

# Transporte dos Guri

## Sistema Gerenciador de Transporte Aéreo

Alunos

Thiago Wurster Balbinot | Thiago Thomasi | Rhyan Mezaroba | Mateus Ferreira da Silva

# Introdução

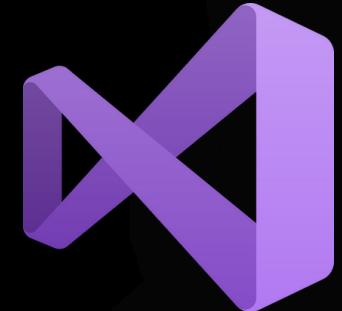
Nosso Sistema possui a finalidade de gerenciar uma empresa aérea no que diz respeito a disponibilidade de Aeronaves, Aeroportos, Usuários, Assentos, Compras, Escalas, Reservas, Viagens, Voos e Assentos disponíveis nos Voos.

O Sistema Transportes dos Guri possui uma divisão clara de papéis de usuários, de forma que dependendo do grupo que certo usuário faz parte, não poderá acessar certa página.

# Tecnologias Utilizadas

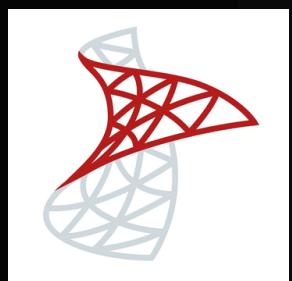
## IDE

Microsoft Visual Studio



## BACK-END

C#  
Entity-Framework  
SQL Server



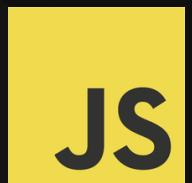
## FRAMEWORK

.NET 9  
Projeto ASP .NET Core MVC



## FRONT-END

CSHTML  
CSS TAILWIND  
JAVASCRIPT



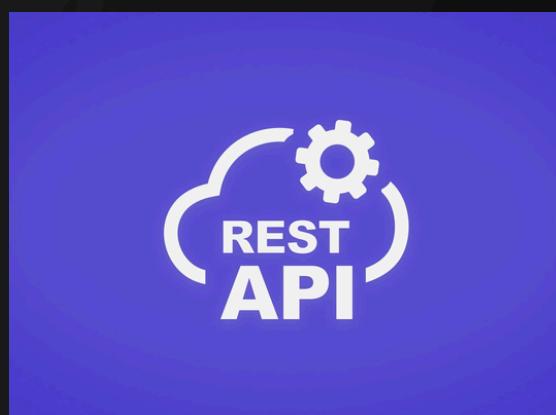
## COMPONENTES E BIBLIOTECAS .NET

Autenticação por conta Individual - IDENTITY  
Scaffolding  
Geração de PDF - QUESTPDF



## CONSUMO DE API

API RestFULL para procura de CEP



# Organização das Pastas

## Projeto: TransporteAereo



wwwroot



Areas



Controllers



Data



Migrations



Models



Repository



PDF



ViewModels



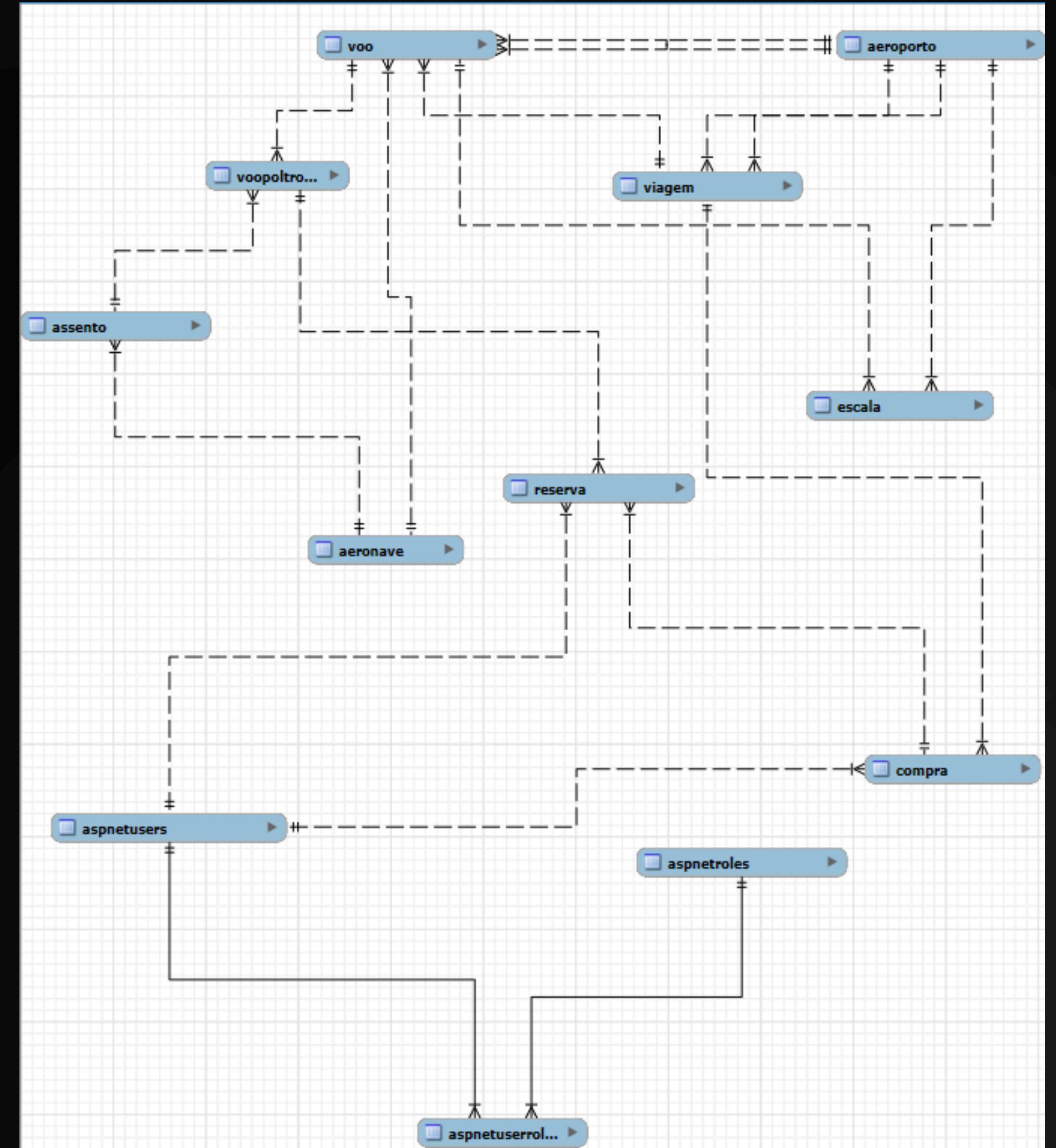
Views



Program.cs

# Modelos da Aplicação





# ApplicationDbContext

Instanciação de todas os Modelos principais: Aeronave, Aeroporto, Assento, Escala, Reserva, Voo, VooPoltrona, Viagem e Compra.

Mapeamento dos relacionamentos entre as entidades com Entity Framework e LINQ, a partir de métodos como .HasOne, .WithMany, .HasForeignKey e .OnDelete.

Ex.:

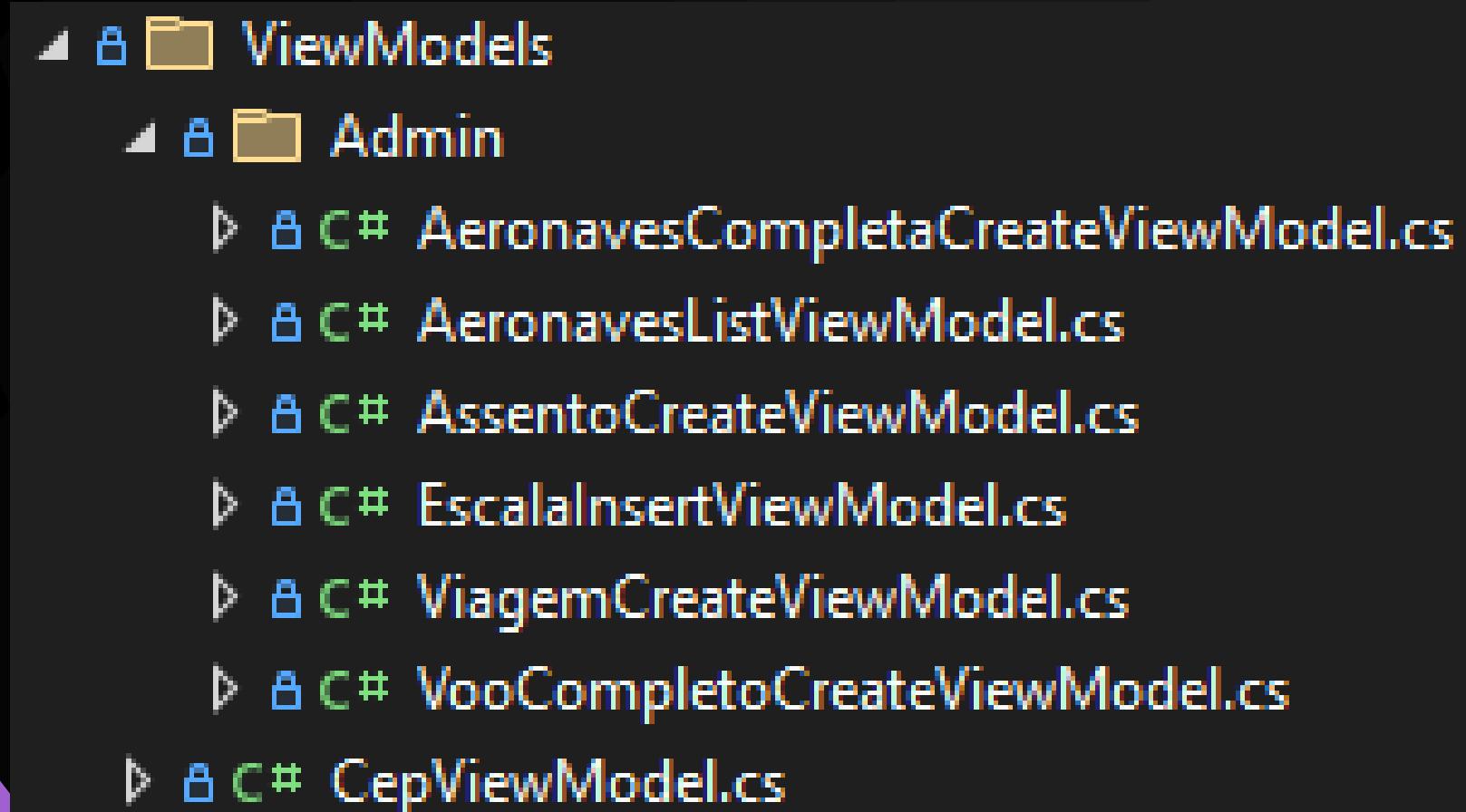
```
modelBuilder.Entity<Compra>()
    .HasOne(c => c.Cliente)
    .WithMany()
    .HasForeignKey(c => c.IdCliente)
    .OnDelete(DeleteBehavior.Restrict);
```

UMA compra tem UM cliente  
UM cliente pode ter VÁRIAS compras  
CHAVE ESTRANGEIRA em compra  
NÃO deixa excluir cliente caso tenha compras

# ViewModels

Servem como molde para criar modelos de Views específicas, de modo em que utilize somente alguns atributos de determinada Classe, não sendo mapeado ao Banco de Dados.

Em nosso caso dividimos o ViewModel entre o que usuários administradores podem acessar do que usuários comuns podem acessar.



O que está na pasta Admin de ViewModels serve somente para acesso de administrador, os quais são moldes para as Views de criação de Objetos.

O CepViewModel é acessível a todos os usuários pois é o molde para as informações que serão retornadas via consumo de API Rest.

# Identity

Forma que o .NET orquestra autenticação de contas, com Registro e Login de usuários.

Em nosso sistema utilizamos a configuração de autenticação por conta individual.

Mesmo com o Identity já gerando atributos suficientes para uma autenticação robusta ao registrar usuário, criamos um novo Model com o nome de ApplicationUser, o qual estende o Identity, aí acrescentamos novos atributos.

Além disso, fizemos a criação de Roles (Papéis) o qual define se um usuário é administrador ou não, ação que impacta no acesso a certos conteúdos do sistema.

- ▶  `dbo.AspNetRoleClaims`
- ▶  `dbo.AspNetRoles`
- ▶  `dbo.AspNetUserClaims`
- ▶  `dbo.AspNetUserLogins`

- ▶  `dbo.AspNetUserRoles`
- ▶  `dbo.AspNetUsers`
- ▶  `dbo.AspNetUserTokens`

# API REST

Tipo de API mais consumida em um ambiente .NET, tem o nome de REST pois é um Estado Representacional de Transferência (Representational State Transfer).

Esse tipo de API permite a comunicação entre sistemas via HTTP.

Facilita muito a integração de serviços, onde em nosso caso, o consumo da API REST tem o intuito de após o usuário registrar sua conta preenchendo seu CEP o nosso sistema entra em contato com o banco de dados/sistema do ViaCEP, o qual possui as informações de endereço completa vinculado a cada CEP, aí o nosso sistema preenche as informações de endereço automaticamente na View de Registro do Identity.



# CSS: Tailwind x Bootstrap

Enquanto o Bootstrap é um framework CSS que fornece elementos já pre-definidos/prontos, como botões, barras de navegação, ...

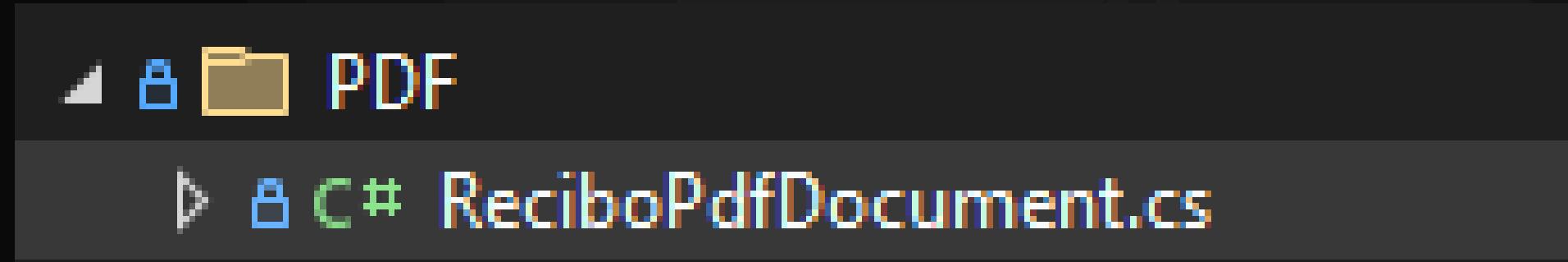
Tailwind é um framework muito mais baseado em utilidade, fornecendo uma vasta coleção de classes de baixo nível, ou seja, você cria design totalmente personalizados do zero, muito mais próximo de um CSS puro.

Fomos pelo caminho do Tailwind justamente por conta dessa alta gama de personalização, sendo mais eficiente que um CSS puro, e possuindo mais desempenho em comparação que o Bootstrap, pois ele remove as classes de design não utilizadas no processo de complacão da aplicação.

# Geração de PDF

Em nosso aplicação, para facilitar a geração de PDF ao mesmo tempo que personalizamos ele para manter um padrão, utilizamos a biblioteca QuestPDF em nosso projeto .NET.

Essa biblioteca open-source, faz a geração do PDF do recibo da passagem que o usuário acabou de comprar em nosso sistema de Transporte Aéreo, baixando na máquina local caso o usuário deseje.



## Banco de Dados II

# Bora para o Código...

Alunos

Thiago Wurster Balbinot | Thiago Thomasi | Rhyan Mezaroba | Mateus Ferreira da Silva