Bezpieczeństwo Aplikacji Webowych – BAW_CHAT – Raport

Patryk Mróz 269882
Kajetan Kuszczyński 246644

Cel projektu

Celem projektu było przygotowanie aplikacji czatu rzeczywistego, implementującego protokół wymiany klucza Diffiego-Hellmana i szyfrującego komunikację end-to-end z użyciem tego klucza.

Opis aplikacji

Aplikacja czatu, korzystająca z protokołu wymiany klucza Diffiego-Hellmana, mająca na celu zapewnienie bezpiecznej komunikacji. Użytkownicy mają możliwość tworzenia kont, logowania się oraz nawiązywania rozmów z innymi użytkownikami, wpisując ich nazwy użytkownika.

Aplikacja zapewnia opcje szyfrowania wiadomości, w tym możliwość wyboru między szyfrowaniem XOR a szyfrem Cezara. Domyślnie, opcja szyfrowania jest ustawiona na brak, czyli żadne szyfrowanie nie jest stosowane.

Główne funkcje aplikacji:

- Rejestracja i logowanie: Użytkownicy mogą tworzyć konta i logować się do aplikacji.
- Wybór rozmówcy: Użytkownicy mogą wybrać z kim chcą rozmawiać, wpisując nazwę użytkownika rozmówcy.
- Bezpieczna komunikacja: Aplikacja wykorzystuje protokół Diffiego-Hellmana do wymiany kluczy i zapewnienia bezpiecznej komunikacji między użytkownikami.
- Opcje szyfrowania: Użytkownicy mają możliwość wyboru sposobu szyfrowania wiadomości, takich jak XOR lub szyfr Cezara.

Wykorzystane technologie

Zdecydowano o wykorzystaniu otwartoźródłowego, crossplatformowego rozwiązania ASP.NET wraz z Blazor Pages do wytworzenia frontendu oraz backendu serwera, pełniącego rolę swoistego NAT-Punchthrough. Do komunikacji między użytkownikami czatu w czasie rzeczywistym wykorzystano bibliotekę SignalR, także otwartoźródłową i crossplatformową. Do przechowywania danych o użytkownikach wykorzystano bazę SQLite.

Technologie zostały wybrane ze względu na znajomość ich przez członków zespołu oraz chęć rozszerzenia wiedzy w ich kierunku. Są one także rozwiązaniami dojrzałymi, obecnymi na rynku od wielu lat.

Przebieg pracy

1. Prace rozpoczęto od utworzenia nowego projektu z szablonu ASP.NET wykorzystującego bibliotekę SignalR do realizacji funkcjonalności prostego czatu czasu rzeczywistego.
2. Zedytowano wygenerowany kod, aby implementował protokół Diffiego-Hellmana.
3. Dodano możliwość rejestrowania użytkowników i logowania się na konto użytkownika.
4. Dodano funkcje do szyfrowania i odszyfrowania oraz enkodowania i dekodowania wiadomości.
5. Ulepszono interfejs frontendu, aby odwzorować zmiany na backendzie.
6. Napisano końcową logikę czatu.
7. Wydzielono funkcje odpowiedzialne za szyfrowania i kodowanie do osobnego pliku.
8. Przygotowano testy automatyczne sprawdzające działanie funkcji szyfrowania i kodowania.

Napotkane problemy

Podczas prac nad projektem nie natrafiono na większe problemy. Dzięki zastosowaniu dobrze znanych członkom zespołu technologii, wszystkie problemy były rozwiązywane w ciągu maksymalnie kilku godzin. Zespół nigdy nie natrafił na tzw. *blockera*, który wstrzymałby cały proces developmentu.

Przykładowy problem, który wystąpił w trakcie prac nad projektem. Podczas implementacji funkcji kodowania w Base64, pojawiły się trudności z wysyłaniem wiadomości zawierających polskie znaki. Napotkano błąd, który wynikał z faktu, że funkcja btoa w języku JavaScript została zaprojektowana do obsługi znaków z zakresu Latin1 (ISO-8859-1). Polskie znaki, takie jak "ż, ą, ł", znajdują się poza tym zakresem i nie mogą być bezpośrednio zakodowane za pomocą funkcji btoa. Aby rozwiązać ten problem, skorzystano z API TextEncoder i TextDecoder, które oferują bardziej zaawansowane możliwości kodowania i dekodowania znaków.

Możliwości rozwoju

- Obecnie interfejs aplikacji jest w pełni funkcjonalny, jednakże można by go było trochę ulepszyć. Przede wszystkim należałoby go upodobnić do istniejących już komunikatorów internetowych, takich jak Messenger, Telegram czy Signal.
- Pomimo tego, że możliwe jest rozmawianie z kilkoma użytkownikami na raz, poprzez otwarcie nowego czatu w nowej karcie, należałoby upodobnić to do stosowanego w innych komunikatorach rozwiązania, gdzie między konwersacjami można się przełączać na jednej karcie.
- Można by było przygotować wersję mobilną aplikacji klienckiej.
- Dodanie wsparcia dla motywów kolorystycznych.

Wnioski

ASP.NET oraz SignalR pozwalają w prosty sposób przygotować aplikację czasu rzeczywistego, rozumianej jako aplikację pozwalającą na interakcję pomiędzy użytkownikami bez potrzeby odświeżania strony czy okresowego wysyłania żądań do serwera.

Łączenie technologii C# oraz JavaScript oferuje wiele zalet dla rozwijanej aplikacji. C# jest językiem programowania o silnym typowaniu i wydajności, często stosowanym do tworzenia aplikacji backendowych, usług sieciowych i oprogramowania korporacyjnego. Dzięki bogatej bibliotece standardowej, C# zapewnia wiele gotowych rozwiązań i narzędzi do efektywnego programowania. Z drugiej strony, JavaScript jest powszechnie używany do tworzenia interaktywnych interfejsów użytkownika, zarówno w przeglądarkach internetowych, jak i na platformach mobilnych. Jego popularność wynika z elastyczności, dynamicznego typowania i dużej liczby dostępnych bibliotek

i frameworków. Korzystanie z obu technologii pozwala na tworzenie kompletnych rozwiązań, łączących moc C# w zakresie logiki biznesowej i backendowej z dynamicznym, interaktywnym charakterem JavaScriptu dla frontendu.