Making Phone Calls from Blazor WebAssembly with Twilio Voice

https://swimburger.net - @RealSwimburger

About me

- Niels Swimberghe
  aka Swimburger
- Grew up in Belgium, working in USA
- .NET Developer / Tech Content Creator
- Blog at swimbuger.net
- Twitter: @RealSwimburger
- Company: MetroStar


twilio
SOLUTIONS
Twilio Flex
Marketing Campaigns
IDENTITY
Authy
Verify
Lookup
INTELLIGENCE
Autopilot
Conversations
ORCHESTRATION
TaskRouter
Studio
CHANNEL APIS
SMS
Voice
Email
Video
Chat
WhatsApp
Facebook Messenger
SUPER NETWORK
Phone Numbers
Short Codes
Interconnect
SIP
SIP
IoT

Programmatic communication using **HTTP Webhooks**

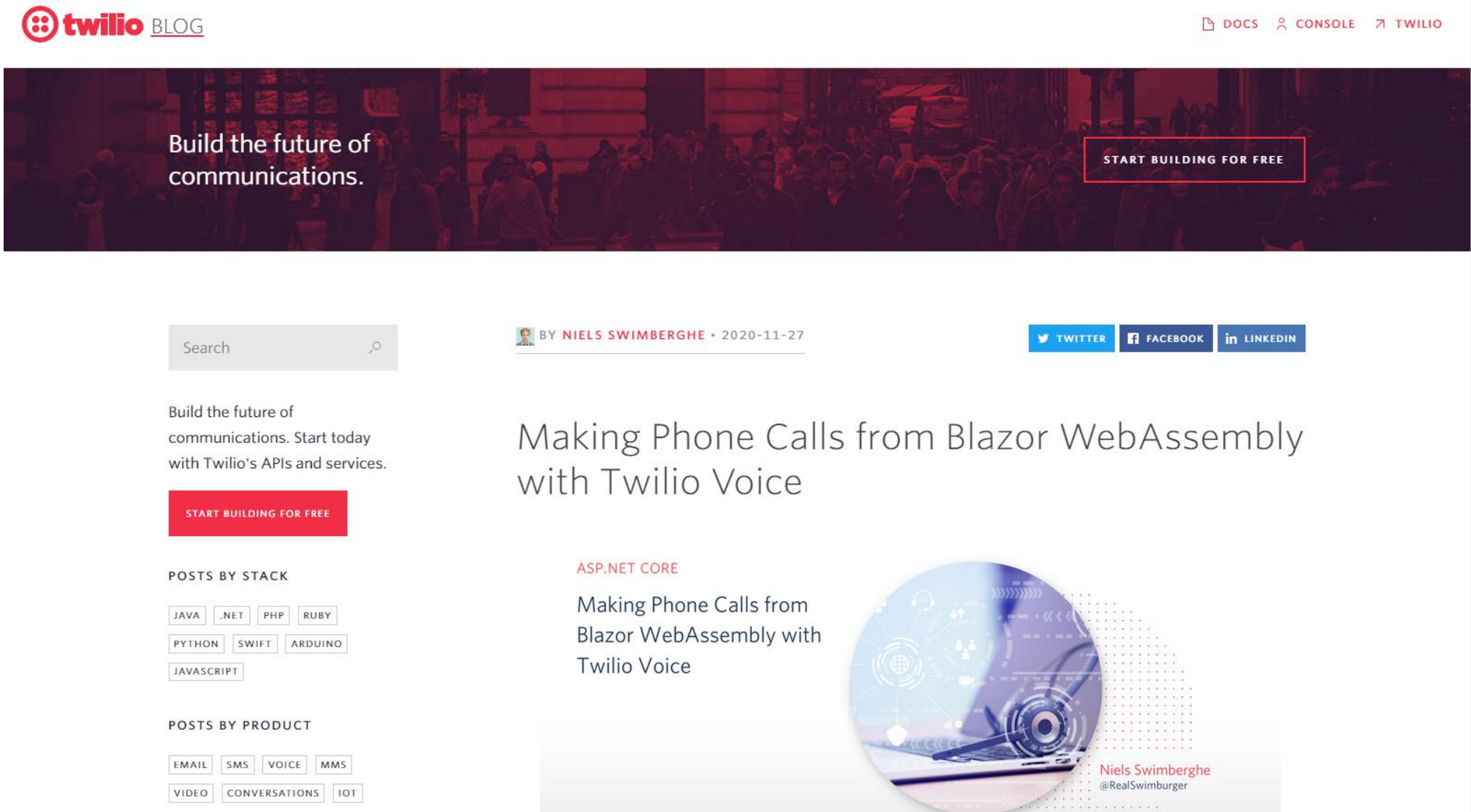
Example TwiML

Sample: https://demo.twilio.com/welcome/voice/

```
<Response>
    <Say voice="alice">Thanks for the call. Configure your number's voice U R L to change this message.</Say>
    <Pause length="1"/>
    <Say voice="alice">Let us know if we can help you in any way during your development.</Say>
</Response>
```

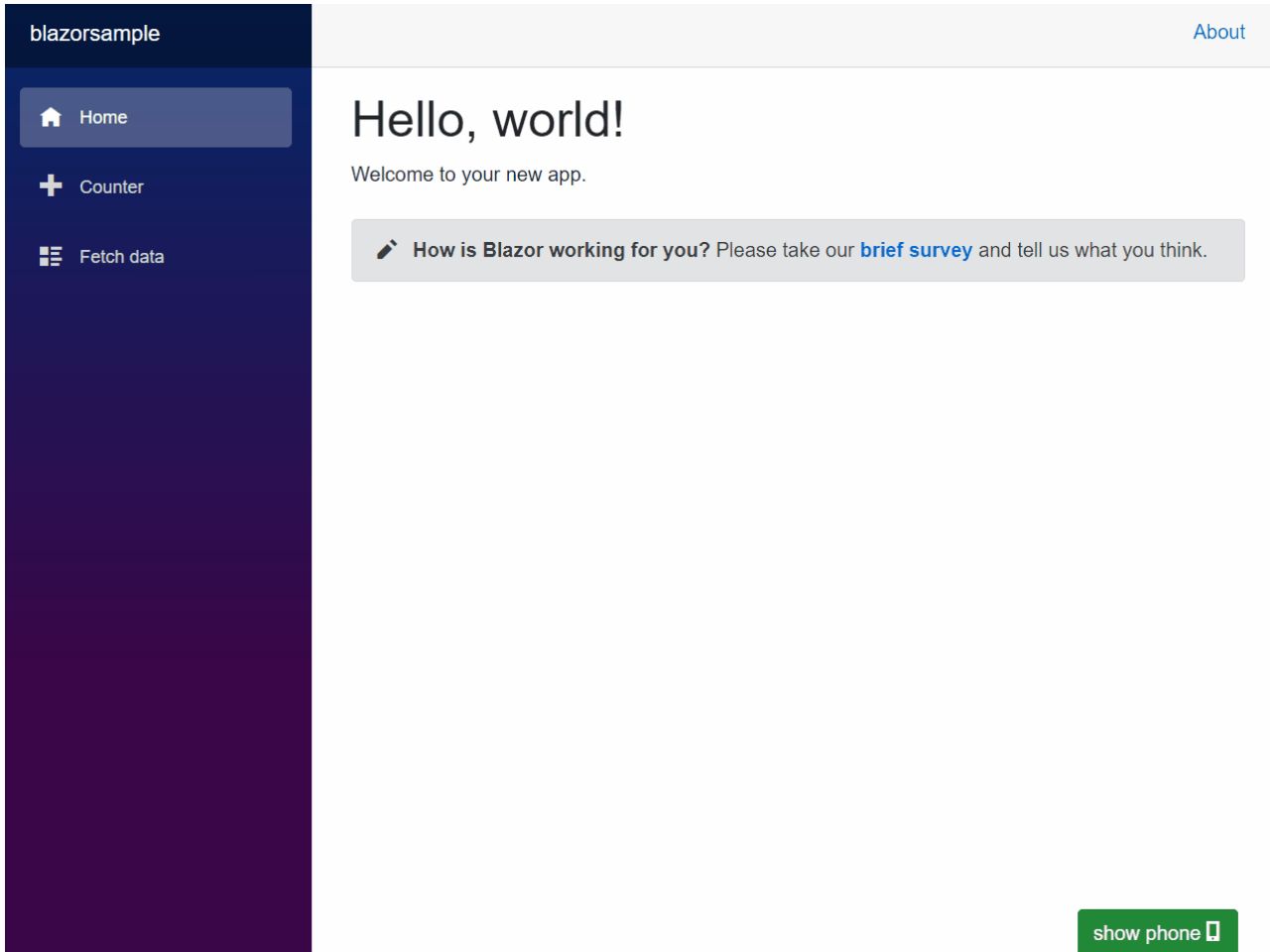
Based on in-depth guide on Twilio Blog

Check out [guide at Twilio Blog]

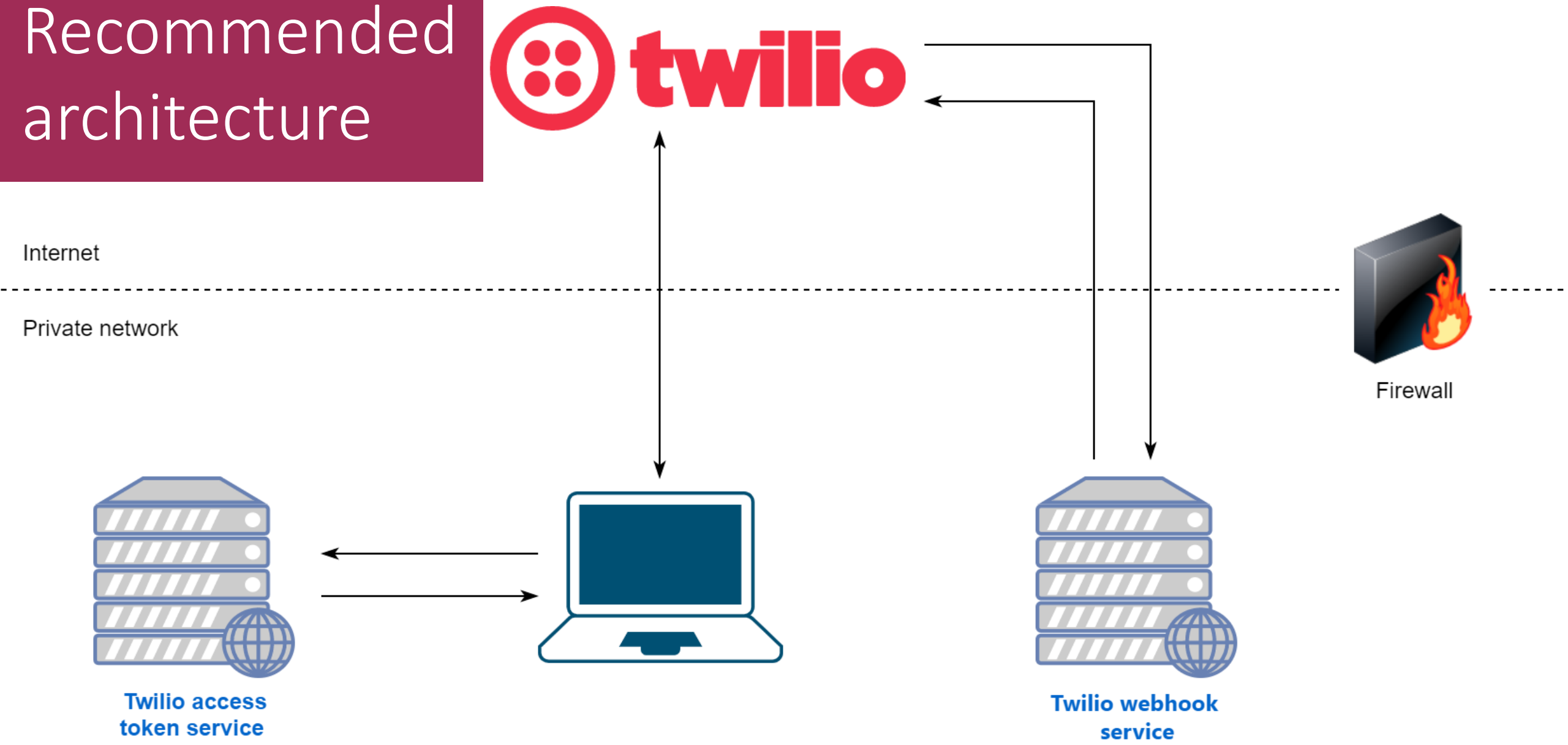


Application

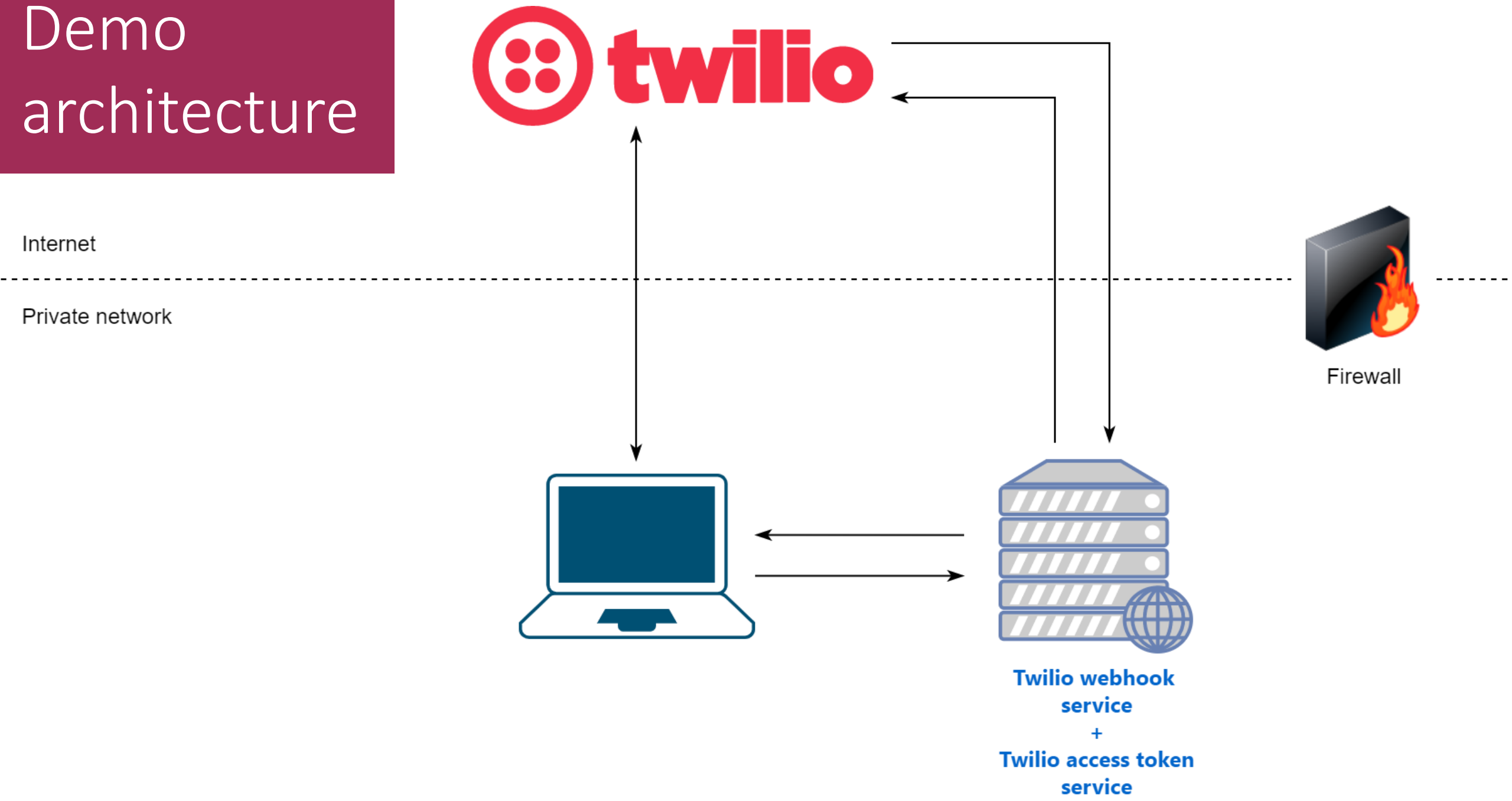
- Out of the box Blazor WebAssembly application
- Phone dialer
  - Initiate phone calls from browser
  - Receive phone calls in browser



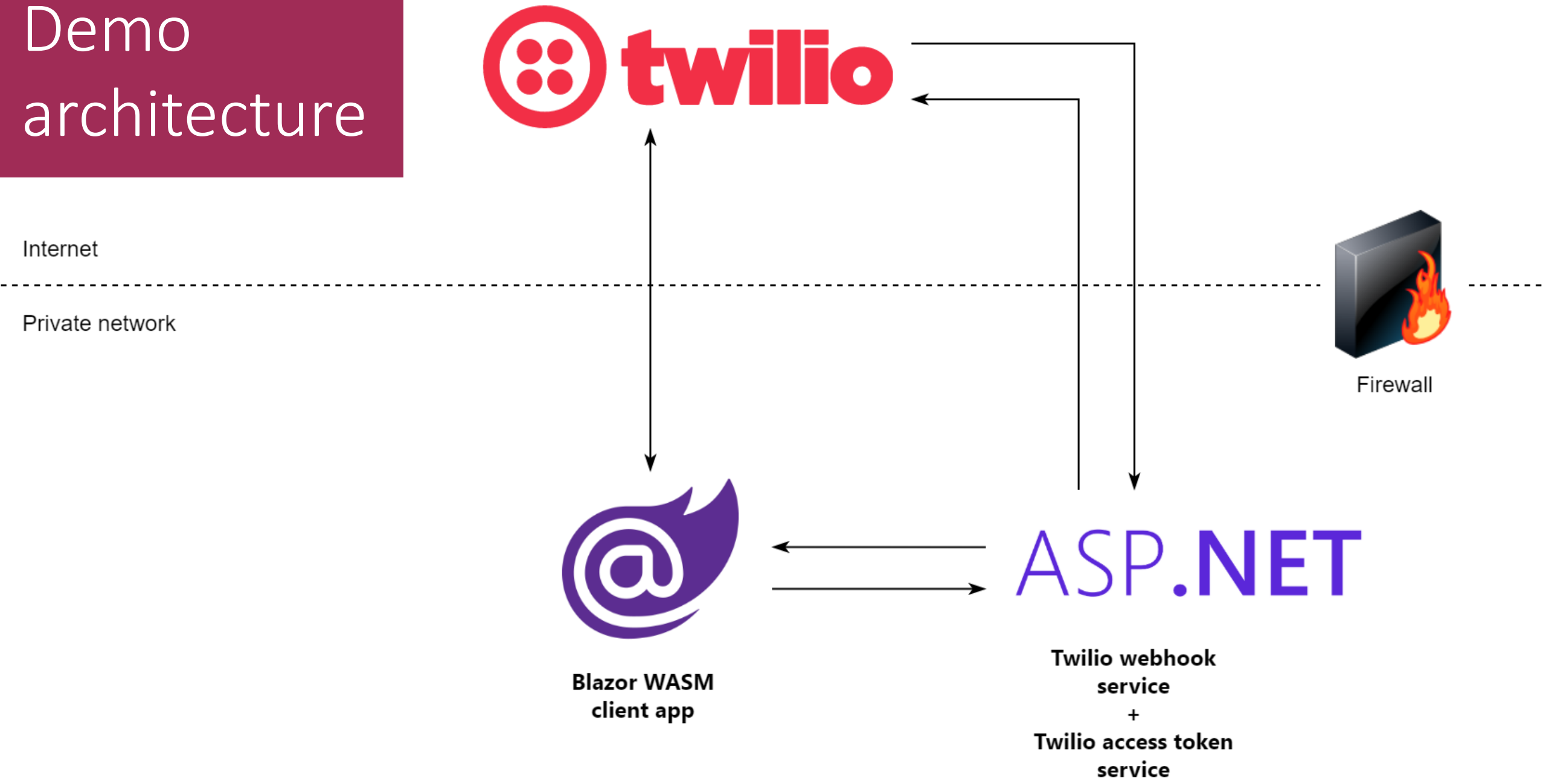
Recommended architecture



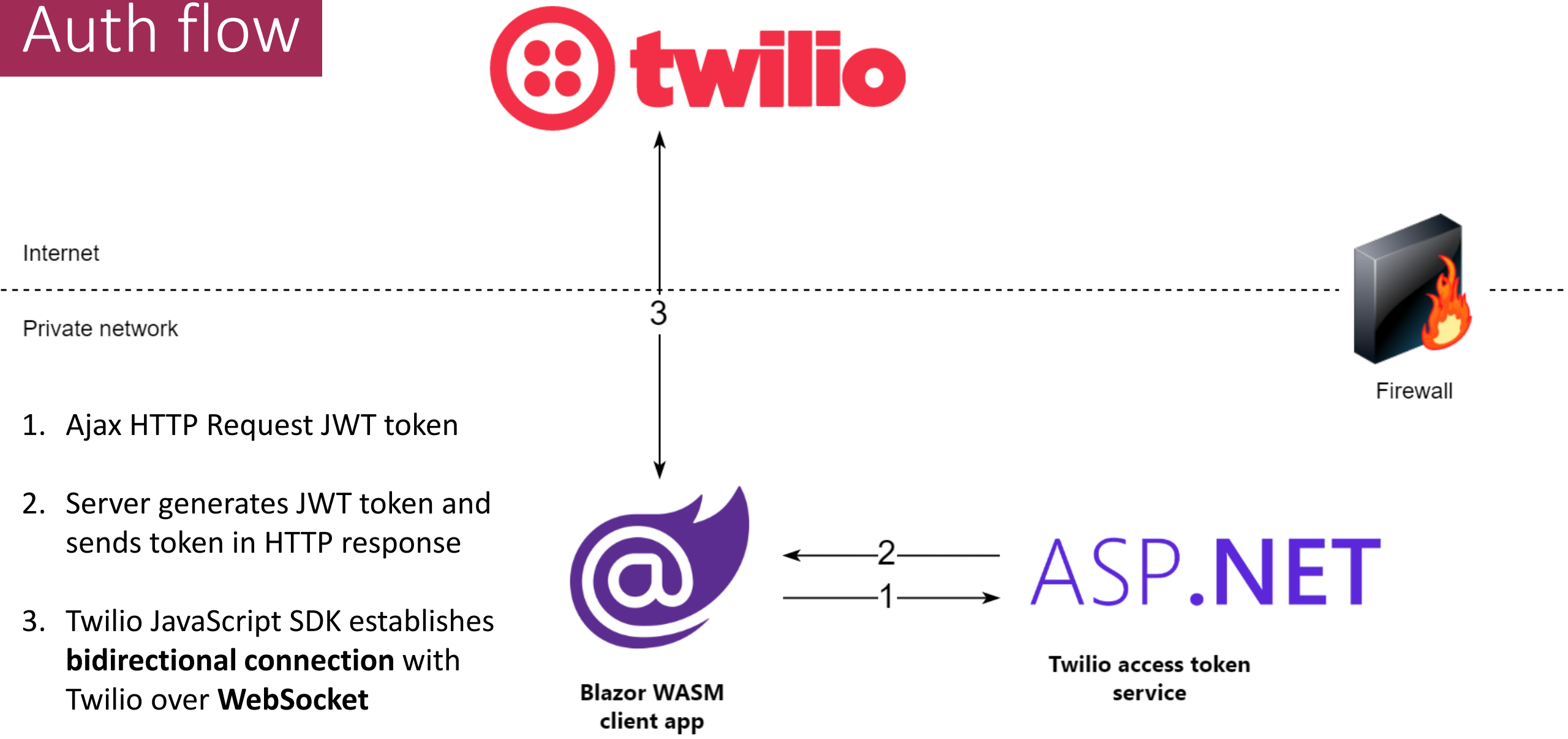
Demo architecture



Demo architecture

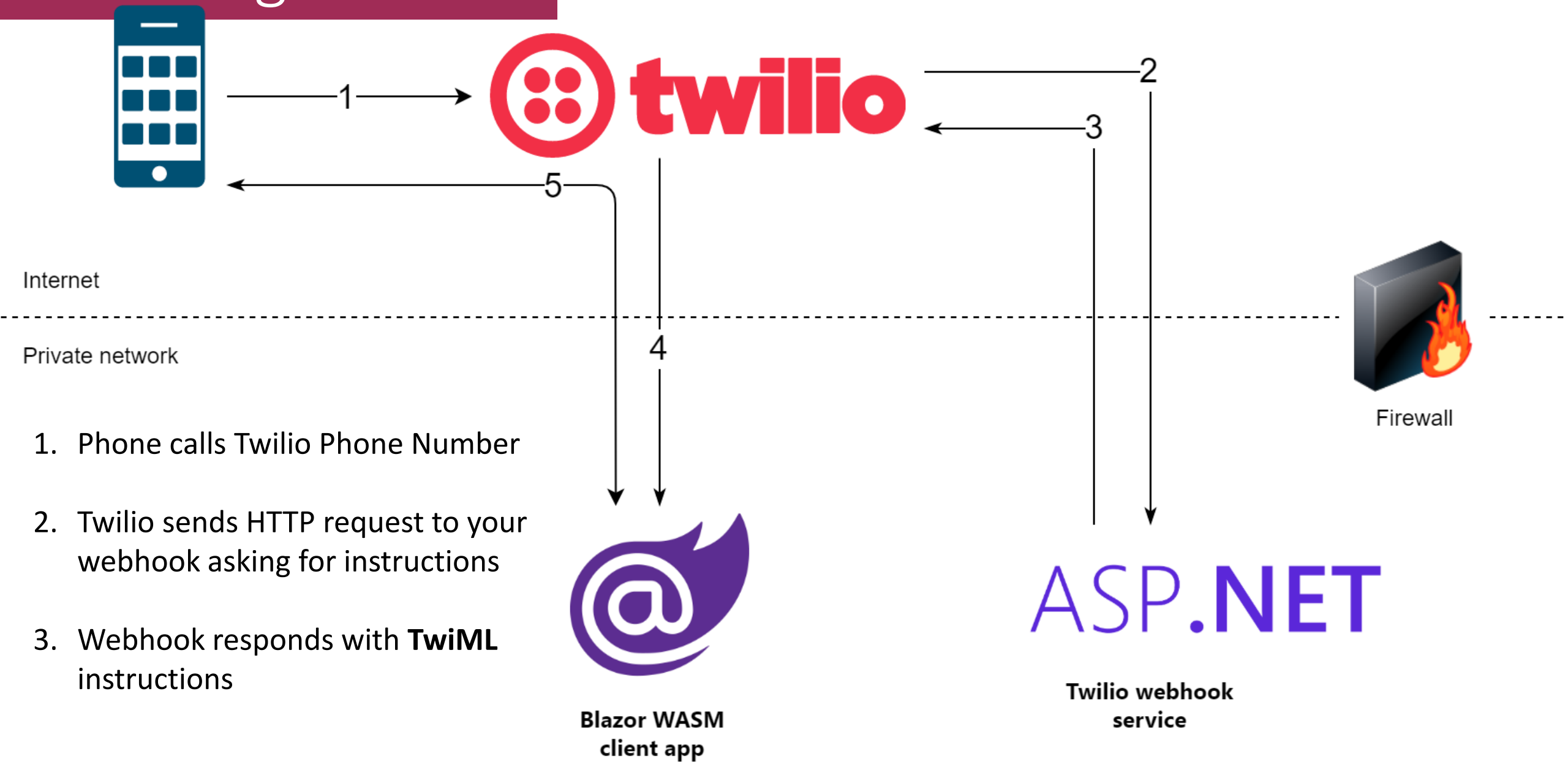


Auth flow



1. Ajax HTTP Request JWT token
2. Server generates JWT token and sends token in HTTP response
3. Twilio JavaScript SDK establishes **bidirectional connection** with Twilio over **WebSocket**

Incoming call flow



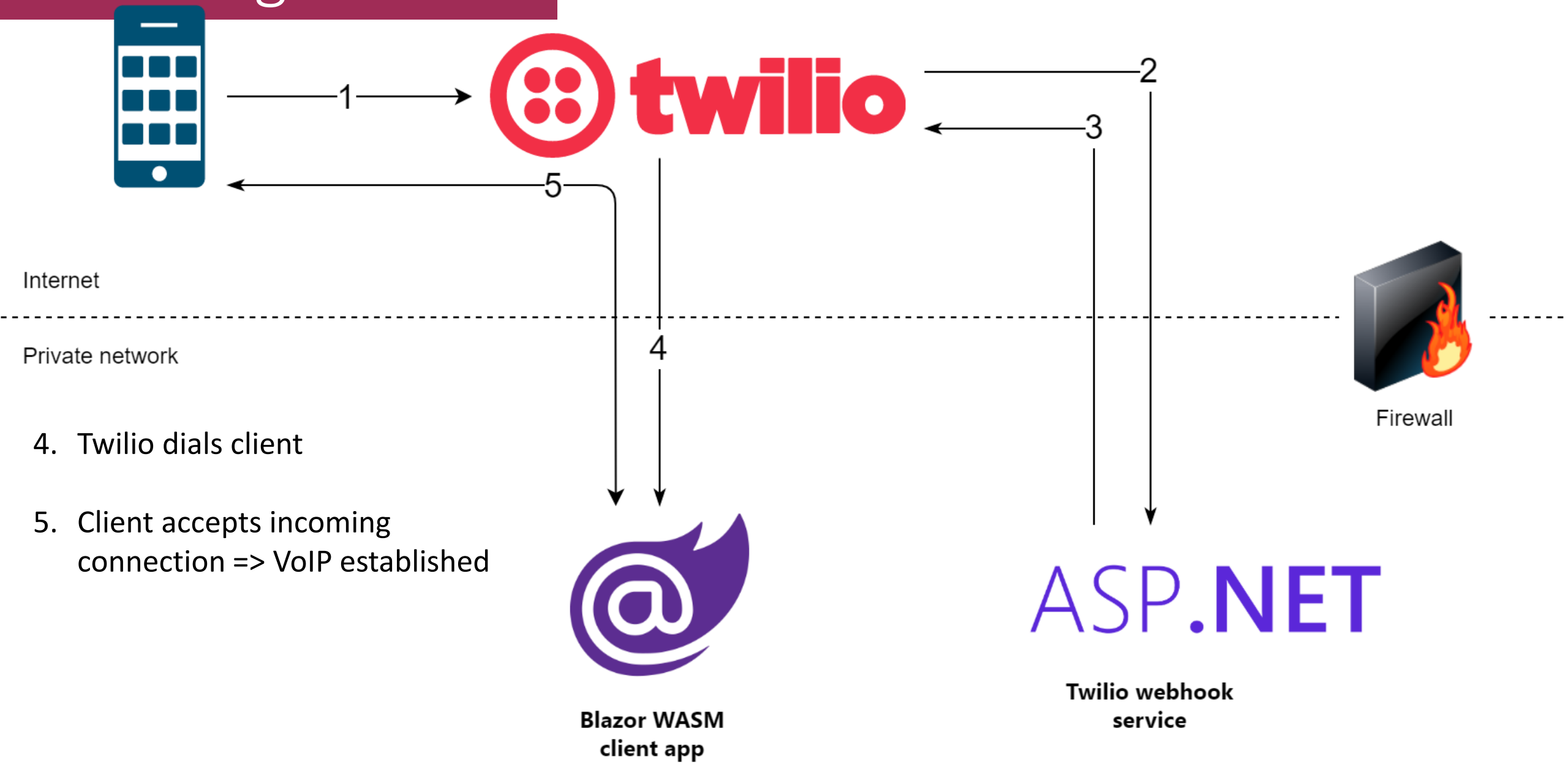
1. Phone calls Twilio Phone Number
2. Twilio sends HTTP request to your webhook asking for instructions
3. Webhook responds with **TwiML** instructions

Incoming call flow

Webhook responds with TwiML instructions

```
<?xml version="1.0" encoding="utf-8"?>
<Response>
    <Dial callerId="+1234567890">
        <Client>blazor_client</Client>
    </Dial>
</Response>
```

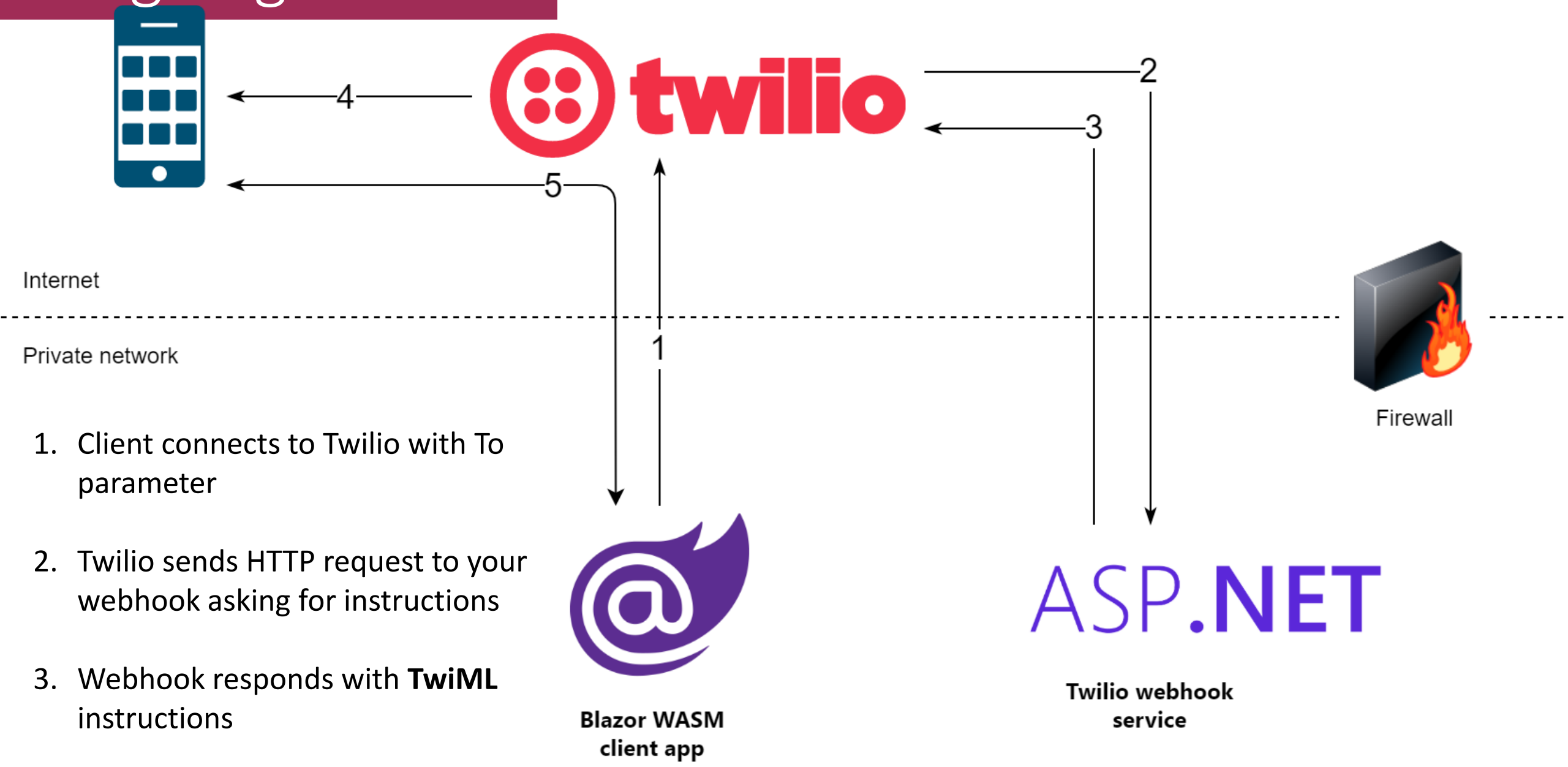
Incoming call flow



4. Twilio dials client

5. Client accepts incoming connection => VoIP established

Outgoing call flow



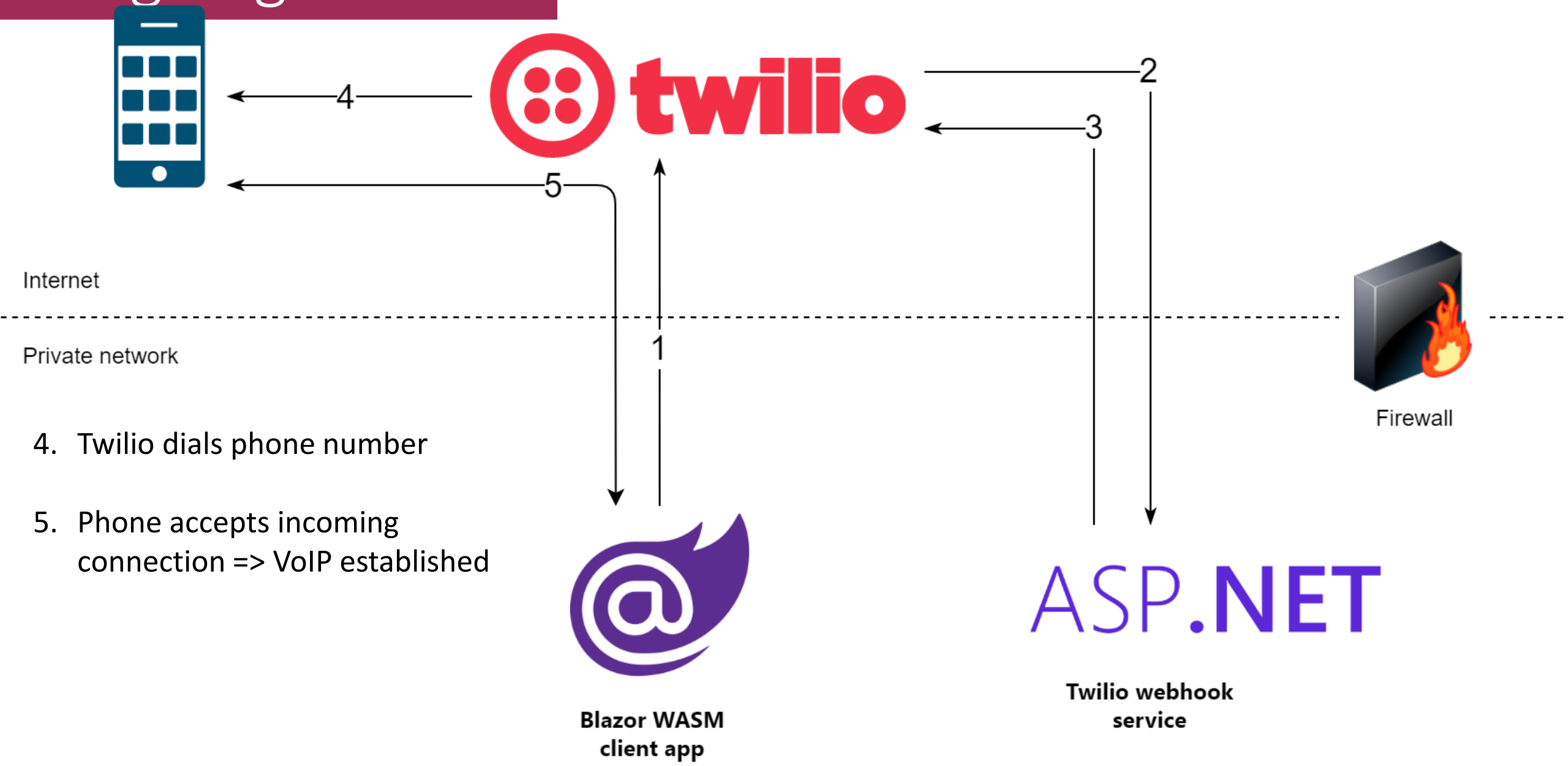
1. Client connects to Twilio with To parameter
2. Twilio sends HTTP request to your webhook asking for instructions
3. Webhook responds with **TwiML** instructions

Outgoing call flow

Webhook responds with TwiML instructions

```
<?xml version="1.0" encoding="utf-8"?>
<Response>
    <Dial callerId="+17035926931">
        <Number>+1234567890</Number>
    </Dial>
</Response>
```

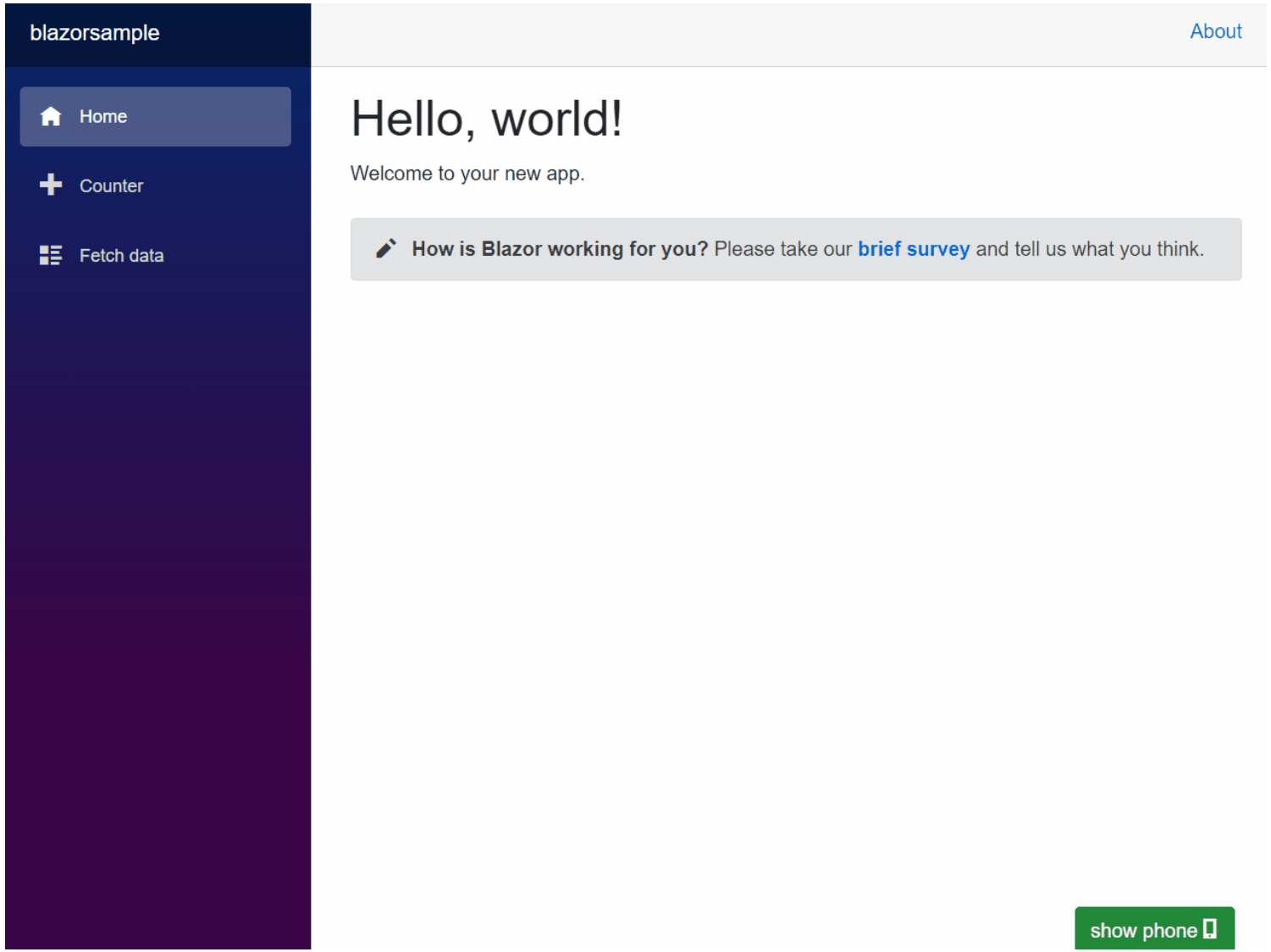
Outgoing call flow



4. Twilio dials phone number

5. Phone accepts incoming connection => VoIP established

Let’s see how its built



Step 1: Create Twilio resources

- You need to create these Twilio resources
  - **Twilio account**
  - **Twilio phone number** (to make phone calls)
  - **TwiML application** (to configure outgoing voice webhook)
  - **Twilio API key & secret**

Buy phone number using Twilio CLI

```
~\source\repos\TwilioBlazorPhonecalls>
```

Phone number details

```
C:\Users\niels> twilio phone-numbers:list -o=json
[
  {
    "accountSid": "AC89dbe1dc475a98405be881dd9cb4e868",
    "addressSid": null,
    "addressRequirements": "none",
    "apiVersion": "2010-04-01",
    "beta": false,
    "capabilities": {
      "voice": true,
      "sms": true,
      "mms": true
    },
    "dateCreated": "2021-05-31T03:59:49.000Z",
    "dateUpdated": "2021-05-31T03:59:49.000Z",
    "friendlyName": "(703) 215-1589",
    "identitySid": null,
    "phoneNumber": "+17032151589",
    "origin": "twilio",
    "sid": "PN93f01c1f91b31ffb48b34468d7b6b125",
    "smsApplicationSid": "",
    "smsFallbackMethod": "POST",
    "smsFallbackUrl": "",
    "smsMethod": "POST",
    "smsUrl": "https://demo.twilio.com/welcome/sms/reply/",
    "statusCallback": "",
    "statusCallbackMethod": "POST",
    "trunkSid": null,
```

incoming webhook HTTP method

incoming webhook URL

Create TwiML application Twilio CLI

```
C:\Users\niels> twilio api:core:applications:create --friendly-name TwilioBlazorPhonecalls
SID                                Friendly Name           Date Created
AP60604a5b56131e5b6c8d8fdf5431795c  TwilioBlazorPhonecalls  May 30 2021 23:19:21 GMT-0500
```

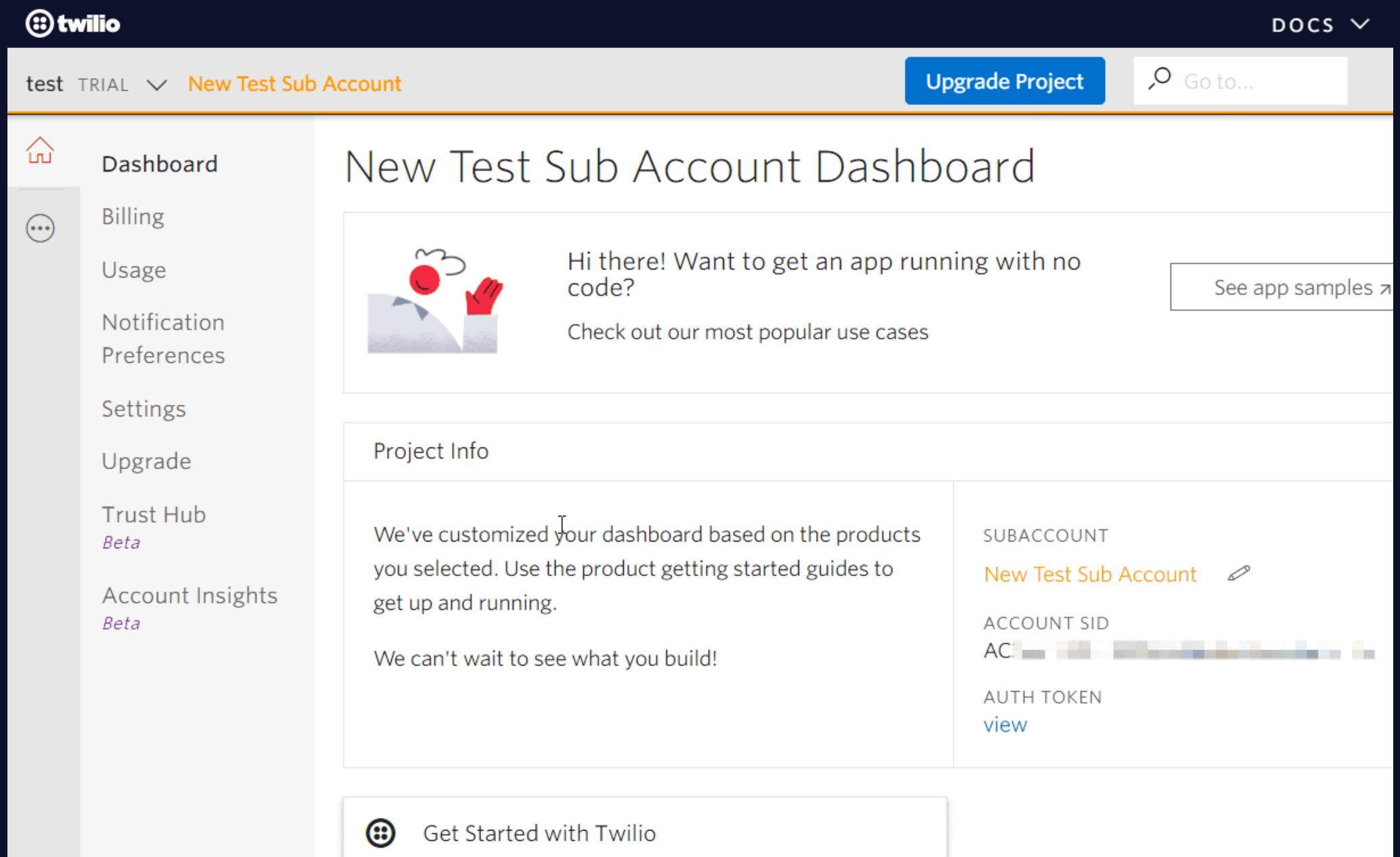
TwiML application details

```
C:\Users\niels> twilio api:core:applications:fetch --sid=AP60604a5b56131e5b6c8d8fdf5431795c -o=json
[
  {
    "accountSid": "AC89dbe1dc475a98405be881dd9cb4e868",
    "apiVersion": "2010-04-01",
    "dateCreated": "2021-05-31T04:19:21.000Z",
    "dateUpdated": "2021-05-31T04:19:21.000Z",
    "friendlyName": "TwilioBlazorPhonecalls",
    "messageStatusCallback": null,
    "sid": "AP60604a5b56131e5b6c8d8fdf5431795c",
    "smsFallbackMethod": "POST",
    "smsFallbackUrl": null,
    "smsMethod": "POST",
    "smsStatusCallback": null,
    "smsUrl": null,
    "statusCallback": null,
    "statusCallbackMethod": "POST",
    "uri": "/2010-04-01/Accounts/AC89dbe1dc475a98405be881dd9cb4e868/Applications/AP60604a5b56131e5b6c8
    "voiceCallerIdLookup": false,
    "voiceFallbackMethod": "POST",
    "voiceFallbackUrl": null,
    "voiceMethod": "POST",
    "voiceUrl": null
  }
]
```

outgoing webhook HTTP method

outgoing webhook URL

Create Twilio API Key



Why API Keys?

BY NIELS SWIMBERGHE ▪ 2021-03-01

TWITTER FACEBOOK LINKEDIN

Better Twilio Authentication with Twilio API Keys

URL: https://www.twilio.com/blog/better-twilio-authentication-csharp-twilio-api-keys

Step 2: Create access token service

- Create ASP.NET Core WebAPI project
- Update server URLs
- Configure secrets & keys
- Add Twilio NuGet package
- Create token controller
- Configure CORS

Create ASP.NET Core WebAPI project

```
> dotnet new webapi -o TwilioBlazorPhonecalls.Server
```

Update server URLs

in TwilioBlazorPhonecalls.Server\Properties\launchSettings.json

```
{
  "$schema": "http://json.schemastore.org/launchsettings.json",
  "profiles": {
    "TwilioBlazorPhonecalls.Server": {
      "commandName": "Project",
      "applicationUrl": "https://localhost:5003;http://localhost:5002",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```

Configure secrets with some PS help

```powershell
$TwilioAccountSid       = Read-Host 'Enter your Twilio Account SID'
$TwilioPhoneNumber      = Read-Host 'Enter your Twilio phone number'
$TwilioApiKey           = Read-Host 'Enter your Twilio API key'
$TwilioApiSecret        = Read-Host 'Enter your Twilio API secret' -AsSecureString
$TwiMLApplicationSid    = Read-Host 'Enter your TwiML Application SID'

$ProjectName = "TwilioBlazorPhonecalls.Server"
dotnet user-secrets init --project $ProjectName
dotnet user-secrets set "TwilioAccountSid" $TwilioAccountSid --project $ProjectName
dotnet user-secrets set "TwilioPhoneNumber" $TwilioPhoneNumber --project $ProjectName
dotnet user-secrets set "TwilioApiKey" $TwilioApiKey --project $ProjectName

# set secret and mask the output with ***
(dotnet user-secrets set "TwilioApiSecret" (ConvertFrom-SecureString $TwilioApiSecret -AsPlainText) --project $ProjectName)`
    -replace (ConvertFrom-SecureString $TwilioApiSecret -AsPlainText), ((ConvertFrom-SecureString $TwilioApiSecret -AsPlainText) -replace ".", "*")

dotnet user-secrets set "TwiMLApplicationSid" $TwiMLApplicationSid --project $ProjectName
```

Configure secrets with some PS help

```
~\source\repos\TwilioBlazorPhonecalls [master ≡ +0 ~2 -0 !]> .\ConfigureSecrets.ps1
```

Add Twilio NuGet package

```
> dotnet add TwilioBlazorPhonecalls.Server package Twilio
```

Add token controller

```
[ApiController]
[Route("token")]
0 references
public class TokenController : ControllerBase
{
    5 references
    private readonly IConfiguration configuration;
    0 references
    public TokenController(IConfiguration configuration)···

    [HttpGet]
    [EnableCors("BlazorClientPolicy")]
    0 references
    public string Get()
    {
        // set config using ConfigureSecrets.ps1
        string twilioAccountSid = configuration["TwilioAccountSid"];
        string twilioApiKey = configuration["TwilioApiKey"];
        string twilioApiSecret = configuration["TwilioApiSecret"];
        string twiMLApplicationSid = configuration["TwiMLApplicationSid"];

        var grants = new HashSet<IGrant>();
        // Create a Voice grant for this token
        grants.Add(new VoiceGrant
```

Configure CORS in Startup

```
public class Startup
{
    1 reference
    private const string BlazorClientPolicyName = "BlazorClientPolicy";

    // This method gets called by the runtime. Use this method to add services to the container.
    0 references
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddCors(options =>
        {
            options.AddPolicy(name: BlazorClientPolicyName, builder =>
            {
                builder.WithOrigins("https://localhost:5001", "http://localhost:5000");
            });
        });

        // omitted for brevity
    }
```

Configure CORS in Controller

```
public class TokenController : ControllerBase
{
    5 references
    private readonly IConfiguration configuration;
    0 references
    public TokenController(IConfiguration configuration)···

    [HttpGet]
    [EnableCors("BlazorClientPolicy")]
    0 references
    public string Get()
```

Verify token controller using PS

```
C:\Users\niels> Invoke-WebRequest https://localhost:5003/token

StatusCode        : 200
StatusDescription : OK
Content           : eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImN0eSI6InR3aWxpby1mcGE7dj
                    5LCJqdGkiOiJTSzE5ODg4ODYxODZmNGNhMDg5ODdiYTZiNDVmODdj…
RawContent        : HTTP/1.1 200 OK
                    Date: Mon, 31 May 2021 06:29:29 GMT
                    Server: Kestrel
                    Transfer-Encoding: chunked
                    Content-Type: text/plain; charset=utf-8

                    eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImN0eSI6InR3aWxpby1mcG…
Headers           : {[Date, System.String[]], [Server, System.String[]], [Transfer
Images            : {}
InputFields       : {}
Links             : {}
RawContentLength  : 512
RelationLink      : {}
```

Verify CORS using PS

```
C:\Users\niels>  Invoke-WebRequest https://localhost:5003/token -Method Options -Headers @{
>>      "Access-Control-Request-Method" = "GET"
>>      "Access-Control-Request-Headers" = "origin, x-requested-with"
>>      "Origin" = "https://localhost:5001"
>>  }

StatusCode        : 204
StatusDescription : NoContent
Content           : {}
RawContent        : HTTP/1.1 204 NoContent
                    Date: Mon, 31 May 2021 06:30:49 GMT
                    Server: Kestrel
                    Vary: Origin
                    Access-Control-Allow-Origin: https://localhost:5001


Headers           : {[Date, System.String[]], [Server, System.String[]], [Vary, System.Strin
RawContentLength  : 0
RelationLink      : {}
```

Demo Twilio JavaScript SDK

```
let response = await fetch("https://localhost:5003/token");
let jwtToken = await response.text();

let device = new Twilio.Device();
device.setup(jwtToken);

device.on('ready', () => console.log('ready'));
device.on('error', (error) => console.error(error));
device.on('connect', (connection) =>
{
    console.log('connect');
    console.log(connection);
});
device.on('disconnect', () => console.log('disconnect'));
device.on('incoming', (connection) =>
{
    console.log('incoming');
    console.log(connection);
});
device.on('cancel', () => console.log('cancel'));

// start call
```

Step 3: Add webhooks for Twilio Voice

- Add voice controller for webhooks (incoming & outgoing action)
- Use **ngrok** to publicly expose local server
- Update incoming & outgoing webhook (phone number & TwiML application)

Add voice controller for webhooks

```
[HttpPost]
[Route("voice/outgoing")]
0 references
public TwiMLResult Outgoing([FromForm] string to)
{
    string twilioPhoneNumber = configuration["TwilioPhoneNumber"];
    var response = new VoiceResponse();
    var dial = new Dial();
    if (enablePrivacyMode)···
    else
    {
        logger.LogInformation($"Calling {to}");
    }
    dial.CallerId = twilioPhoneNumber;
    dial.Number(to);
    response.Append(dial);
    return TwiML(response);
}

[HttpPost]
[Route("voice/incoming")]
0 references
public TwiMLResult Incoming([FromForm] string from)
{
```

Verify voice controller using PS

```
C:\Users\niels> Invoke-WebRequest https://localhost:5003/voice/incoming -Method Post -Body @{"From" = "+1234567890"}

StatusCode        : 200
StatusDescription : OK
Content           : <?xml version="1.0" encoding="utf-8"?>
                    <Response>
                      <Dial callerId="+1234567890">
                        <Client>blazor_client</Client>
                      </Dial>
                    </Response>


C:\Users\niels> Invoke-WebRequest https://localhost:5003/voice/Outgoing -Method Post -Body @{"To" = "+1234567890"}

StatusCode        : 200
StatusDescription : OK
Content           : <?xml version="1.0" encoding="utf-8"?>
                    <Response>
                      <Dial callerId="+17035926931">
                        <Number>+1234567890</Number>
                      </Dial>
                    </Response>
```

Use ngrok to publicly expose local server

```
C:\Users\niels> ngrok http https://localhost:5003

ngrok by @inconshreveable

Session Status                online
Account                       Niels Swimberghe (Plan: Free)
Update                        update available (version 2.3.40, Ctrl-U to update)
Version                       2.3.35
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://dbe6fa0943aa.ngrok.io -> https://localhost:5003
Forwarding                    https://dbe6fa0943aa.ngrok.io -> https://localhost:5003

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

Update incoming & outgoing webhook

```
$ApplicationSid = "AP24aaf7938ece2d44e2f89b13131c4e41";
$TwilioPhonenumber = "+17032151589";

# Make sure ngrok is running before running this
$NgrokApi = 'http://127.0.0.1:4040/api';
$TwilioTunnelsObject = (Invoke-WebRequest "$NgrokApi/tunnels").Content | ConvertFrom-Json;
$PublicBaseUrl = $TwilioTunnelsObject.tunnels.public_url | Where-Object {$_.StartsWith('https')};

$PublicOutgoingVoiceUrl = "$PublicBaseUrl/voice/outgoing";
twilio api:core:applications:update --sid=$ApplicationSid --voice-url=$PublicOutgoingVoiceUrl --voice-method=POST;

$PublicIncomingVoiceUrl = "$PublicBaseUrl/voice/incoming";
twilio phone-numbers:update $TwilioPhonenumber --voice-url=$PublicIncomingVoiceUrl --voice-method=POST;
```

Demo Twilio JavaScript SDK with webhooks

```
// start call
device.connect({ "To": "+1234567890" });

// end call
device.activeConnection().disconnect();

// accept incoming call
device.activeConnection().accept();

// reject incoming call
device.activeConnection().reject();
device.destroy();
```

Step 4: Create Blazor WebAssembly client

- Create Blazor WASM project
- Add Twilio JavaScript SDK
- Create Blazor Dialer component
  - Razor, CSS
  - JavaScript & .NET interop

Create a Blazor WebAssembly project

```
> dotnet new blazorwasm -o TwilioBlazorPhonecalls.Client
```

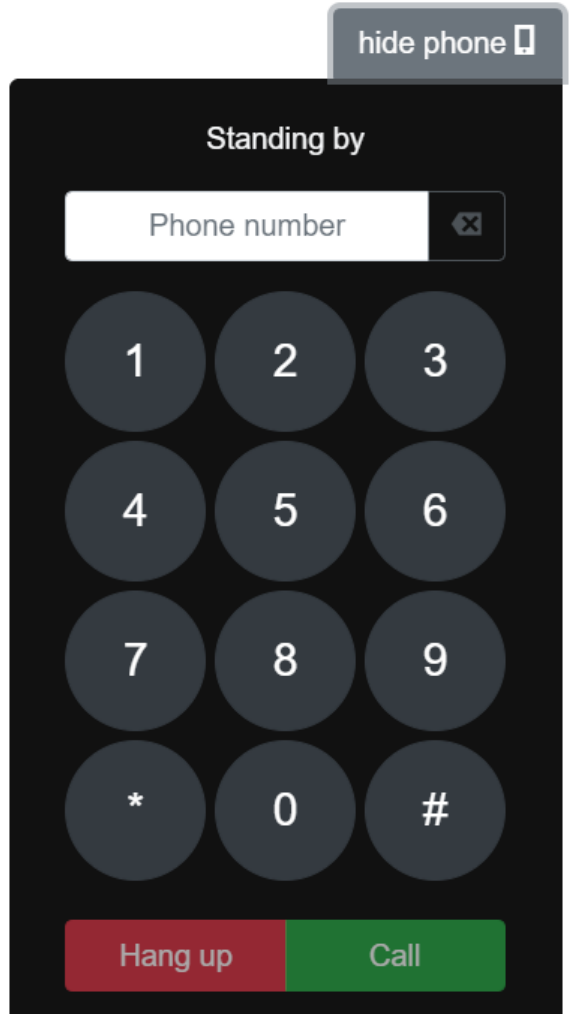
Add Twilio JavaScript SDK

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no" />
    <title>TwilioBlazorPhonecalls.Client</title>
    <base href="/" />
    <link href="css/bootstrap/bootstrap.min.css" rel="stylesheet" />
    <link href="css/app.css" rel="stylesheet" />
    <script src="https://sdk.twilio.com/js/client/releases/1.12.5/twilio.js"></script>
</head>
<body>
    <app>Loading...</app>

    <div id="blazor-error-ui">
        An unhandled error has occurred.
        <a href="" class="reload">Reload</a>
        <a class="dismiss">✕</a>
    </div>
    <script src="_framework/blazor.webassembly.js"></script>
</body>
</html>
```

Create Dialer component

- Razor, CSS
- JavaScript & .NET interop



Communicating between .NET and JavaScript in Blazor with in-browser samples

Niels Swimberghe - 11/5/2020 - .NET

Follow me on Twitter, buy me a coffee

Sample:

Send message from .NET to JavaScript

Message from .NET to JavaScript: **Hello from .NET**

JavaScript Source:

```
window.dotNetToJsSamples = {
    setText: function(node, text)
    {
        node.textContent = text;
    }
};
```

Blazor Source:

```
@inject IJSRuntime js

<button type="button" class="btn btn-primary" @onclick="PrintMessageAsync">Send message from .NET to JavaScript</button>
<br><br>
<div class="alert alert-primary" role="alert">
  Message from .NET to JavaScript:
  <span class="font-weight-bold" @ref="spanMessage"></span>
</div>

@code{
  private ElementReference spanMessage;

  private async Task PrintMessageAsync()
  {
    await js.InvokeVoidAsync("dotNetToJsSamples.setText", spanMessage, "Hello from .NET");
  }
}
```

Call static .NET ConsoleWriteLine function from JavaScript

Call static .NET NamedConsoleWriteLine function named 'otherConsoleWriteLine' from JavaScript

Blazor Source:

Note the usage of the JavaScript `onclick` attribute instead of the Blazor `@onclick` attribute.

```
<button type="button" class="btn btn-primary" onclick="DotNet.invokeMethod('BlazorJavaScriptInterop', 'ConsoleWriteLine', 'Hello from JavaScript')">
  Call static .NET ConsoleWriteLine function from JavaScript
</button>
<br>
<br>
<button type="button" class="btn btn-primary" onclick="DotNet.invokeMethod('BlazorJavaScriptInterop', 'otherConsoleWriteLine', 'Hello from JavaScript 2')">
  Call static .NET NamedConsoleWriteLine function  named 'otherConsoleWriteLine' from JavaScript
</button>

@code{
  [JSInvokable]
  public static void ConsoleWriteLine(string message)
  {
    Console.WriteLine(message);
  }

  [JSInvokable("otherConsoleWriteLine")]
  public static void NamedConsoleWriteLine(string message)
  {
    Console.WriteLine(message);
  }
}
```

Blazor Source:

```
@implements IDisposable
@inject IJSRuntime js

<button type="button" class="btn btn-primary" onclick="jsToDotNetSamples.printPersonFromDotNet()">Print person from .NET Blazor component instance method</button>
<br><br>
<code id="personCodeBlock"></code>

@code{
  private DotNetObjectReference<JsToDotNetSample3> dotNetObjectReference;

  protected override Task OnInitializedAsync()
  {
    dotNetObjectReference = DotNetObjectReference.Create(this);
    js.InvokeVoidAsync("jsToDotNetSamples.setDotNetReference", dotNetObjectReference);
    return base.OnInitializedAsync();
  }

  [JSInvokable]
  public object GetPerson()
  {
    var obj = new
    {
      FistName = "Jon",
      LastName = "Doe",
      Age = 27,
      BirthDate = DateTime.Now.AddYears(-27),
      LikesDotNet = true
    };
    return obj;
  }

  public void Dispose()
  {
    dotNetObjectReference.Dispose();
  }
}
```

Print person from .NET Blazor component instance method

{"fistName":"Jon","lastName":"Doe","age":27,"birthDate":"1994-06-06T18:16:43.325-04:00","likesDotNet":true}

JavaScript Source:

```
window.jsToDotNetSamples = {
    dotNetReference: null,
    setDotNetReference: function(dotNetReference)
    {
        this.dotNetReference = dotNetReference;
    },
    printPersonFromDotNet: function(){
        var person = this.dotNetReference.invokeMethod("GetPerson");
        document.getElementById('personCodeBlock').textContent = JSON.stringify(person);
    }
};
```

Print person from .NET Blazor component instance method

{"fistName":"Jon","lastName":"Doe","age":27,"birthDate":"1994-06-06T18:16:43.325-04:00","likesDotNet":true}

JavaScript Source:

```
window.jsToDotNetSamples = {
    dotNetReference: null,
    setDotNetReference: function(dotNetReference)
    {
        this.dotNetReference = dotNetReference;
    },
    printPersonFromDotNet: function(){
        var person = this.dotNetReference.invokeMethod("GetPerson");
        document.getElementById('personCodeBlock').textContent = JSON.stringify(person);
    }
};
```

Blazor Source:

```
@implements IDisposable
@inject IJSRuntime js

<button type="button" class="btn btn-primary" onclick="jsToDotNetSamples.printPersonFromDotNet()">Print person from .NET Blazor component instance method</button>
<br><br>
<code id="personCodeBlock"></code>

@code{
  private DotNetObjectReference<JsToDotNetSample3> dotNetObjectReference;

  protected override Task OnInitializedAsync()
  {
    dotNetObjectReference = DotNetObjectReference.Create(this);
    js.InvokeVoidAsync("jsToDotNetSamples.setDotNetReference", dotNetObjectReference);
    return base.OnInitializedAsync();
  }

  [JSInvokable]
  public object GetPerson()
  {
    var obj = new
    {
      FistName = "Jon",
      LastName = "Doe",
      Age = 27,
      BirthDate = DateTime.Now.AddYears(-27),
      LikesDotNet = true
    };
    return obj;
  }

  public void Dispose()
  {
    dotNetObjectReference.Dispose();
  }
}
```

Print person from .NET Blazor component instance method

{"fistName":"Jon","lastName":"Doe","age":27,"birthDate":"1994-06-06T18:18:26.079-04:00","likesDotNet":true}

JavaScript Source:

```
window.jsToDotNetSamples = {
    dotNetReference: null,
    setDotNetReference: function(dotNetReference)
    {
        this.dotNetReference = dotNetReference;
    },
    printPersonFromDotNet: function(){
        var person = this.dotNetReference.invokeMethod("GetPerson");
        document.getElementById('personCodeBlock').textContent = JSON.stringify(person);
    }
};
```

Dialer calling JS functions

1 reference
```
private async Task StartCall()
{
    await js.InvokeVoidAsync("dialer.startCall", dialNumber);
}
```

1 reference
```
private async Task EndCall()
{
    await js.InvokeVoidAsync("dialer.endCall");
}
```

1 reference
```
private async Task AcceptCall()
{
    await js.InvokeVoidAsync("dialer.acceptCall");
}
```

1 reference
```
private async Task RejectCall()
{
    hasIncomingConnection = false;
    message = "";
    await js.InvokeVoidAsync("dialer.rejectCall");
}
```

.NET methods to called by JS

```
[JSInvokable]
0 references
public void OnTwilioDeviceReady()
{
    isTwilioDeviceReady = true;
    StateHasChanged();
}

[JSInvokable]
0 references
public void OnTwilioDeviceError(string error)
{
    message = error;
    logger.LogError(error);
    StateHasChanged();
}

[JSInvokable]
0 references
public void OnTwilioDeviceConnected(string phoneNumber)
{
    isTwilioDeviceConnected = true;
    hasIncomingConnection = false;
    if(enablePrivacyMode){
        phoneNumber = MaskPhoneNumber(phoneNumber);
    }
    message = $"Calling {phoneNumber}";
    StateHasChanged();
}
```

JS events calling back into .NET

```
setupTwilioEvents: function () {
    // inside of Twilio events scope the 'this' context is redefined and will not point to 'window.dialer'
    // the variable 'self' is introduced to conveniently access the 'this' context from the outer scope inside of the events scope
    var self = this;
    this.device.on('ready', function () {
        self.dotNetObjectReference.invokeMethod('OnTwilioDeviceReady');
    });

    this.device.on('error', function (error) {
        console.error(error);
        self.dotNetObjectReference.invokeMethod('OnTwilioDeviceError', error.message);
    });

    this.device.on('connect', function (connection) {
        if (connection.direction === "OUTGOING") {
            self.dotNetObjectReference.invokeMethod('OnTwilioDeviceConnected', connection.message['To']);
        } else {
            self.dotNetObjectReference.invokeMethod('OnTwilioDeviceConnected', connection.parameters['From']);
        }
    });

    this.device.on('disconnect', function () {
        self.dotNetObjectReference.invokeMethod('OnTwilioDeviceDisconnected');
    });

    this.device.on('incoming', function (connection) {
        self.dotNetObjectReference.invokeMethod('OnTwilioDeviceIncomingConnection', connection.parameters['From']);
    });

    this.device.on('cancel', function () {
        self.dotNetObjectReference.invokeMethod('OnTwilioDeviceCanceled');
    });
},
```

Improvements to consider

- Validate webhook requests come from Twilio
- Split up webhook service from token service
- Add authentication & authorization to the token controller
- Handle no-pick up scenario’s
- Use JavaScript Blazor interop improvements from .NET 5
- Use minimal API from .NET 6

More .NET, Web, Azure, etc.

swimburger.net

@RealSwimburger

yt.swimburger.net

fb.swimburger.net