|  |  |  |  |
| --- | --- | --- | --- |
|  | MODALIDADE: EDUCAÇÃO PROFISSONAL | | |
| CURSO: TÉCNICO EM INFORMÁTICA  SÉRIE/MÓDULO/PERÍODO: TI3AN | | |
| COMPONENTE CURRICULAR:  Desenvolvimento de Software | Professor (a): Fabrício Passos | |
| Atividade prática em sala | | Data:dd/mm/yyyy |

**PASSOS**

* Criar o projeto MVC
* Botao direito no projeto e Gerenciar extensões do Nuget e instalar



* Outro framework



* Criar na pasta model a classe Aviao



* Criar classe Contexto na pasta Model



* Inserir a propriedade ConnectionStrings no appsettings.json

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"ConexaoSQLite" : "Data Source=Avioes.db"

}

}



* Alterar a classe Program.cs



* Abrir o console de pacotes do nugget e executar os comandos add-migration criacaoBDSqLite

update-database

* Confirmar a criação do BD no projeto



* Criar na Controllers a Action AvioesController.cs

public async Task<IActionResult> Index()

{

return View(await \_contexto.Avioes.ToListAsync());

}

[HttpGet]

public IActionResult NovoAviao() {

return View();

}

//Ação que de fato irá criar o registro no banco de dados de fato

[HttpPost]

public async Task<IActionResult> NovoAviao (Aviao aviao)

{

await \_contexto.Avioes.AddAsync(aviao);

await \_contexto.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

[HttpGet]

public async Task<IActionResult> AtualizarAviao(int idAviao)

{

Aviao aviao = await \_contexto.Avioes.FindAsync(idAviao);

return View(aviao);

* }
* Criar a View Index (Read para listar) da AviaoController



Selecionar o template Exibição do Razor



Selecionar o modelo de lista

* Criar a View de NovoAvião (Create)



* Ajustar a “asp-action" na view Index (read). Referenciar a Action criada e a classe CSS de Botão



* Ajustar as Actions do CRUD na view Index
* Remover a Action de Details, pois vamos retornar a lista sempre na Index
* Alterar a asp-action de Update
* Transformar a Action “Delete” em um formulário para submetermos ao servidor o registro a ser excluído. Lembrando que enviamos o Id (chave) do registro que deverá ser removido.

//Criando a Action de Delete. Neste caso apenas HttpPost, pois apenas redirecionamos.

[HttpPost] public async Task<IActionResult> ExcluirAviao (int idAviao)

{

Aviao aviao = await \_contexto.Avioes.FindAsync (idAviao);

\_contexto.Avioes.Remove(aviao);

await \_contexto.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

* Agora vamos rodar o projeto. Ao executar a action de Edit vamos ter um erro, pois ainda não criamos a ACTION. Neste caso eu redireciono para onde?

ENTAO BORA CRIAR A ACTION AtualizarAviao

* Primeiro vamos ajustar na index o parâmetro de idAviao que será passado como parâmetro. Obs: Tem que ser o mesmo nome de parâmetro definido como parâmetro na Action AtualizaAviao. No caso idAviao
* Criar o código da Action AtualizarAviao na AvioesController

[HttpGet]

public async Task<IActionResult> AtualizarAviao(int idAviao)

{

Aviao aviao = await \_contexto.Avioes.FindAsync(idAviao);

return View(aviao);

}

[HttpPost]

public async Task<IActionResult> AtualizarAviao (Aviao aviao)

{

\_contexto.Avioes.Update(aviao);

await \_contexto.SaveChangesAsync();

return RedirectToAction(nameof(Index));

* }
* Criar a View de Atualização



Rodar o projeto e mostrar o processo de atualização funcionado

* Criar a nossa Action de delete na AvioesController
* Baixar o DBBrowser SQ Lite. Abaixo link para download. Pegar a versão sem instalação <https://sqlitebrowser.org/dl/>
* Abrir o banco criado na aplicação



* Abrir a opção executar SQL e realizar um select