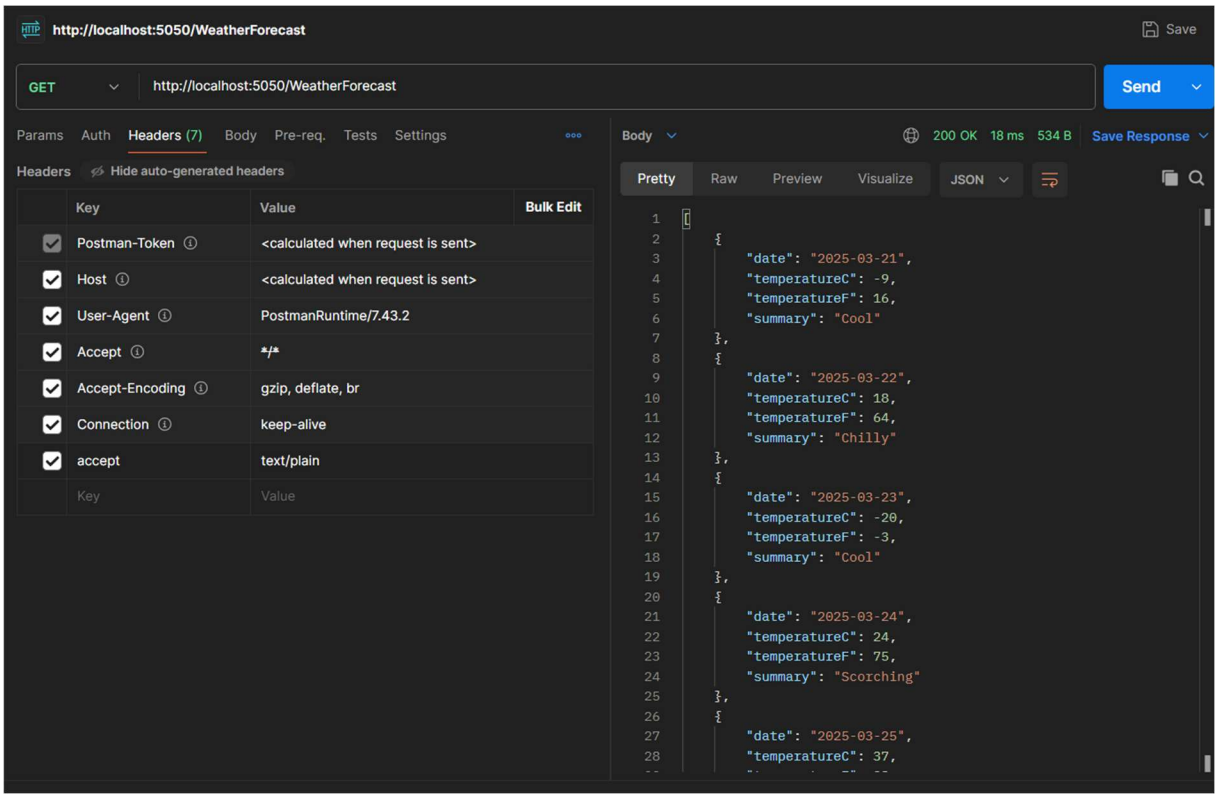


Nadaj nazwę projektu: WeatherChecker_TwojeInicjały



Pamiętaj o zaznaczeniu checkboxa Use Controllers.

Uruchom projekt i sprawdź za pomocą swaggera, przeglądarki lub postmana czy endpoint: https://localhost:{twoj_port_pod_ktorym_udostepniona_jest_aplikacja}/WeatherForecast zwraca dane bez podania tokenu JWT. API powinno działać poprawnie i zwrócić wygenerowane losowo dane.

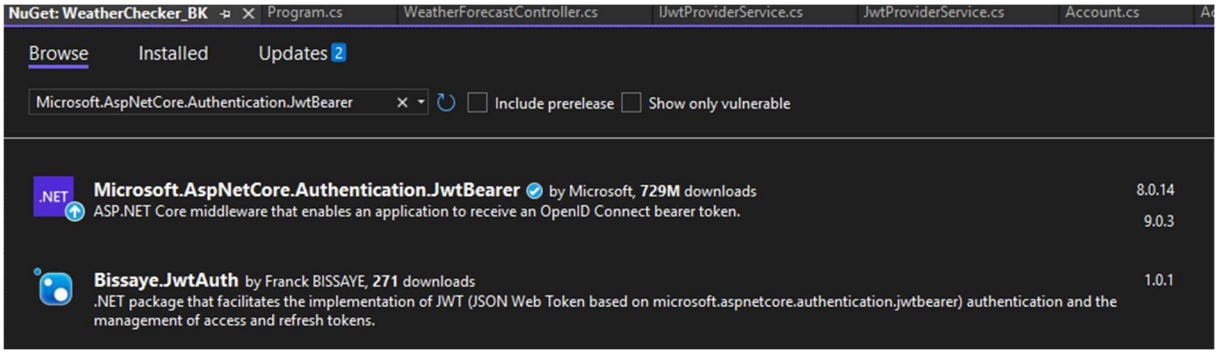


Pytania dodatkowe:

Sprawdź czym są tokeny JWT, jaka jest ich rola w aplikacji, do czego można je wykorzystać. Znajdź lub wygeneruj i opisz przykładowy token, z czego się składa oraz co można w nim zapisać? Sprawdź jakie paczki nuget można wykorzystać do generowania tokenów w aplikacji .NET, wypisz te rozwiązania i opisz czym się charakteryzują. Odpowiedzi zapisz w dokumencie, z którego później wygenerujesz plik pdf, zapisz informacje do, którego z punktów ćwiczenia się odnoszą.

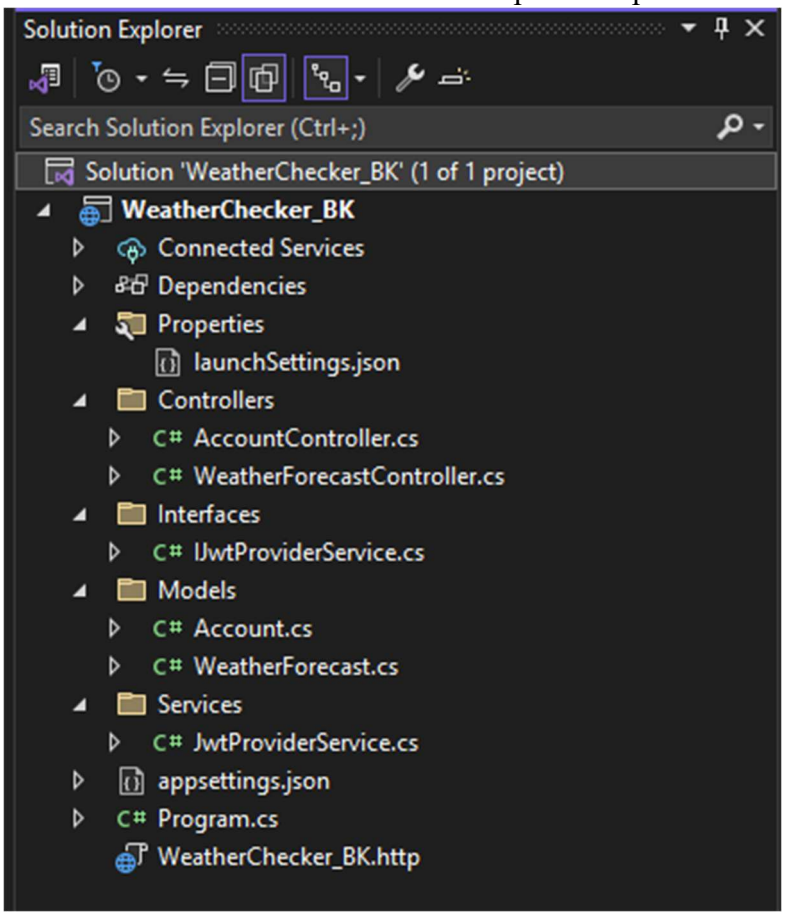
3. Implementacja tokenów

W aplikacji WeatcherChecker Zainstaluj paczkę nuget o nazwie: Microsoft.AspNetCore.Authentication.JwtBearer



Ćwiczenie zawiera wiele uproszczeń, celem jest stworzenie metody API, która przyjmie od użytkownika dane do logowania (email oraz hasło), a w kolejnym kroku wygeneruje dla niego indywidualny token, który następnie posłuży do autoryzacji w istniejącej już metodzie generującej prognozę pogody. W pierwszej wersji aplikacji wprowadzone przez użytkownika (email i hasło) nie będą w żaden sposób walidowane i weryfikowane, token będzie generowany dla każdego wywołania.

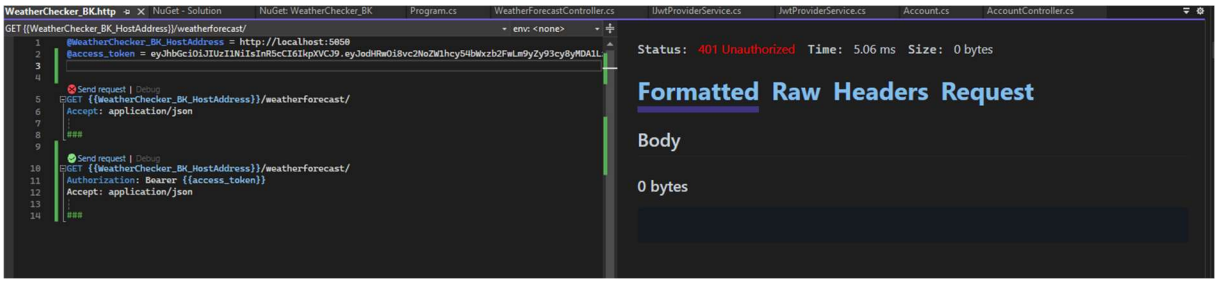
Dodaj nowy API kontroler o nazwie: AccountController, zaimplementuj w nim metodę służącą do logowania użytkownika, zastanów się nad odpowiednim typem metody (GET/POST/PUT/DELETE). Metoda powinna zwracać token jako string oraz przyjmować obiekt użytkownika, w tym celu stwórz klasę Account, która zawierać będzie dwie propercje: Email oraz Passowrd. Właściwy token będzie generowany w serwisie o nazwie JwtTokenService, utwórz go, a także dedykowany dla niego interfejs, który zarejestrujesz również w kontenerze. Struktura plików powinna być zbliżona do następującej:



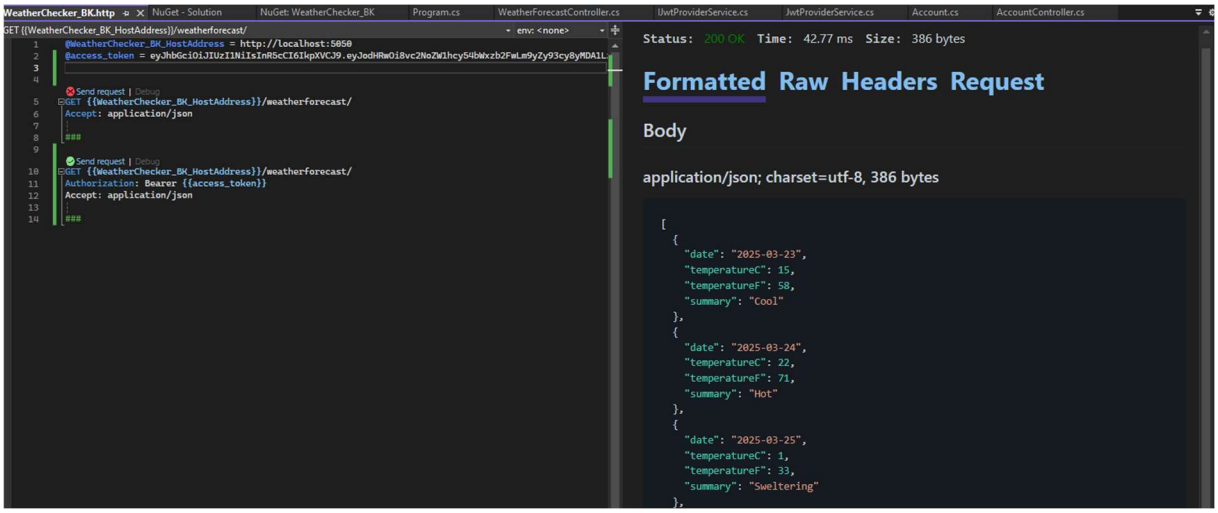
Zaimplementuj logikę generowania tokenów w JwtProviderService, która do claimów tokenu doda email użytkownika. Następnie wstrzyknij serwis do kontrolera i zwróć token w metodzie do logowania. Nałóż na kontroler WeatherForecastController atrybut, który wymusi autoryzację, a następnie przetestuj działanie aplikacji. Dokumentację do paczki nuget znajdziesz pod linkiem:

https://learn.microsoft.com/en-us/aspnet/core/security/authentication/configure-jwt-bearer-authentication?view=aspnetcore-9.0

Przykładowy test, przedstawiający oczekiwane działanie aplikacji:



Wywołanie metody http://localhost:5050/WeatherForecast bez tokena.



Wywołanie metody http://localhost:5050/WeatherForecast z tokenem.

Zastanów się i zapisz jakie jeszcze informacje można zawrzeć w tokenie JWT, sprawdź i opisz jakie algorytmy bezpieczeństwa można zastosować do kodowania credentiali (klasa: Microsoft.IdentityModel.Tokens.SecurityAlgorithms), wypisz trzy oraz zdefiniuj różnice pomiędzy nimi.

4. *Weryfikacja danych logowania użytkowników

Na ten moment aplikacja nie weryfikuje danych do logowania użytkowników, należy to poprawić, w tym celu:

1. Utwórz i zaimplementuj bazę danych (In memory lub MSSQL), a w niej tabelę do przechowywania użytkowników o nazwie Accounts zawierającą odpowiednie kolumny.
2. Utwórz serwis oraz metodę w kontrolerze do rejestrowania użytkowników, zadbaj o sprawdzenie czy użytkownik z danym emailem już istnieje. Rezultatem działania powinno być dodanie użytkownika do tabeli Accounts.
3. W metodzie do logowania użytkowników przed wygenerowaniem tokenu porównaj hasło przesłane przez użytkownika z tym zapisanym w bazie danych.
4. Hasło w bazie danych nie może być przechowywane w sposób jawny dodaj hashowanie hasła przy rejestracji oraz porównanie hashy przy logowaniu.

Po zakończonej pracy nie zapomnij przesłać projektu oraz odpowiedzi!