Technologie oraz protokoły umożliwiające integrację aplikacji mobilnych z serwisami internetowymi (S13)

1 Definicje

Protokół - w najbardziej ogólnym sensie: zestaw reguł umożliwiających porozumienie

Protokół komunikacyjny - ścisła specyfikacja działań jakie podejmują urządzenia, aby ustanowić między sobą połączenie, a następnie móc przekazywać dane

Aplikacja mobilna – ogólna nazwa dla oprogramowania działającego na urządzeniach przenośnych, takich jak telefony komórkowe, smartfony czy tablety

Web service - system zaprojektowany w celu obsługi interakcji pomiędzy maszynami w sieci

2 Integracja aplikacji mobilnych z serwisami internetowymi

Zdecydowana większość współczesnych aplikacji mobilnych opera swoje działanie na dostępnie do Internetu. Wymieniają one niezbędne do swojego działania dane z serwisami internetowymi, dlatego niezbędne są odpowiednie technologie i protokoły, które pozwolą na integrację i zapewnią jednolity sposób komunikacji.

Ze względu na to, że jak sama nazwa wskazuje aplikacje mobilne używane są niemal w każdych warunkach, ważne jest aby zachować responsywność aplikacji tj. pożądane jest, aby odpowiedź z serwisu nadeszła jak najszybciej oraz ze względu na wykorzystanie mobilnego internetu, aby przesyłanych danych było jak najmniej.

3 Wybrane technologie i protokoły

3.1 HTTP i HTTPS

Najczęściej wykorzystywanym protokołem komunikacyjnym, który umożliwia integrację aplikacji mobilnych z serwisami internetowymi jest protokół HTTP (Hypertext Transfer Protocol),

który to z kolei opiera się na protokole TCP.

Jak sama nazwa wskazuje jest to protokół tekstowy i w takim formacie są przesyłane w nim dane. Komunikacja poprzez ten protokół odbywa się poprzez wysyłanie żądań i oczekiwanie na odpowiedzi serwera.

Wersja 1.0 protokołu wydana została w 1996 roku, a obecnie najpopularniejsza wersja 1.1 zaledwie rok później. W 2015 roku wydano usprawnioną wersję 2.0 protokołu o nazwie HTTP/2, która wprowadza wiele usprawnień w stosunku do oryginału m.in. zerwano z tekstowym przesyłaniem danych na rzecz danych binarnych oraz wprowadzono multiplexing, dzięki czemu nie ma już konieczności otwierania wielu osobnych połączeń TCP.

Żądania i odpowiedzi posiadają odpowiedni nagłówek z informacjami zapisanymi jako klucz: wartość oraz opcjonalne ciało, w którym zawarte są dodatkowe przesyłane dane. Protokół korzysta z odpowiednich metod, które wyrażają oczekiwane akcje. Cztery najpopularniejsze metody to: GET, POST, PUT i DELETE.

Każde żądanie HTTP kończy się odpowiedzią od serwera, który zwraca odpowiedni kod odpowiedzi. Grupa 100 to kody informacyjne np. w przypadku przekroczenia czasu połączenia. Grupa 200 to powszechne kody powodzenia, grupa 300 kody przekierowań, grupa 400 – błędy aplikacji klienta np. zasób nie został znaleziony, grupa 500 – błędy po stronie serwera np. błąd wewnętrzny serwera.

Istnieje także bliźniaczy protokół HTTPS, który różni się tym, że połączanie jest szyfrowane za pomocą protokołu TLS.

3.2 SOAP

SOAP (Simple Object Access Protocol) jest protokołem który umożliwia wymianę danych pomiędzy klientem a serwisem internetowym. W chwili obecnej jest to jednak bardzo mało popularne rozwiązanie w kontekście aplikacji mobilnych.

Jest to protokół komunikacyjny zaprojektowany w 1998 roku, a od roku 2003 określany standardem rekomendowanym przez organizację W3C.

Protokół nie zakłada wykorzystania konkretnego protokołu transmisyjnego, lecz najczęściej przesyłanie wiadomości dokonywane jest z wykorzystaniem protokołu HTTP. Struktura wiadomości natomiast musi być obligatoryjnie zakodowana przy pomocy znaczników XML.

Dokument SOAP zawiera w sobie główny znacznik envelope, a następnie obowiązkowy znacznik Body i opcjonalny Header.

XML służy także do opisywania usług SOAP w dokumentach WSDL (Web Services Description Language). Dokumenty te opisują co oferuje usługa, gdzie jest umieszczona oraz jak z niej skorzystać. Tak przygotowane opisy mogą zostać umieszczone następnie w specjalnym rejestrze UDDI (Universal Description, Discovery and Integration).

3.3 REST API

REST API jest w chwili obecnej najpopularniejszym podejściem w integracji aplikacji mobilnych z serwisami internetowymi.

Nie jest to protokół sam w sobie, lecz styl architektoniczny zaproponowany w 2000 roku przez Roya T. Fieldinga. Zakłada on wykorzystanie protokołu HTTP w celu jednokierunkowej komunikacji klient-serwer, a tworzone mogą być z wykorzystaniem niemal dowolnego języka programowania.

W przypadku REST API, każdy zasób jest jednoznacznie identyfikowalny przy pomocy oddzielnego adresu URL i metody HTTP.

Przykłady takich adresów:

- GET /projects
- GET /projects/:id
- GET /projects/:id/tasks
- POST /projects
- GET /tasks
- GET /tasks/:id
- POST /tasks

Podstawowe operacje, które można wykonać na zasobach (tzw. CRUD) mają swoje odpowiedniki w metodach HTTP. Są to odpowiednio:

- Create - POST
- Read - GET
- Update - PUT

- Delete - DELETE

Format w jakim zwracane są dane z serwera (payload) nie jest ściśle określony - może być to np. XML, lecz obecnie zazwyczaj jest to JSON ze względu na lekkość tego formatu.

3.4 GraphQL i Falcor

REST API jest najpopularniejszym rozwiązaniem, ale wadą tego rozwiązania jest nadmierna liczba zapytań i przesyłanie danych, których część może być aktualnie zbędna. W 2015 niezależnie od siebie firmy Facebook oraz Netflix wydały dwa rozwiązania, które pozwalają zniwelować te problemy i ograniczyć ruch w sieci.

Rozwiązanie Facebooka - GraphQL jest jedynie zatwierdzoną specyfikacją i doczekało się wielu implementacji w różnych językach programowania, z kolei Falcor - rozwiązanie Netflixa dostępne jest wyłącznie jako biblioteka JavaScript.

Oba rozwiązania wymagają określenia po stronie serwera struktury danych i tego w jaki sposób można uzyskać do nich dostęp. GraphQL wymaga zdefiniowanie schemy i resolverów, gdzie określamy też dostępne zapytania i mutacje - czyli to co można pobrać i zmodyfikować. Falcor wymaga zdefiniowanie jednego, dużego obiektu - JSON Virtual Graph do którego można odwoływać się przy pomocy odpowiednich ścieżek, tutaj określone są też odpowiednie metody get, set i call, które pozwalają na zwracanie i manipulację danych.

Oba te rozwiązania pozwalają na wybór wyłącznie niezbędnych danych. W przypadku GraphQL tworzy się zapytanie o odpowiedniej strukturze, a odpowiedź z serwera ją odzwierciedla. Falcor wymaga podania odpowiednich ścieżek, które odpowiadają tym zdefiniowanym w wirtualnym grafie, dzięki czemu zwracane są wybrane, odpowiednie części tego grafu.

Komunikacja z serwerem GraphQL odbywa się poprzez wysyłanie wszystkich zapytań pod jeden adres POST. W przypadku Falcora założeniem jest, że operujemy na danych tak jakby dostępne były one lokalnie. Podajemy odpowiednią ścieżkę, a bezpośredniej komunikacji z serwerem dokonuje "pod spodem" już sama biblioteka.

3.5 Websockets

Websockety zapełniają dwukierunkowy kanał komunikacji za pośrednictwem jednego gniazda TCP. Zanim to jednak nastąpi musi zostać nawiązane połączenie przy pomocy protokołu HTTP i dopiero po wykonaniu HTTP Upgrade, komunikacja przechodzi na protokół TCP.

Podobnie jak w przypadku protokołów HTTP i HTTPS, w przypadku websocketów dostępne są odpowiednie URI - ws dla połączeń zwykłych i wss dla szyfrowanych.

Websockety umożliwiają wielokrotne przesyłanie danych w dowolną ze stron tj. od serwera do klienta lub od klienta do serwera, aż do momentu zamknięcia połączenia przez którąś ze stron. Przesyłane dane mogą mieć dowolny format, ważne jest tylko to, aby był on jednakowy i zrozumiały dla obu stron.

3.6 gRPC

RPC, czyli zdalne wywołanie procedury jest protokołem bardzo starym i jeszcze do niedawna był bardzo rzadko stosowany w świecie aplikacji mobilnych. Sytuacja zmieniła się w 2015 roku po wydaniu przez firmę Google nowoczesnego frameworka gRPC.

gRPC do komunikacji wykorzystuje protokół HTTP/2. Można wyróżnić dwa dostępne typy komunikacji: unarny i strumieniowanie. Ten pierwszy pozwala na wywołanie pojedynczego zapytania i otrzymanie odpowiedzi z serwera. Strumieniowanie z kolei oznacza możliwość wysyłanie wielu wiadomości (strumienia) i można podzielić je na trzy rodzaje:

- klient wysyła strumień wiadomości do serwera
- serwer wysyła strumień wiadomości do klienta
- klient i serwer wysyłają strumień wiadomości wzajemnie

Co ważne w każdym przypadku to klient jako pierwszy musi zapoczątkować połączenie - nawet w przypadku wysyłania wiadomości z serwera do klienta.

gRPC wspiera wiele formatów danych, jednak domyślnie opiera się na Protocol Buffers, które odpowiadają także za serializację i deserializację danych. W przeciwieństwie do JSONa, czy XML dane przesyłane są w formie binarnej, a nie tekstowej, co także wpływa pozytywnie na szybkość i ilość przesyłanych danych.

Serwisy opisywane są w specjalnych plikach .proto, gdzie znajdują się informacje o dostępnych serwisach i strukturze wiadomości. Tak przygotowane pliki mogą zostać następnie wykorzystane do wygenerowania kodu w wybranym języku programowania, co pozwoli na wywoływanie procedur.

3.7 Push Notifications

Technologia powiadomień push, czyli tych wyświetlanych u góry ekranu smartfona także pozwala na integrację z serwisami internetowymi.

Technologia ta pozwala na wysyłanie krótkich wiadomości do urządzeń klienckich. Odbywa się to z wykorzystaniem serwera pośredniczącego do którego wysyłane są wiadomości, a on dystrybuuje je już dalej, bezpośrednio do urządzeń klienckich.

Co ważne nie ma gwarancji tego, kiedy wiadomość dotrze do klienta. Jeśli jest on niedostępny w momencie wysłania wiadomości, wiadomość zostanie do niego dostarczona w późniejszym czasie, kiedy nawiąże on połączenie z Internetem. Czasem może to prowadzić do sytuacji, że otrzymywane wiadomości są już nieaktualne.

Domyślnie do komunikacji wykorzystywany jest protokół HTTP. Istnieje jednak możliwość wykorzystania także protokołu XMPP, dzięki czemu dodatkowo możliwe jest wysyłanie wiadomości w drugą stronę - od urządzenia klienckiego do serwera.

Dostępne są dwa rodzaje przesyłanych wiadomości - standardowa notyfikacja wyświetlana u góry ekranu lub wiadomość z danymi, która nie jest wyświetlana. Obsługą pierwszej z nich zajmuje się system operacyjny, drugi rodzaj natomiast należy obsłużyć samodzielnie wewnątrz własnej aplikacji. Istnieje także możliwość połączenia tych rodzajów i dodanie własnych pól z danymi do standardowej wiadomości.

Aby przesyłanie takich wiadomości było możliwe, niezbędne jest uzyskanie odpowiednich tokenów, które pozwolą na jednoznaczną identyfikację urządzenia i poprawne wysyłanie skierowanych do niego wiadomości.

4 Reprezentacja danych

4.1 JSON

JSON (JavaScript Object Notation) – lekki format wymiany danych komputerowych. JSON jest formatem tekstowym, bazującym na podzbiorze języka JavaScript.

Pomimo nazwy, JSON jest formatem niezależnym od konkretnego języka. Wiele języków programowania obsługuje ten format danych przez dodatkowe pakiety bądź biblioteki.

Nazwy właściwości w JSON są otoczone cudzysłowami, a wartości mogą być typu: string, number lub jedną ze stałych: true, false lub null. Jako wartość dozwolone są także obiekty i tablice obiektów, które mogą być dowolnie zagnieżdżane.

Przykład:

```
{
  "employees": [
    { "firstName": "John", "lastName": "Doe" },
    { "firstName": "Anna", "lastName": "Smith" },
    { "firstName": "Peter", "lastName": "Jones" }
  ]
}
```

4.2 XML

XML (Extensible Markup Language) – uniwersalny język znaczników przeznaczony do reprezentowania różnych danych w strukturalizowany sposób. Jest niezależny od platformy, co umożliwia łatwą wymianę dokumentów pomiędzy różnymi systemami. XML jest standardem rekomendowanym oraz specyfikowanym przez organizację W3C.

Każdy element musi zaczynać się znacznikiem początkowym oraz kończyć identycznym znacznikiem końcowym (wyjątkiem są elementy puste).

Elementy można dowolnie zagnieżdżać w sobie. Informacje, które zawiera element muszą być zapisane pomiędzy znacznikiem początku i końca elementu.

Przykład:

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
```

```
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```