Laboratório

Olá Xamarin!

Versão: 1.0.0
Abril de 2017

Cynthia Zanoni
@cynthiazanoni

Suporte: maratonaxamarin@microsoft.com

CONTEÚDO

Introdução

Aqui terá acesso ao código-fonte de um aplicativo em Xamarin Forms, pronto para integração com back-end desenvolvido com o recurso Easy Tables.

Objetivos

Após a conclusão deste laboratório, os participantes serão capazes de:

- Criar uma conexão Xamarin com Azure.

Requisitos

Para acompanhar o hands on do vídeo, você precisará ter instalado em sua máquina uma versão do Visual Studio e uma conta no Microsoft Azure.

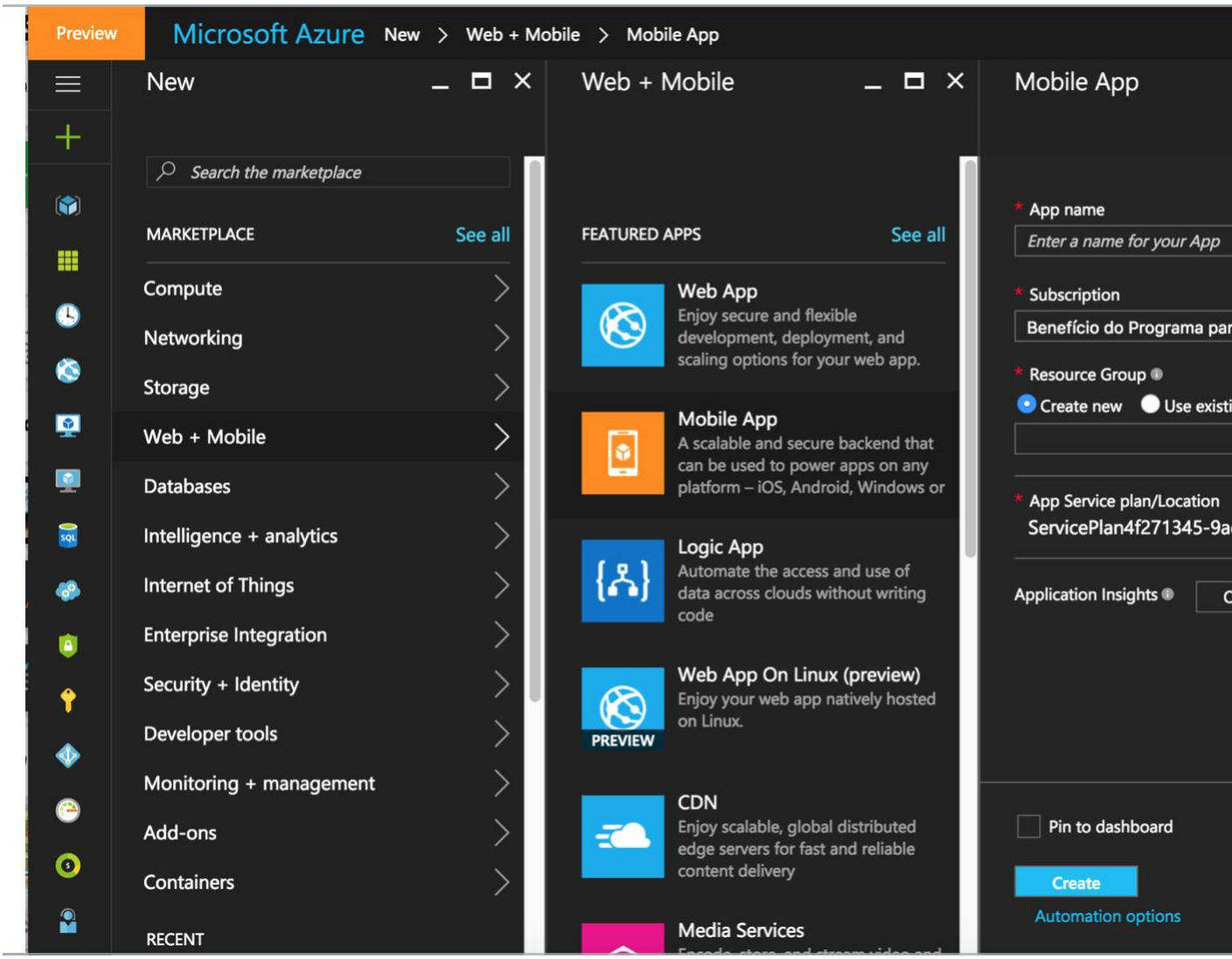
- Faça download gratuito do Visual Studio Community 2015. Após a instalação, faça a configuração dos emuladores de android.
- Crie sua conta gratuita no Microsoft Azure.
- Uma equipe de desenvolvimento com o sistema operacional Windows 10 e Visual Studio 2015 o 2017 Community, Professional ou Enterprise.

  Tempo estimado para completar este laboratório: **60 minutos**.

Exercício 1: Conexão Xamarin com Azure

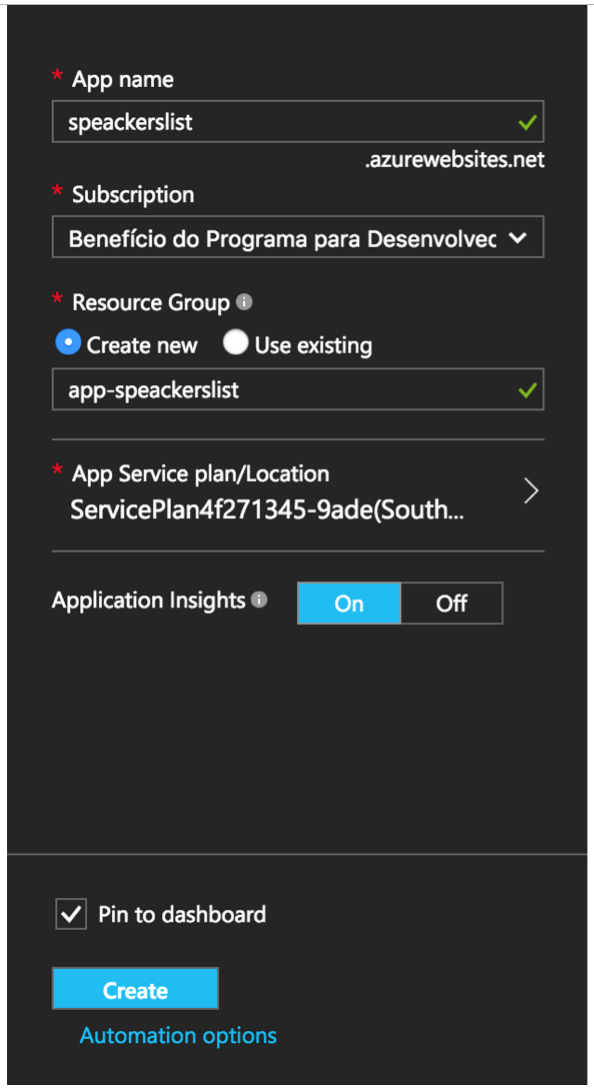
Criando o App Service

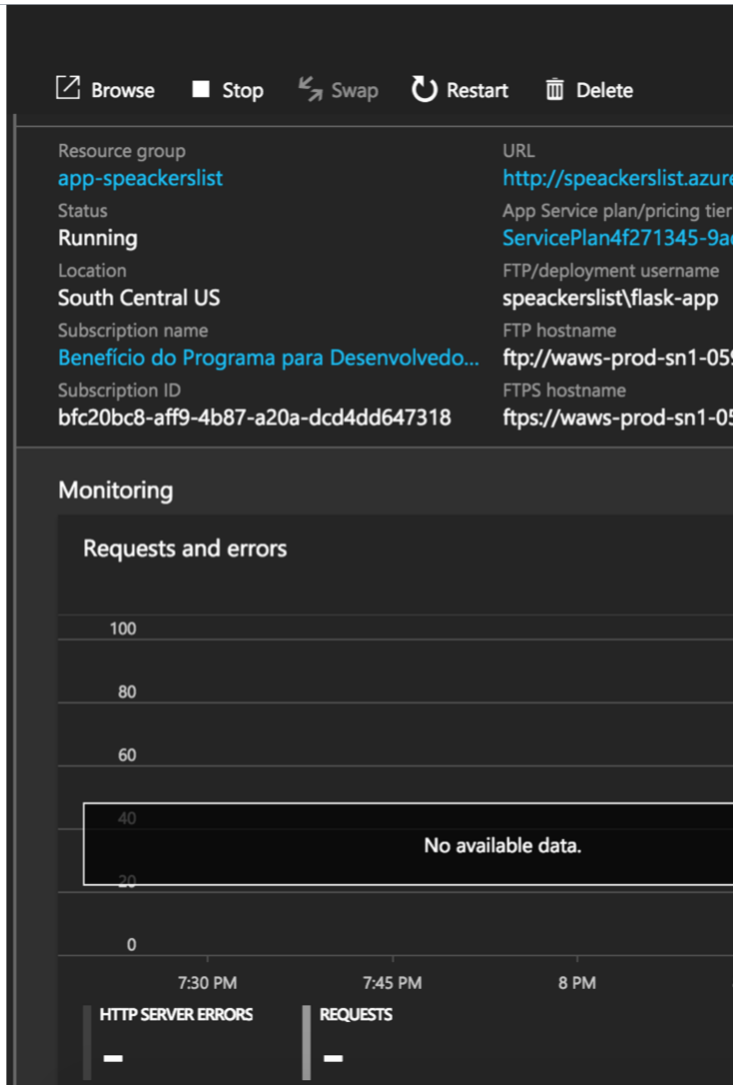
Na tela inicial do portal do Microsoft Azure, você deverá clicar no botão de **new (representação pelo símbolo +)** e localizar a opção `Web + Mobile`. Ao clicar no menu, você encontrará uma lista com alguns dos principais serviços disponíveis neste recurso. Para criar o back-end, vamos utilizar o `Mobile App`. Na aba seguinte, vamos definir o nome da nossa aplicação, resource group e também habilitar o Application Insights.



Acessando o App Service

Com o deploy concluído, teremos acesso ao dashboard do nosso aplicativo, nele conseguimos visualizar as principais informações e recursos disponíveis. Teremos, também, um gráfico resumido do Application Insights e no momento em que iniciarmos a utilização do back-end, registrará informações como requests e performance.

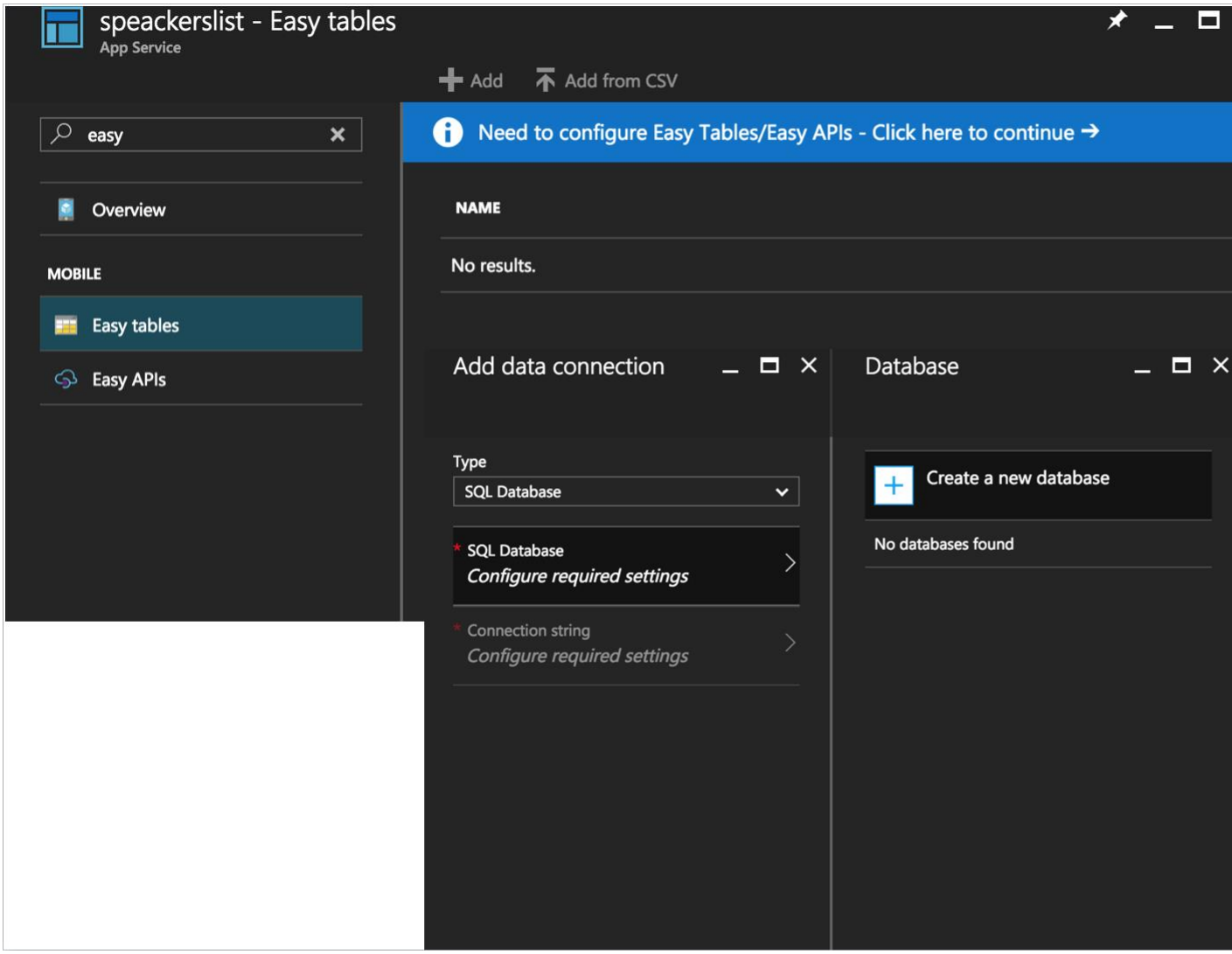




Conexão de Dados

O próximo passo é a criação da nossa Easy Tables e para isso precisamos criar uma data connection e um servidor. No menu lateral do dashboard do App Service, procure por Easy Tables e na sequência, clique na barra azul com a legenda `Need to configure Easy Tables/Easy APIs - Click here to continue`. Selecione o tipo de conexão de dados, que pode ser SQL Database ou Storage e clique na opção abaixo do seletor para realizar as configurações necessárias. Na aba que irá surgir, clique em `Create new database` e define o nome do banco.

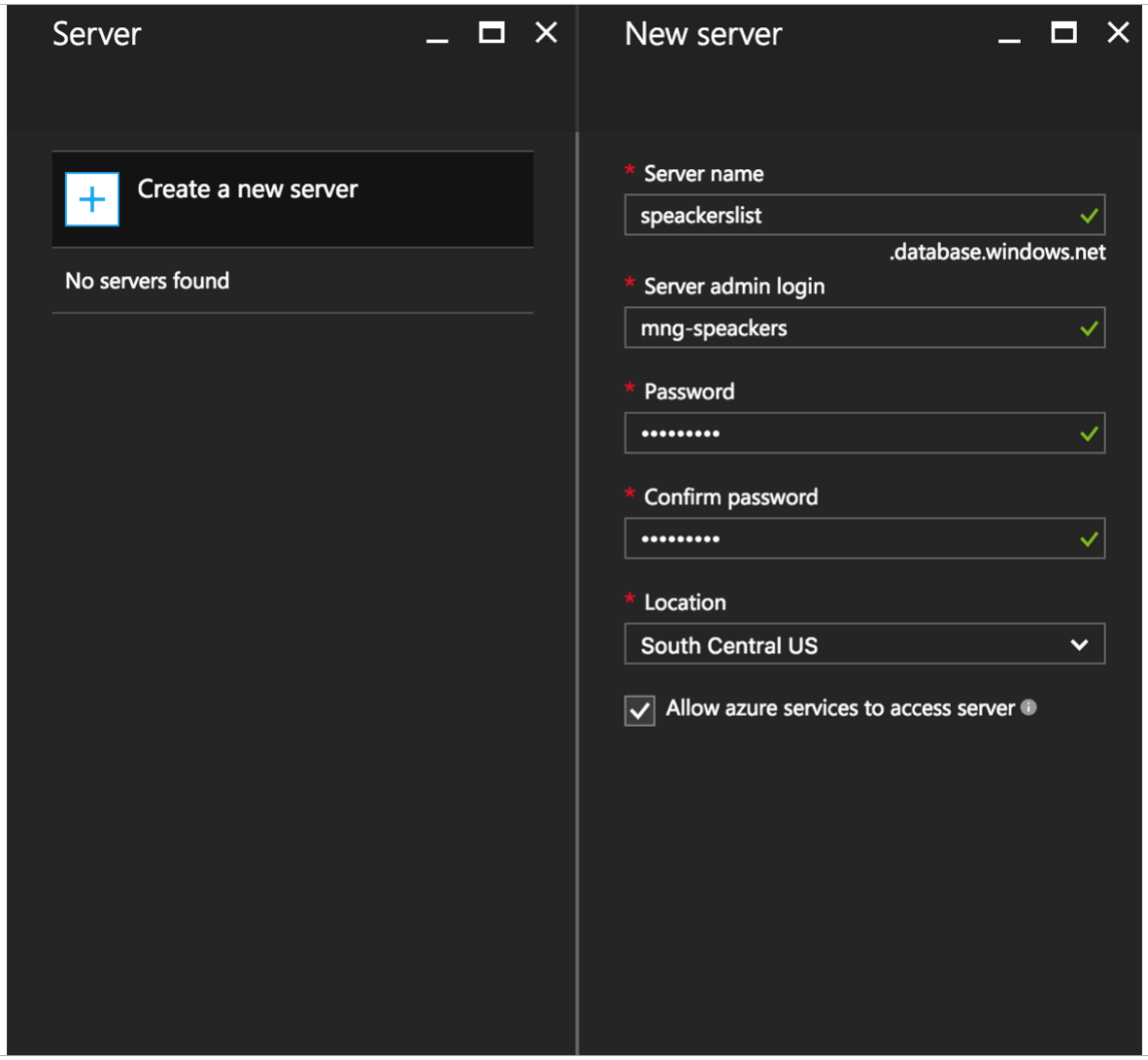
#DICA: No item Pricing Tier, busque pela opção de server gratuito. Ele oferece 32MB de espaço e já é suficiente para começar a desenvolver e testar sua app.



Definição de servidor

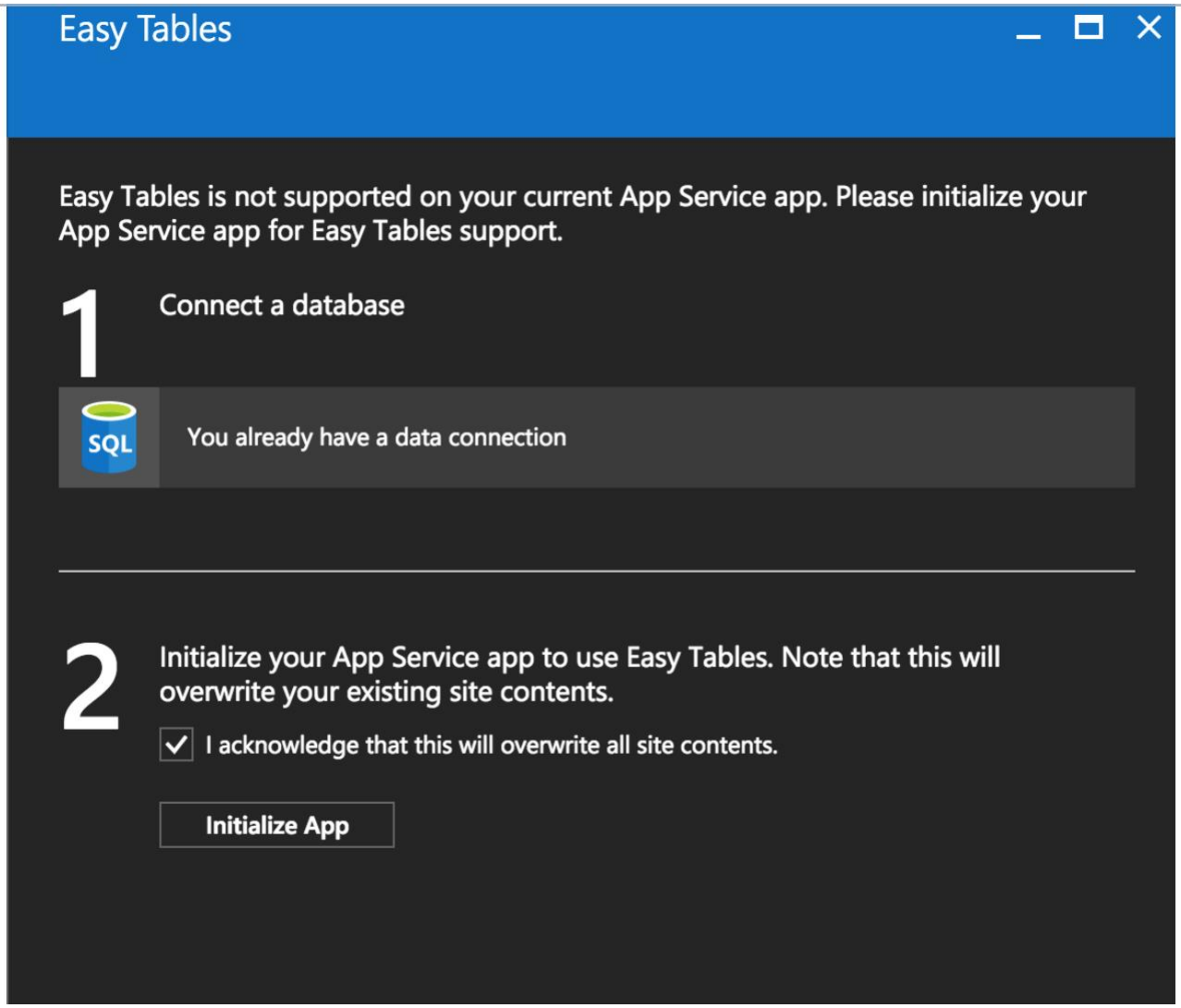
Para criar o servidor, clicamos em `Create new server`, definimos o nome, usuário e senha de acesso e a localização.

#DICA: Para economizar, defina para o seu servidor a mesma região do seu App Service.



Inicialização do Aplicativo

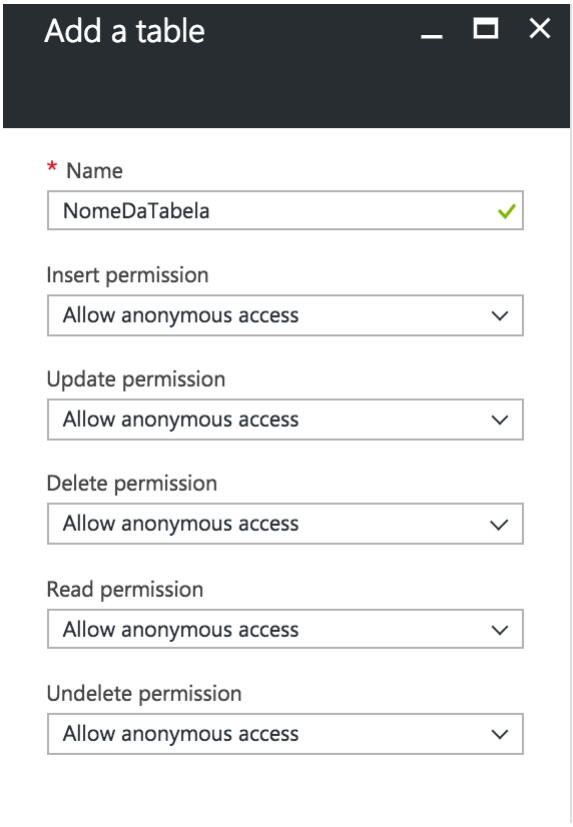
Depois de criar a conexão de dados e o servidor, em alguns instantes o Microsoft Azure fará o deploy do serviço e o Easy Tables já os identificará como ativos. Com isso, basta você inicializar o App, clicando no botão `Initialize App`. Feito isso, já iremos para última etapa dentro do Microsoft Azure, que é criar as nossas tabelas de dados.



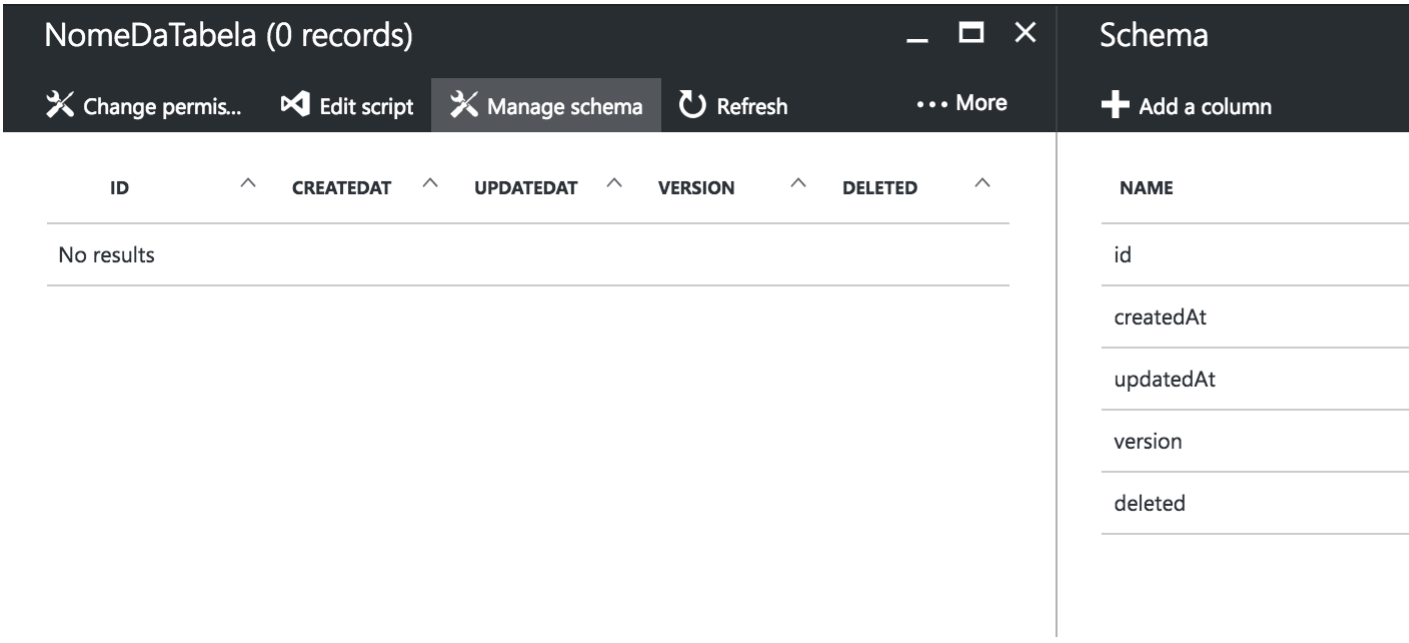
Criando o back-end com Easy Tables

O Easy Tables facilita a criação de tabelas em um banco de dados SQL no Azure e o acesso aos dados pode ser feito via REST ou utilizando uma SDK. Existem SDK disponíveis para Cordova, Xamarin, bem como para linguagens nativas, como Objective-C e Swift. Inclusive, algumas SDKs ainda oferecem suporte à sincronização off-line.

Para criar a tabela, basta clicar no botão de "+" e preencher o nome da tabela e os níveis de permissão aos métodos de insert, update, delete e etc.

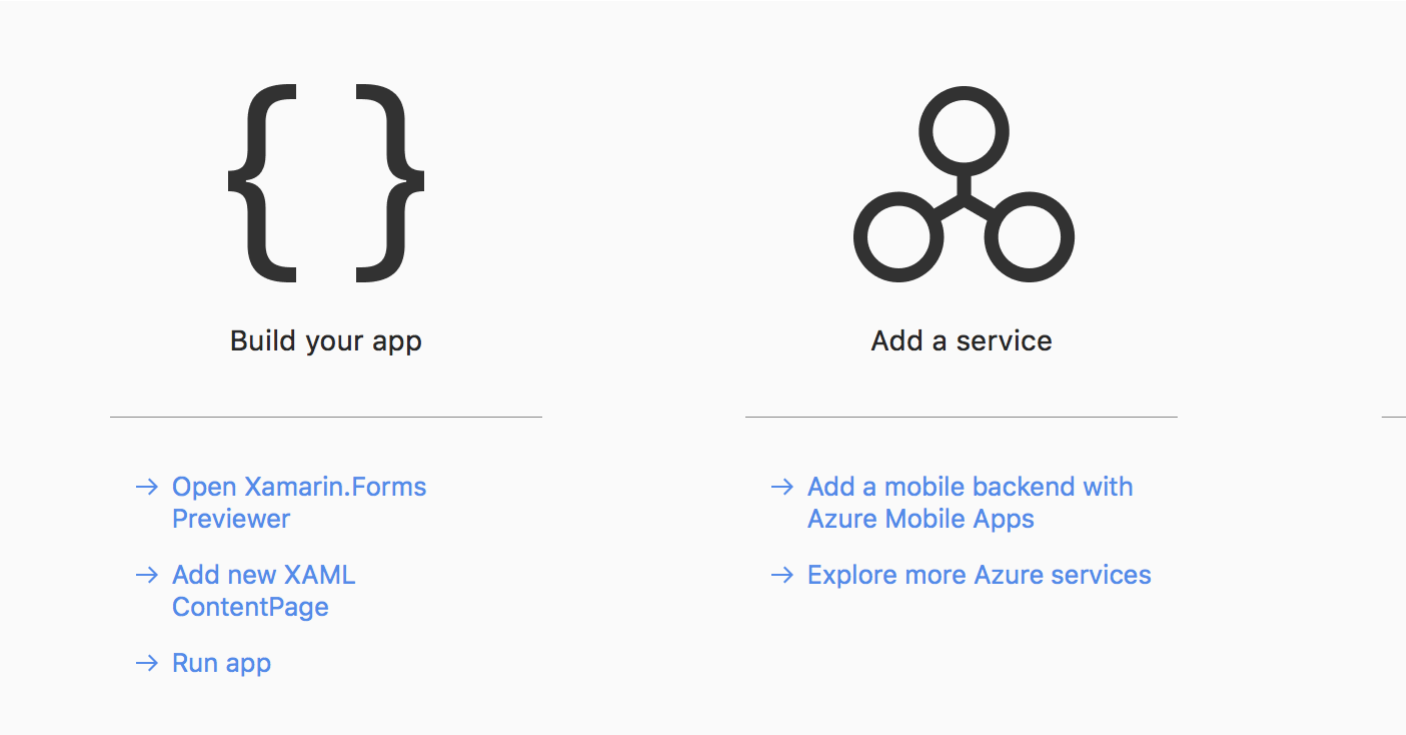


Com a tabela criada, clicamos em `Manage schema > Add a column`. Note que já temos algumas colunas na tabela, que são populadas dinamicamente durante as utilizações da API.

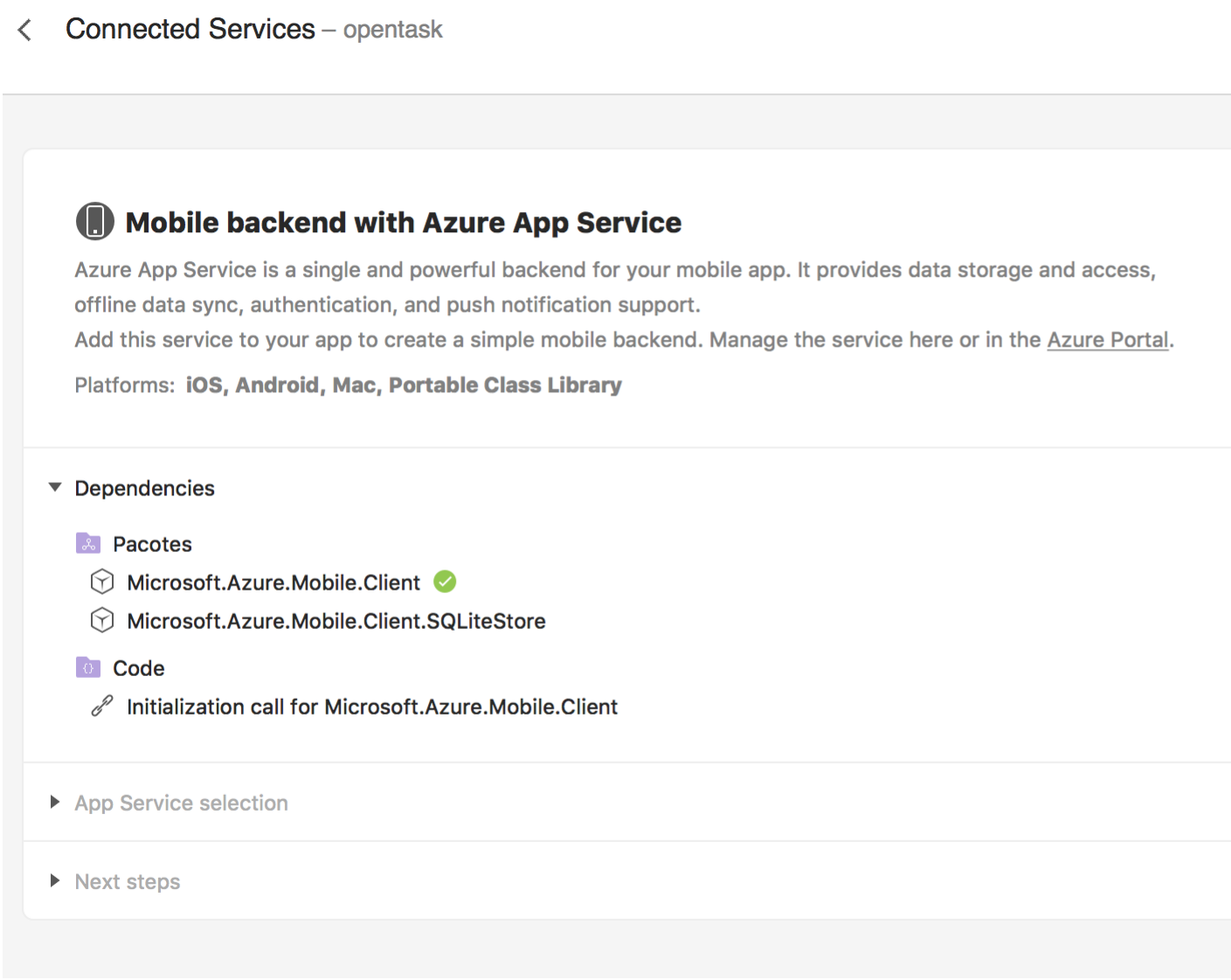


Overview do código

Abrindo o projeto no Visual Studio, acesse `Getting Started` e clique na opção de `Add a mobile backend with Azure Mobile Apps`.



Na sequência, podemos instalar as dependências para comunicação com o back-end na nuvem.



Constants.cs

Neste código, adicionamos a url do nosso App Service no Microsoft Azure, que é o mesmo que você visualiza no dashboard inicial do app.

```
namespace opentask
{
	public static class Constants
	{
		// Substitua pelo nome do seu app service .azurewebsites.net
		public static string ApplicationURL = @"URL DO APP SERVICE";
	}
}
```

TodoItem.cs

Agora é hora de criar o modelo de dados que usaremos localmente para exibir informações, mas também salvar no nosso back-end na nuvem. Ele deve ter o mesmo nome da tabela que criamos

anteriormente no portal do Azure. No trecho `JsonProperty`, indicamos qual é o campo lá na base de dados.

```
public class TodoItem
{
    string id;
    string name;
    bool done;

    [JsonProperty(PropertyName = "id")]
    public string Id
    {
        get { return id; }
        set { id = value;}
    }

    [JsonProperty(PropertyName = "text")]
    public string Name
    {
        get { return name; }
        set { name = value;}
    }

    [JsonProperty(PropertyName = "complete")]
    public bool Done
    {
        get { return done; }
        set { done = value;}
    }

    [Version]
    public string Version { get; set; }
}
```

TodoItemManager.cs

Nesta classe, fazemos todo gerenciamento da aplicação, como chamada para o App Service e salvamento de dados.

```
        private TodoItemManager()
        {
            this.client = new MobileServiceClient(Constants.ApplicationURL);

#if OFFLINE_SYNC_ENABLED
            var store = new MobileServiceSQLiteStore(offlineDbPath);
            store.DefineTable<TodoItem>();

            //Initializes the SyncContext using the default IMobileServiceSyncHandler.
            this.client.SyncContext.InitializeAsync(store);

            this.todoTable = client.GetSyncTable<TodoItem>();
#else
            this.todoTable = client.GetTable<TodoItem>();
#endif
        }

        .....

        public async Task SaveTaskAsync(TodoItem item)
        {
```

```
        if (item.Id == null)
        {
            await todoTable.InsertAsync(item);
        }
        else
        {
            await todoTable.UpdateAsync(item);
        }
    }
```

Treinamento gratuito de Xamarin

Quer aprender mais sobre desenvolvimento mobile cross-platform? Acesse nosso portal OSS Mobile e assista aos treinamentos técnicos e fique pode dentro das novidades sobre a plataforma.

Resumo

Neste laboratório, você criou uma conexão Xamarin com Azure.

Veja o exercício completo aqui: https://github.com/cyz/demo-xamarin-azure