

AGENDA 12

Adicionando Páginas,
Classes em C#,
Biblioteca LINQ e
Secure Storage



GEEaD - Grupo de Estudos de Educação a Distância
Centro de Educação Tecnológica Paula Souza

GOVERNO DO ESTADO DE SÃO PAULO
EIXO TECNOLÓGICO DE INFORMAÇÃO E COMUNICAÇÃO
CURSO TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS
PROGRAMAÇÃO MOBILE I

Expediente

Autores:

TIAGO ANTONIO DA SILVA

KELLY CRISTIANE DE OLIVEIRA DAL POZZO

São Paulo – SP, 2024



Vamos falar sobre o GitHub?

O GitHub é uma plataforma amplamente utilizada para hospedagem de código-fonte, controle de versão e colaboração em projetos de desenvolvimento. Ele funciona em conjunto com o Git, um sistema de controle de versão que permite rastrear mudanças no código ao longo do tempo, facilitando o trabalho em equipe e a integração contínua. Com o GitHub, desenvolvedores podem criar, gerenciar e compartilhar projetos de forma eficiente, além de colaborar com outros por meio de ferramentas como issues, pull requests e GitHub Actions para automação de processos.

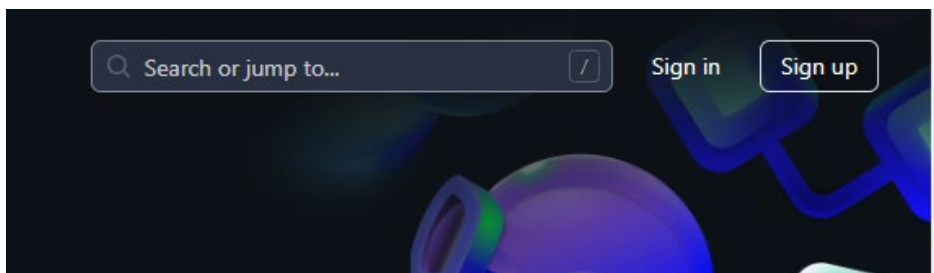
No Visual Studio, o GitHub está integrado, permitindo que desenvolvedores criem, versionem e publiquem seus projetos diretamente na plataforma. Projetos como os desenvolvidos com .NET MAUI podem ser facilmente sincronizados com repositórios do GitHub, oferecendo controle de versão, gerenciamento de branches e colaboração simplificada, tornando o desenvolvimento mais ágil e organizado.

Saiba mais sobre GitHub acessando sua documentação:
<https://docs.github.com/en/get-started>

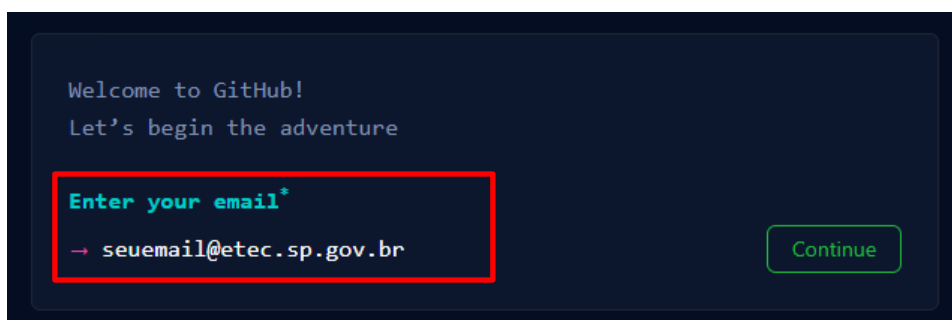


Passo a passo para acessar o GitHub

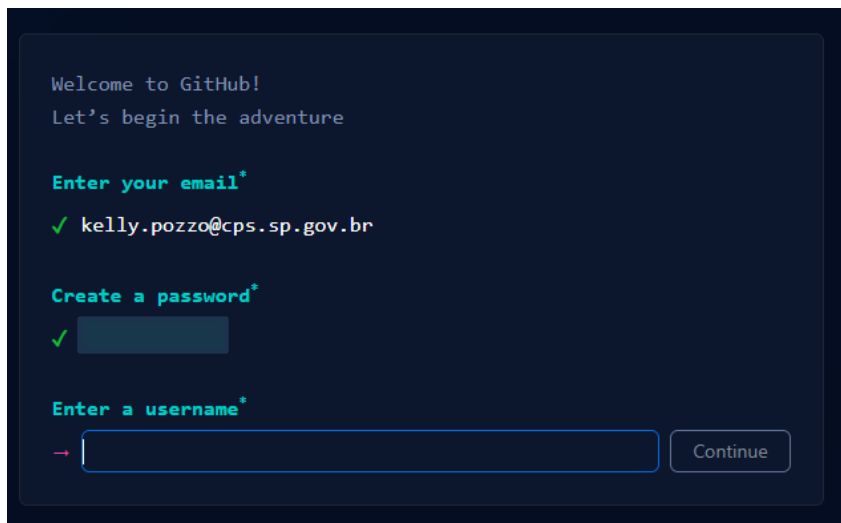
1. Acesse o site oficial do GitHub: <https://github.com>.
2. Na página inicial do GitHub, clique no botão **Sign up** (ou "Cadastrar-se").



3. No campo de cadastro, insira seu endereço de email e clique em **Continue**.



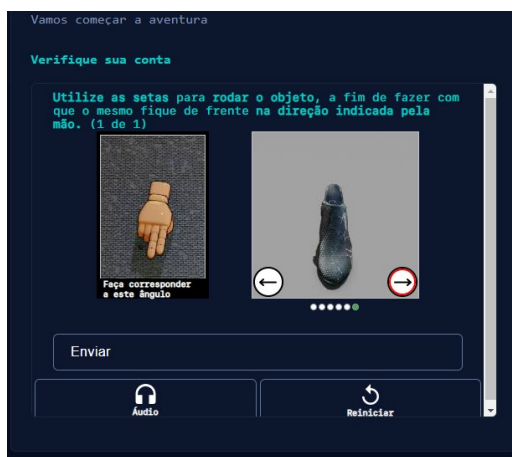
4. Crie uma senha segura para sua conta e clique em **Continue**.
5. Escolha um nome de usuário que será associado à sua conta no GitHub. Esse nome precisa ser único e aparecerá na URL do seu perfil (ex: github.com/seu-nome-usuario).



Para definição do username:

Permitido	Proibido
<ul style="list-style-type: none">• Letras (A-Z, a-z): letras maiúsculas ou minúsculas.• Números (0-9): Números podem ser usados no nome de usuário, desde que não seja o primeiro caractere.• Hífen (-): Pode ser usado para separar palavras. Exemplo, joao-silva.• Ponto (.): Pode ser usado para separar palavras. Exemplo, joao.silva.	<ul style="list-style-type: none">• Espaços: Não são permitidos.• Outros Caracteres Especiais (@, !, #, \$, %, &, *, etc.): Caracteres especiais não podem ser usados no nome de usuário.

6. O GitHub pode perguntar sobre suas preferências, como se você deseja receber notificações por email. Responda de acordo com suas preferências e clique em **Continue**.
7. Para garantir que você não é um robô, o GitHub pode solicitar que você complete uma verificação de captcha. Siga as instruções.



8. O GitHub enviará um email de verificação para o endereço de email fornecido. Acesse sua caixa de entrada, abra o email e clique no link para verificar sua conta.
9. Você agora está registrado no GitHub e pode começar a criar repositórios, colaborar em projetos e explorar a plataforma.

Durante as aulas, o GitHub será utilizado para armazenar e compartilhar os projetos desenvolvidos em .NET MAUI. O GitHub facilitará o acompanhamento da evolução dos projetos permitindo o compartilhamento de forma organizada e acessível. Entretanto, se você não tiver o GitHub instalado ou preferir não utilizá-lo, também será possível realizar as atividades normalmente armazenando os projetos localmente no seu computador.

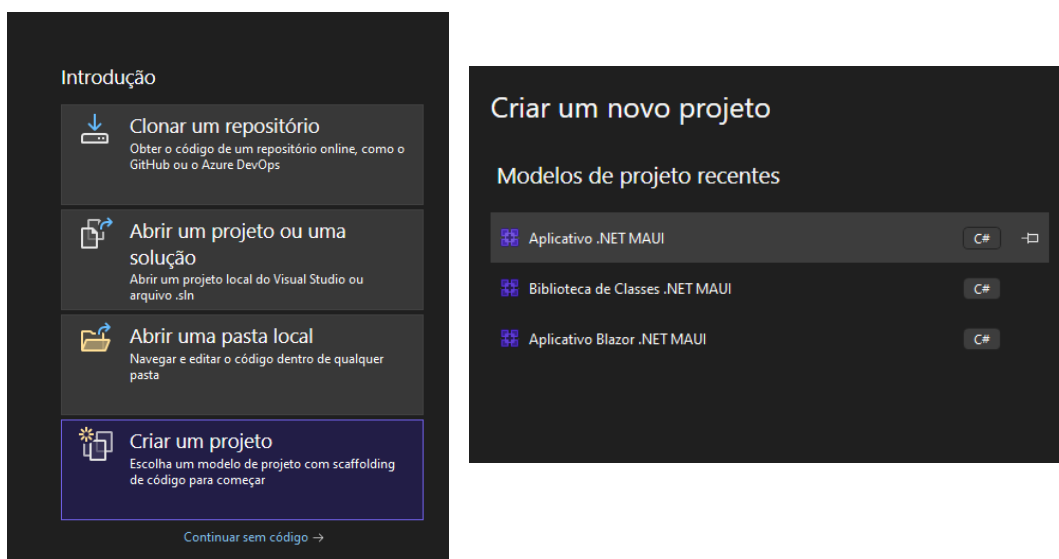


Navegar por telas em um aplicativo é uma prática comum e fundamental para garantir que o usuário consiga acessar de forma intuitiva e organizada todas as funcionalidades disponíveis. Em um aplicativo bem projetado, as telas funcionam como pontos de acesso para diferentes recursos. Uma navegação fluida entre telas ajuda a criar uma experiência de uso mais coerente, onde o usuário se sente orientado e encontra o que precisa de forma rápida e direta. Quando as telas estão bem estruturadas, o aplicativo se torna mais simples de utilizar, mesmo em casos de sistemas com grande volume de informações ou múltiplas funcionalidades. Por isso, garantir que a navegação entre telas seja clara e eficiente é essencial no desenvolvimento de aplicativos.

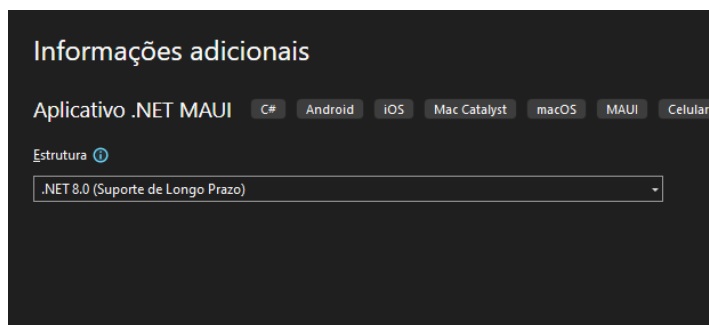
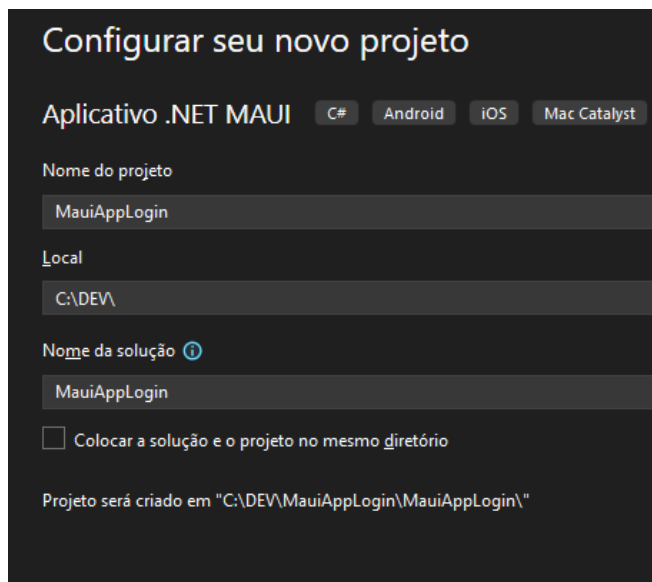
Nesta agenda, vamos desenvolver um aplicativo para autenticação de usuários, adicionando telas, navegando entre elas e enviando alterações para o GitHub. Para isso, vamos explorar recursos importantes do C# e da .NET MAUI, como a biblioteca LINQ e a API Secure Storage, além de incluir classes.

Vamos iniciar criando um novo projeto denominado MauiAppLogin.

Na tela inicial do VisualStudio, na tela Introdução selecione **Criar um Projeto**, escolha a opção Aplicativo .NET MAUI, clique no botão **Próximo**.



Na tela **Configurar seu Novo projeto**, adicione o nome do projeto e local para armazenamento, clique no botão **Próximo** e seguida no botão **Criar**.



Após carregamento do projeto já podemos testar a tela inicial clicando botão **Windows Machine**.

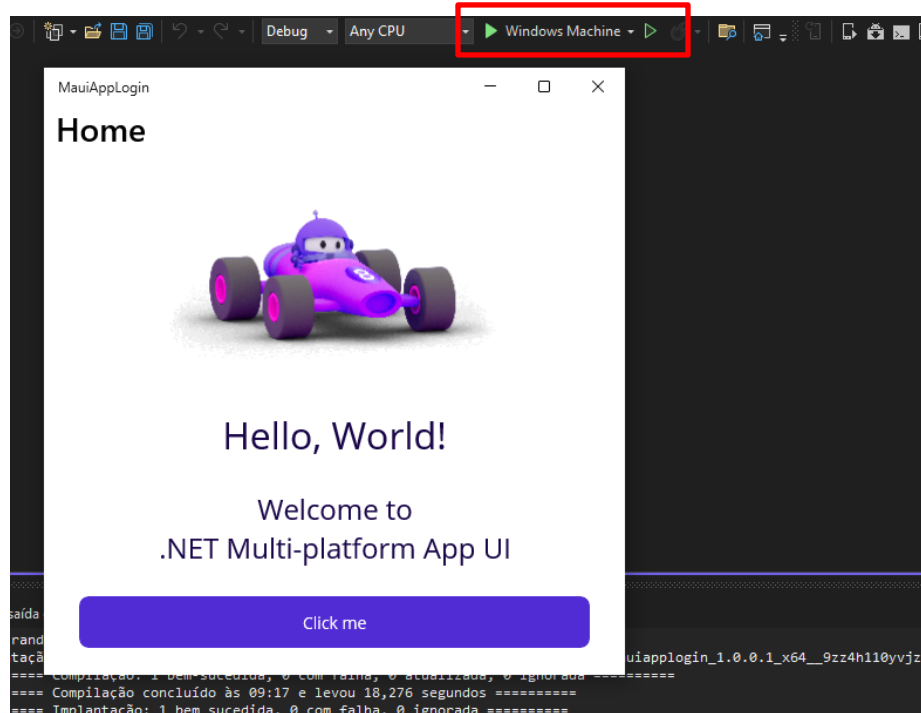
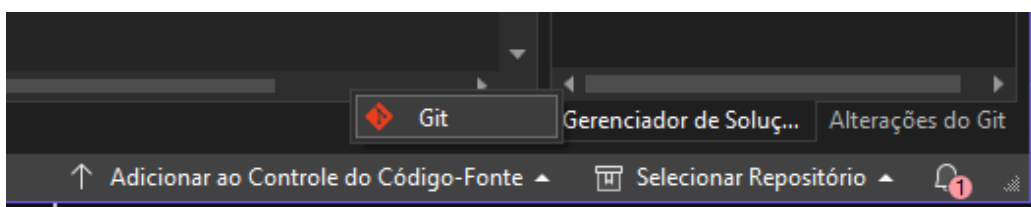
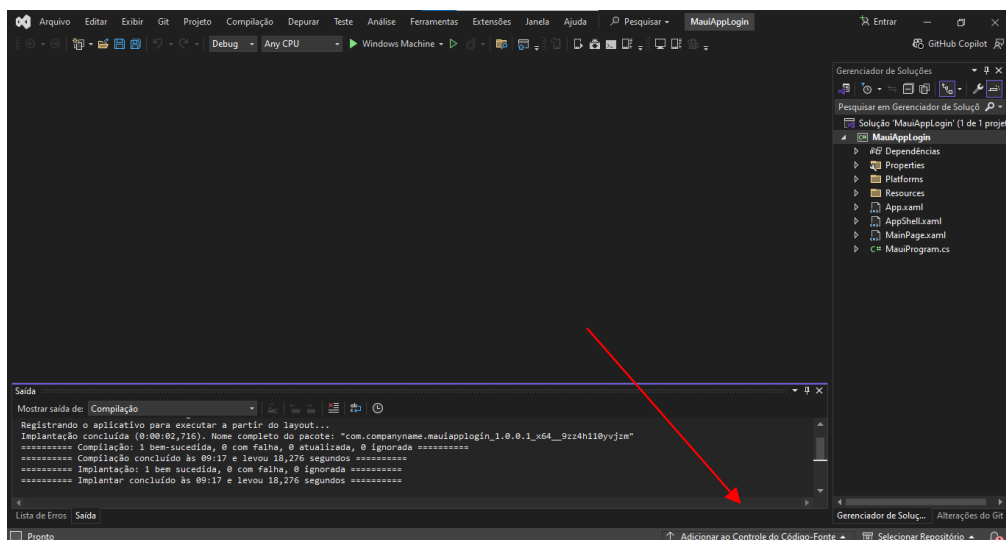


Figura 1 - Fonte: www.freepik.com

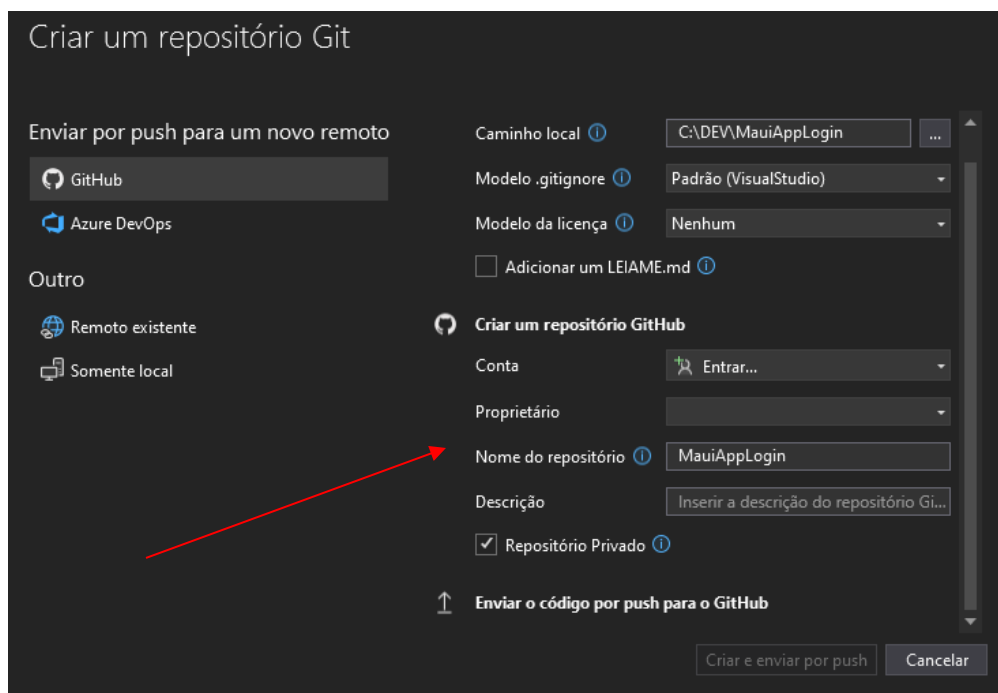
Adicionando a Codificação do Projeto no GitHub

Vamos hospedar o projeto recém criado no GitHub permitindo assim o controle de versão.

Para iniciar o armazenamento do projeto, selecione o botão **Adicionar ao Controle do Código-Fonte**, disponível no lado direito do rodapé da tela do VisualStudio, em seguida clique na opção **Git**.



Na tela **Criar um Repositório Git** caso ainda não tenha realizado a identificação, será necessário adicionar os dados da conta criada no GitHub.

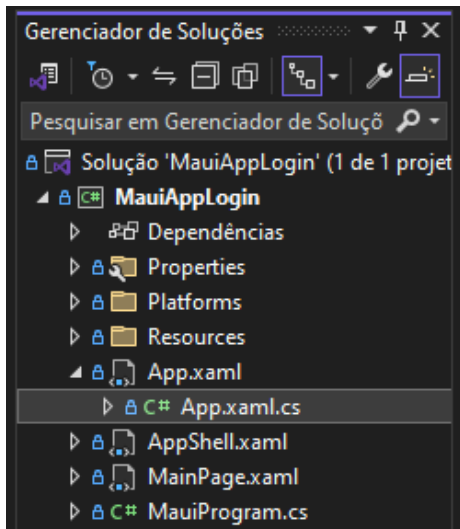


Para realizar a autenticação você será direcionado para o navegador.

Após autenticação concluída, clique no botão Criar e enviar por push.

A partir de agora seu repositório esta conectado ao projeto.

Modificando as dimensões de tela do projeto



Vamos iniciar o projeto modificando as dimensões da tela de exibição, para isso vamos modificar o arquivo **App.xaml.cs**, contido na pasta **App.xaml**.

Faça a sobrecarga do método **CreateWindow()** alterando o arquivo da seguinte forma:

```
namespace MauiAppLogin
{
    public partial class App : Application
    {
        public App()
        {
            InitializeComponent();

            MainPage = new AppShell();
        }
        protected override Window CreateWindow(IActivationState activationState)
        {
            var window = base.CreateWindow(activationState);

            window.Width = 400;
            window.Height = 600;

            return window;
        }
    }
} // Fecha classe
} // Fecha namespace
```

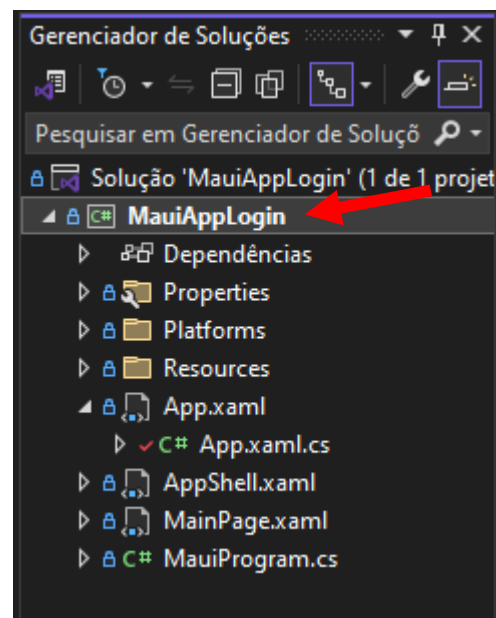
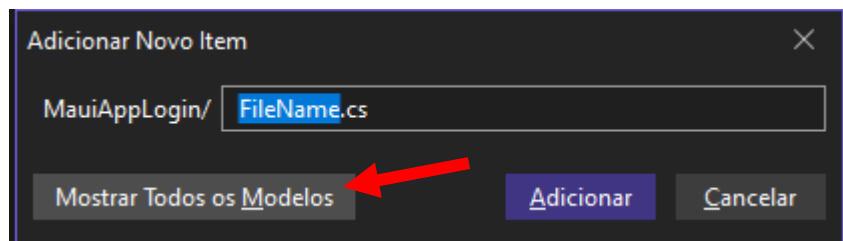
Adicionando um novo arquivo ao projeto

Adicionar um novo arquivo .xaml em um projeto é essencial para criar uma nova tela de interface de usuário de forma organizada e modular. O arquivo .xaml separa a estrutura visual da lógica do aplicativo, permitindo que a parte visual da tela seja desenvolvida de forma declarativa, enquanto o código de comportamento é gerenciado no arquivo code-behind correspondente, em C#.

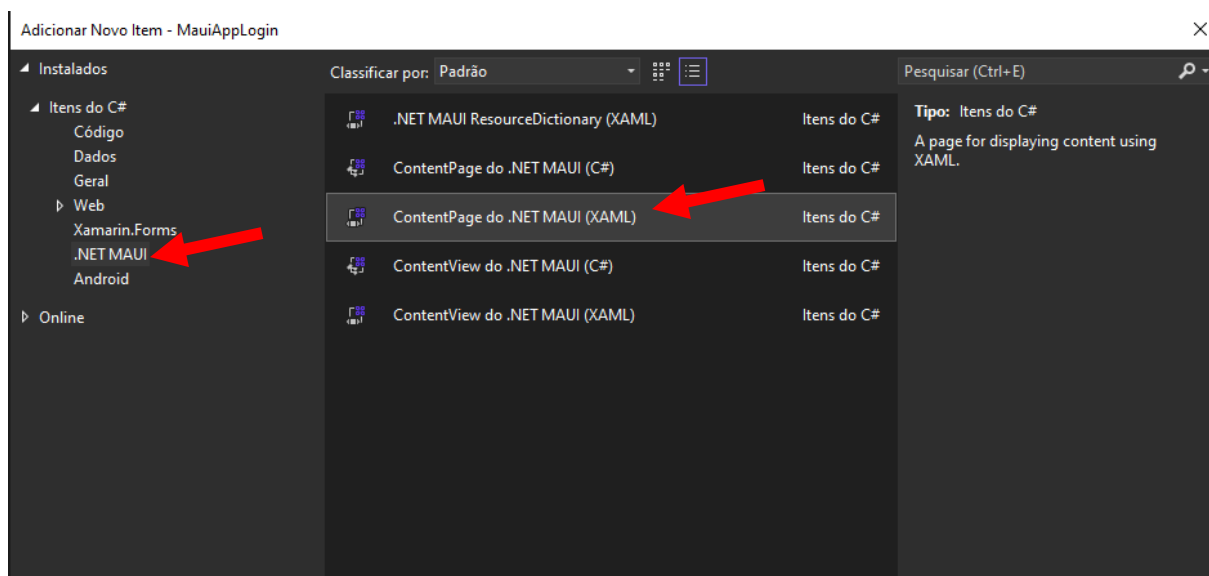
Para adicionar um novo arquivo, na área Gerenciador de Soluções, clique com o botão direito sobre o nome do projeto.

No menu exibido selecione o submenu **Adicionar**, em seguida **Novo Item**.

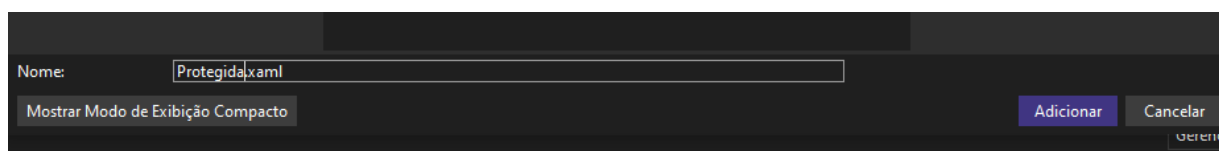
Na caixa de dialogo **Adicionar Novo Item** selecione o botão **Mostrar Todos os Modelos**.



Na nova caixa de dialogo exibida, selecione a opção **.NET MAUI**, em seguida selecione a opção **ContentPage do .NET MAUI (XAML)**.



Adicione um nome para nova tela e clique em **Adicionar**.



Utilize os mesmos procedimentos para adicionar mais uma tela, denominada Login.

Para acompanhar o desenvolvimento das telas a partir de agora, modifique o método principal do arquivo **App.xaml.cs** para carregar a tela Login, recém-criada.

```
public App()
{
    InitializeComponent();

    MainPage = new Login();
}
```

Vamos adicionar elementos para interação na tela de login, modifique o arquivo **Login.xaml** da seguinte forma:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MauiAppLogin.Login"
    Title="Login">
    <VerticalStackLayout
        VerticalOptions="Center"
        Spacing="10"
        Padding="10">

        <Label Text="Sistema de Login"
            HorizontalOptions="Center" />

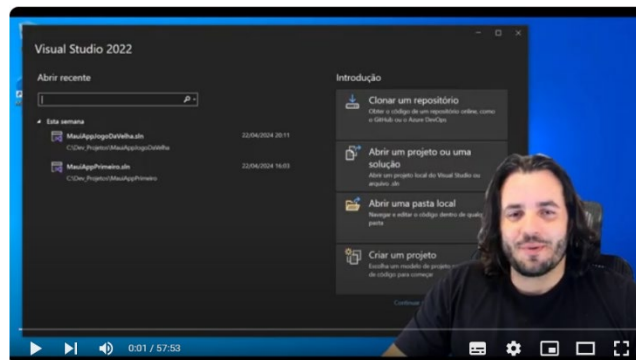
        <Label Text="Usuário:" />
        <Entry x:Name="txt_usuario" />

        <Label Text="Senha:" />
        <Entry x:Name="txt_senha" IsPassword="True" />

        <Button Text="Entrar"
            Clicked="Button_Clicked"
            HorizontalOptions="Center" />

    </VerticalStackLayout>
</ContentPage>
```

Assista a vídeo aula explicativa sobre o projeto:



Disponível em: <https://www.youtube.com/watch?v=ETH0eSCyvSE>

Vamos agora compreender alguns conceitos apresentados neste projeto.

LINQ (Language Integrated Query)

LINQ, ou Language Integrated Query, é uma característica do C# que permite que consultas sejam integradas diretamente no código, facilitando a manipulação e a consulta de dados em coleções de objetos. Com LINQ, é possível escrever consultas de forma declarativa, expressando o que deseja fazer, em vez de como fazê-lo.

Utilizando LINK é possível:

- ✓ **Consulta de Dados:** Com LINQ, é possível realizar consultas em coleções de objetos, como arrays, listas, dicionários e bancos de dados. Permitindo filtrar, ordenar, agrupar e projetar.
- ✓ **Sintaxe de Consulta e Sintaxe de Método:** LINQ oferece duas formas de escrever consultas: sintaxe de consulta e sintaxe de método. A sintaxe de consulta utiliza palavras-chave semelhantes ao SQL, enquanto a sintaxe de método utiliza métodos de extensão como Where, Select, OrderBy, entre outros.
- ✓ **Expressões Lambda:** Expressões lambda são frequentemente usadas em conjunto com LINQ para criar predicados e projeções de forma concisa, permitindo escrever consultas de forma mais compacta e legível.
- ✓ **Tipos de Operadores:** LINQ oferece uma variedade de operadores para manipular e consultar dados, incluindo operadores de projeção (Select), filtragem (Where), ordenação (OrderBy), agrupamento (GroupBy), junção (Join), entre outros.
- ✓ **Integração com Tipos Anônimos:** LINQ funciona bem com tipos anônimos, permitindo projetar e retornar resultados de consulta em novos tipos definidos inline.

Método Any

O método Any é usado para verificar se uma coleção contém algum elemento, ou se pelo menos um elemento satisfaz uma determinada condição. Sendo aplicável a qualquer coleção que implemente a interface IEnumerable, como arrays, listas, e outras coleções. Sendo um dos métodos de extensão LINQ mais utilizados em C#, o método Any é usado para determinar se uma coleção de elementos contém algum elemento que atende a uma determinada condição. O método retorna true se pelo menos um elemento na coleção satisfizer a condição especificada e false caso contrário.

O método Any pode ser utilizado em conjunto com outras operações LINQ, como filtragem (usando Where) ou projeção (usando Select), para determinar se a coleção resultante contém algum elemento. Ele pode ser chamado

com ou sem um predicado para determinar se a coleção contém algum elemento ou se algum elemento satisfaz uma condição específica.

É um método eficiente, pois ele para de iterar pela coleção assim que encontra o primeiro elemento que atende à condição especificada. Isso significa que ele não precisa percorrer toda a coleção se encontrar um elemento que satisfaça a condição.

Exemplos de Uso:

Verificando se a lista contém algum elemento:

```
List<int> numeros = new List<int> { 1, 2, 3, 4, 5 };  
bool temElemento = numeros.Any();
```

Verificar se uma lista contém algum número maior que 10:

```
bool temNumeroMaiorQue10 = numeros.Any(x => x > 10);
```

Retorno de Valor: O método Any retorna um valor booleano (true ou false) indicando se a coleção contém pelo menos um elemento que satisfaz a condição especificada.

Tratamento de Coleções Vazias: Se a coleção estiver vazia, o método Any retornará false, indicando que não há elementos na coleção.

O método Any é uma ferramenta útil para verificar se uma coleção contém elementos e para aplicar lógica condicional com base nesse resultado. Ele é comumente utilizado em situações onde você precisa determinar se uma coleção contém pelo menos um elemento antes de realizar outras operações.

ASYNC, AWAIT e TASK

Em C#, uma Task representa uma operação assíncrona que pode ser executada em segundo plano de forma concorrente com o thread principal do aplicativo. As Tasks são fundamentais para a programação assíncrona e são utilizadas em operações demoradas, como chamadas de rede, acesso a bancos de dados, cálculos intensivos e outras tarefas que não devem bloquear o fluxo principal do programa.

Para criar uma Task, existem diversas maneiras. Pode-se utilizar o construtor Task manualmente, ou métodos de fábrica como Task.Run e Task.Factory.StartNew. Além disso, a combinação com ASYNC e AWAIT facilita a criação de operações assíncronas de forma mais simples.

Uma Task pode estar em vários estados, como WaitingToRun, Running, RanToCompletion, Faulted, entre outros,

sendo possível verificar o estado por meio das propriedades `IsCompleted`, `IsFaulted` e `IsCanceled`.

Para aguardar a conclusão de uma `Task`, a palavra-chave `AWAIT` é utilizada em métodos assíncronos. Ela permite que o thread principal continue executando outras tarefas enquanto a operação assíncrona ainda está em andamento, sem bloquear o fluxo de execução.

Após a conclusão de uma `Task`, é possível encadear outras operações com métodos como `ContinueWith` ou diretamente com `AWAIT`, o que facilita a execução de ações subsequentes de forma lógica.

A cancelamento de `Tasks` também é possível utilizando o `CancellationTokenSource`, que permite interromper operações que não são mais necessárias. Isso é especialmente útil em operações demoradas que precisam ser interrompidas quando não são mais relevantes.

Tratamento de exceções em `Tasks` deve ser feito com cuidado, utilizando blocos `try-catch`, especialmente ao utilizar `AWAIT`, garantindo que erros sejam tratados de forma adequada.

A programação assíncrona em C# permite a criação de aplicativos mais responsivos e escaláveis. Ao dominar conceitos como a criação, o controle de estado, o aguardo e o cancelamento de `Tasks`, você pode desenvolver aplicativos mais eficientes. Essa abordagem é particularmente importante para operações de entrada e saída (I/O), como acesso a bancos de dados e chamadas de rede, onde a espera pode ser demorada. Usar a palavra-chave `ASYNC` permite que o método seja executado de forma assíncrona, enquanto `AWAIT` indica que o programa deve aguardar a conclusão de uma operação antes de prosseguir. O .NET gerencia essas operações assíncronas por meio de um pool de threads, otimizando o uso de recursos. O tratamento de exceções em operações assíncronas é essencial para garantir que os erros sejam capturados e gerenciados de forma adequada, o que é crucial para manter a robustez e escalabilidade do código.

Secure Storage

O `Secure Storage` refere-se à capacidade de armazenar dados sensíveis de forma segura em um dispositivo, como senhas, chaves de acesso, tokens de autenticação e outras informações confidenciais. Sendo importante para proteger os dados do usuário contra acessos não autorizados ou exploração por parte de aplicativos maliciosos.

Algumas das opções disponíveis para o `Secure Storage` em .NET MAUI:

1. Secure Storage API: .NET MAUI oferece uma API para armazenamento seguro que permite que você armazene e recupere dados sensíveis de forma segura no dispositivo do usuário. Essa API geralmente oferece métodos simples para armazenar e recuperar dados de forma criptografada, garantindo que eles não sejam acessíveis a aplicativos não autorizados.

2. Keychain (iOS) e KeyStore (Android): Em dispositivos iOS, o Keychain é uma área segura onde você pode armazenar chaves, senhas e outros dados sensíveis. Em dispositivos Android, o KeyStore é uma funcionalidade semelhante que oferece um local seguro para armazenamento de chaves e segredos. Ambos os sistemas operacionais oferecem APIs para acessar e gerenciar esses armazenamentos seguros.

3. Plataforma de Segurança Integrada: Além das APIs específicas para armazenamento seguro, muitas plataformas móveis oferecem recursos integrados de segurança, como armazenamento criptografado, autenticação biométrica e proteção contra-ataques de força bruta. É importante aproveitar esses recursos integrados para garantir a segurança dos dados do usuário.

4. Boas Práticas de Segurança: Além de usar as APIs e recursos de segurança oferecidos pelo .NET MAUI e pelas plataformas móveis, é importante seguir as boas práticas de segurança ao lidar com dados sensíveis. Isso inclui usar técnicas de criptografia robustas, proteger os dados em repouso e em trânsito, e implementar autenticação e autorização adequadas em seu aplicativo.

Ao desenvolver um aplicativo que lida com dados sensíveis, é fundamental priorizar a segurança e proteger esses dados contra acessos não autorizados. Utilizando as APIs de Secure Storage e aproveitando os recursos de segurança integrados nas plataformas móveis, você pode garantir que os dados do usuário estejam protegidos contra ameaças de segurança.

Elemento Frame

O Frame no .NET MAUI é uma ferramenta valiosa para organizar e destacar o conteúdo, conferindo à interface uma aparência moderna e sofisticada, além de agregar uma sensação de profundidade e separação visual entre os elementos da aplicação. No .NET MAUI, o elemento Frame funciona como um contêiner que exibe uma página ou conteúdo envolto em uma borda com sombra, proporcionando uma sensação visual de profundidade e distinção em relação ao restante da interface do usuário. Seu uso oferece várias vantagens práticas no desenvolvimento de interfaces.

O Frame é eficaz na organização e apresentação do conteúdo em um aplicativo, fornecendo uma borda ao redor do conteúdo, o que ajuda a destacá-lo dos demais elementos visuais da interface. Essa característica contribui para uma organização mais clara e eficiente do layout.

Em termos de design, o Frame é frequentemente utilizado em interfaces com estilo moderno, conferindo uma aparência elegante e sofisticada ao aplicativo. É possível personalizar o elemento com cores, sombras e bordas, permitindo que ele se alinhe ao estilo visual adotado no projeto. Além disso, adiciona uma sombra sutil ao redor de seu conteúdo, criando um efeito de profundidade que realça os elementos principais da interface. Isso é

particularmente útil para destacar seções importantes da tela de maneira visualmente agradável.

O Frame também se adapta de forma flexível ao layout, sendo capaz de conter desde uma única página até um conjunto de controles, conforme a necessidade do aplicativo. Sua versatilidade o torna útil para diferentes cenários de design e layout.

Outro ponto relevante é a ampla capacidade de personalização visual que o Frame oferece. Propriedades como cor da borda, cor da sombra, espessura da borda e raio dos cantos podem ser ajustadas para atender a requisitos de design específicos, garantindo uma aparência personalizada e harmônica.

Além disso, o Frame é compatível com todas as plataformas suportadas pelo .NET MAUI, incluindo iOS, Android, Windows e macOS, proporcionando uma experiência consistente em diversos dispositivos e ambientes operacionais.

CÓDIGO-FONTE COMPLETO

<https://github.com/tiagotas/MauiAppLogin>