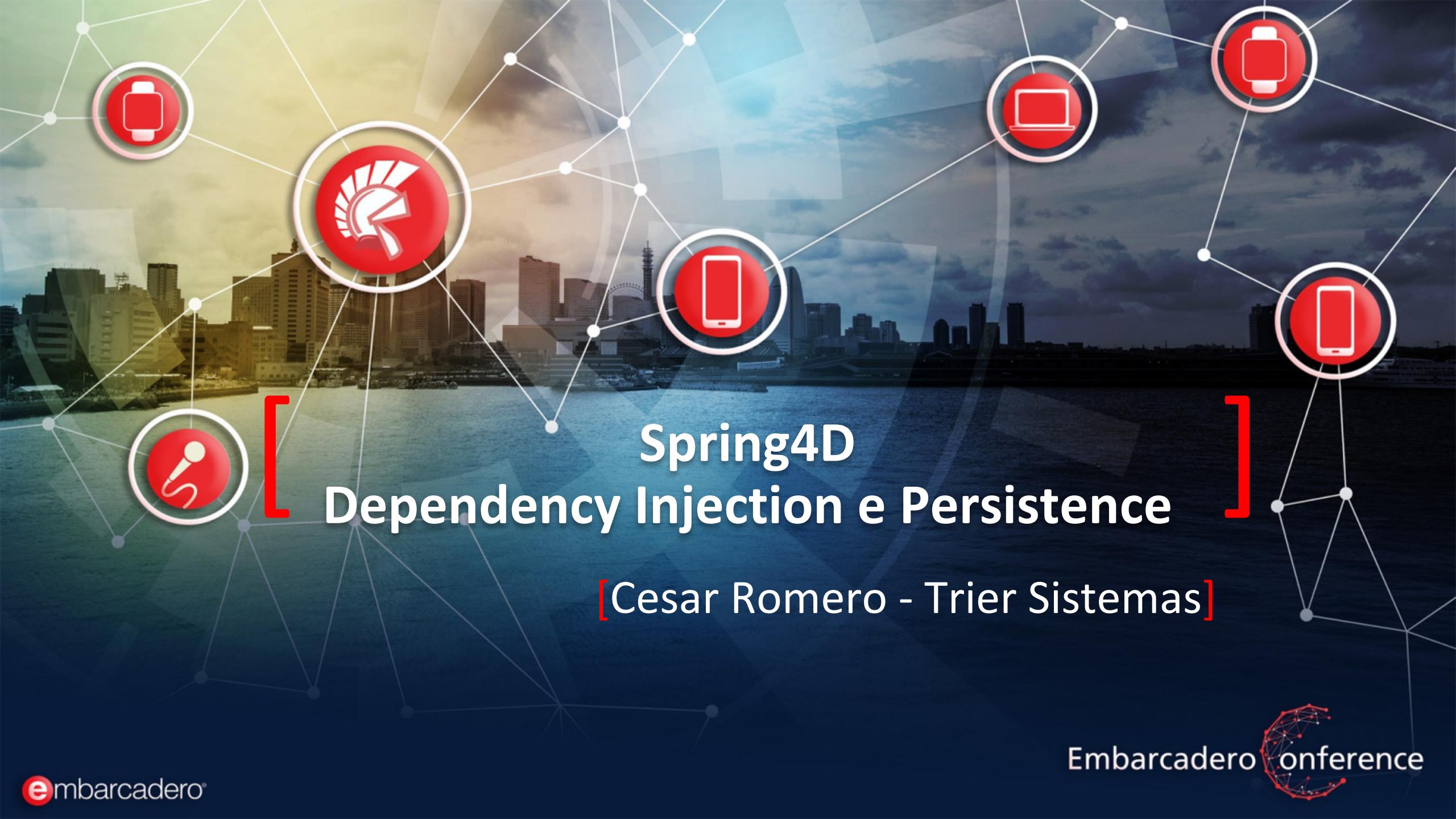




Embarcadero conference

embarcadero®



Spring4D Dependency Injection e Persistence

[Cesar Romero - Trier Sistemas]

Biografia

- Graduado em Ciências Contábeis pela UFPR
- Especialização em Software Orientado a Objetos PUC-PR
- Especialista em Arquitetura de Software
- Desenvolvedor Delphi desde 1995
- Natural do Rio De Janeiro
- Curitiba - Blumenau - Tubarão
- Arquiteto de Software e Coordenador de Desenvolvimento Delphi na Trier Sistemas

Trier Sistemas

- Fundada em 1992
- Localizada em Tubarão SC/Braço do Norte SC
- Líder no Segmento de Gestão para Farmácias e Drogarias
 - Mais de 8000 farmácias
- Grupo Trier
 - Trier Sistemas
 - Logtec Sistemas
 - ConvCard
 - Eireli
- Mais 350 funcionários diretos e indiretos

Referência - Projeto Central DFe

Serviço para monitorar NFe Entrada emitidas para um CNPJ

- Sefaz
 - Consulta NFe
 - Manifestação Destinatário
 - Download XML
- Cofre NFe Segura - REST
- Cadastro Inteligente - REST
- Servidor de Licenças
- Atualização automática
- Servidor POP3 - Recebimento NFe
- Integração com ERP
 - Database Notification
 - REST
 - Sistema Trier
 - Sistema Logtec

Central DFe - Desenvolvimento

Processos

- Levantamento de Requisitos
- Modelagem de Negócio - BPMN
- Modelagem do Banco de Dados
- **Design Patterns**
- Testes Unitários
- Métricas - Sonar Qube
- Continuous Integration

Frameworks e Componentes

- Spring4D 1.2
- OmniThreadLibrary
- JsonDataObjects
- ACBr
- *Foundation*
- *Fluent REST Client Library*
- *Ferramenta para gerar código*

Como atingir o objetivo?

- Design Patterns
 - S.O.L.I.D.
- Frameworks
 - Spring4D

Design Patterns

Os padrões de projeto, também conhecidos como design patterns, surgem para descrever, capturar e nomear técnicas para solução de problemas de software.

(Buschmann et al, 2007)

S.O.L.I.D.

S.O.L.I.D é um acrônimo para os primeiros cinco princípios de design orientado a objetos por Robert C. Martin, popularmente conhecido como Uncle Bob.

Quando combinados, ajudam o programador a criar software que são mais fáceis de manter e expandir.

Estes princípios também ajudam o programador a evitar “Code Smell”, refatorar com facilidade e também é parte do desenvolvimento Ágil.

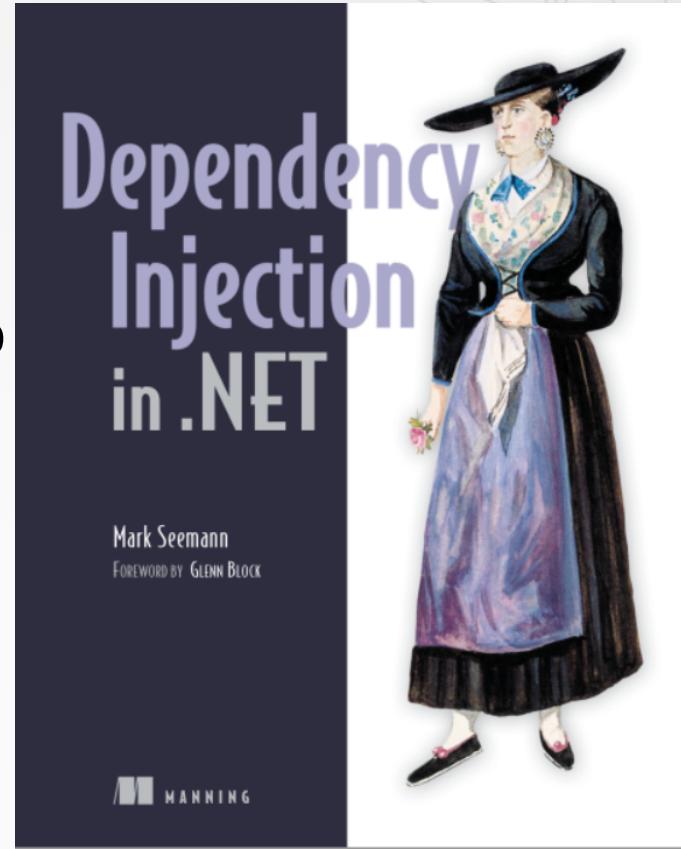
S.O.L.I.D.

S	SRP	Princípio da Responsabilidade Única	Uma classe deve ter apenas uma única responsabilidade.
O	OCP	Princípio Aberto-Fechado	Você deve ser capaz de estender o comportamento de uma classe, sem modificá-lo.
L	LSP	Princípio da Substituição de Liskov	As classes derivadas devem ser substituíveis por suas classes base.
I	ISP	Princípio da Segregação da Interface	Muitas interfaces específicas são melhores do que uma interface única.
D	DIP	Princípio da inversão da dependência	Dependa de uma abstração e não de uma implementação.

Dependency Injection

- Conjunto de padrões de projetos e Princípios.
- É uma forma de pensar no design do código do que uma tecnologia.
- Criar software de fácil manutenção utilizando Orientação a Objetos.

**DI não é o objetivo
DI é um meio para o Objetivo**



Dependency Injection in .NET
Autor Mark Seemann
Editora Manning

Dependency Injection - Benefícios

- Baixo Acoplamento
- Reduz Dependências
- Código Reutilizável
- Código Testável
- Código Legível

Benefícios do baixo acoplamento

Benefício	Descrição	Quando tem valor?
Instanciado quando necessário	Serviços podem ser trocados por outros.	Em aplicações padrão, nem tanto em aplicações empresariais onde o ambiente de execução é bem definido
Extensibilidade	Pode ser extendido e reutilizado de maneiras que não foram planejadas.	Sempre
Desenvolvimento paralelo	O código pode ser desenvolvido em paralelo por outras equipes.	Em aplicações grandes e complexas. Pouco necessário em aplicações simples e pequenas
Mantenabilidade	Classes com definição clara de responsabilidade são fáceis de manter.	Sempre
Testabilidade	Pode ser testado com Testes Unitários.	Somente se utiliza Testes Unitários. (Você realmente deve sempre fazer)

Spring4D

Originalmente escrito por Baoquan Zuo, e atualmente por Stefan Glienke.

- **Dependency Injection Container**
- Persistence
- ObjectDataSet - Binding
- Collections baseadas em Interface
- Logging
- Reflection que estende a RTTI do Delphi
- Testes Unitários

<http://www.spring4d.org>

Embarcadero Conference

Spring4D - Dependency Injection

- Dependency Injection sem Container
 - Poor man's Dependency Injection

The screenshot shows a Delphi IDE interface. On the left is the Project Explorer (Object Inspector) with a tree view of projects and files. Projects include 'IntroToDependencyInjectionProjectGroup' with sub-exes like 'StartingOut.exe', 'MovingToInterfaces.exe', 'ConstructorInjection.exe', 'UseMocks.exe', and 'IsolateInterfaces.exe'. 'IsolateInterfaces.exe' contains files 'uDoOrderProcessing.pas', 'uOrder.pas', 'uOrderEntry.pas', 'uOrderInterfaces.pas', 'uOrderProcessor.pas', and 'uOrderValidator.pas'. Below these are 'UseContainer.exe', 'CustomConstructor.exe', 'FieldInjection.exe', 'DecoratorInjection.exe', and 'InterceptorInjection.exe'. The main window shows a portion of the 'uDoOrderProcessing.pas' code:

```
implementation
uses
  uOrder,
  uOrderInterfaces,
  uOrderProcessor,
  uOrderValidator,
  uOrderEntry;

procedure DoOrderProcessing;
var
  Order: TOrder;
  OrderProcessor: IOrderProcessor;
begin
  Order := TOrder.Create;
  OrderProcessor := TOrderProcessor.Create(
    TOrderValidator.Create, TOrderEntry.Create);
  try
    if OrderProcessor.ProcessOrder(Order) then
      Writeln('Order successfully processed....');
  finally
    Order.Free;
  end;
end;
```

Spring4D - Dependency Injection - Demonstração

```
.  
. procedure Main;  
70 begin  
  Configuration := TApplicationSetup.BuildConfiguration;  
  Container     := TApplicationSetup.BuildContainer(Configuration);  
  try  
    SystemApplication := Container.Resolve<ISystemApplication>;  
    SystemApplication.RunApplication;  
  finally  
    SystemApplication := nil;  
    Container.Free;  
  end;  
80 end;  
>81 begin  
  Foundation.EventLog := TEventLog.Create(SERVICE_NAME, nil);  
  try  
    Main();  
  except  
    on E: Exception do  
    begin  
      EventLog.WriteException(E);  
    end;  
  end;  
90 end;  
. end.
```

Spring4D - DI

- Configuration
- Composition root
 - Register Types
 - Build
- Constructor Injection
- Property Injection
- Field Injection
- Method Injection
- Container

Spring4D Persistence - História

- Originalmente criado por Linas Naginionis, iniciando em 2012.
- Inspirado em micro ORM's .NET, na maioria PetaPoco e Java Hibernate.
- Utiliza as recursos atuais da linguagem Delphi como generics, atributos, RTTI avançada, records operator overloading.
- Integração com Spring4D.
- Já utilizado em produção com sucesso.

Fonte ItDevCon2016 - Stefan Glienke.

Spring4D Persistence - Recursos

- Pode ser utilizado com qualquer objeto
- Utiliza Atributos para definir o Mapeamento das Classes
- Tipos Nullable
- Suporta relacionamentos
 - Um para um
 - Um para muitos
 - Muitos para muitos (combinando 2 um para muitos)
- Lazy Load
- Optimistic Locking - [Version] column attribute
- Criteria API
- Binding - ObjectDataSet Virtual

Spring4D Persistence - Banco de Dados

- Interbase/Firebird
- MongoDB
- MS SQL Server
- MySQL/Maria DB
- Oracle
- PostgreSQL
- SQLite

Extensível para suportar outros bancos de dados

Spring4D Persistence

```
.  
. .  
750 [Entity]  
[Table(WEB_PROXY_TABLE, CENTRALDF_SCHEMA)]  
[Sequence(WEB_PROXY_PROXY_ID_SEQ, 1, 1)]  
TProxy = class(TCommonProxy)  
752 public  
  .  
  .  
  .  
  .  
  .  
  .  
  [AutoGenerated]  
  [Column(PROXY_ID_COL, [cpPrimaryKey, cpRequired])]  
  property Id;  
  .  
  .  
  .  
  .  
  .  
  [Column(ENDERECO_COL, [cpRequired], 100)]  
  property Endereco;  
  .  
  .  
  .  
  .  
  .  
  [Column(PORTA_COL, [cpRequired])]  
  property Porta;  
  .  
  .  
  .  
  .  
  .  
  [Column(USUARIO_COL, [], 50)]  
  property Usuario;  
  .  
  .  
  .  
  .  
  .  
  [Column(SENHA_COL, [], 50)]  
  property Senha;  
  .  
  .  
  .  
  .  
  .  
  [Column(ACTIVE_COL, [cpRequired])]  
  property Active;  
  .  
  .  
  .  
  .  
  .  
  [Column(CREATED_COL, [cpRequired])]  
  property Created;  
  .  
  .  
  .  
  .  
  .  
  [Column(UPDATED_COL, [])]  
  property Updated;
```

Spring4D Persistence - Produtividade

- Geração de código automatizada utilizando Templates
 - Classes
 - Mapeamento
 - Dicionário de Dados
- Binding com TObjectDataSet
- CRUD Dinâmico
 - MVVM

Transformer Code Generator

- Database First
- Baseado em templates mustache
- Usa metadados do banco de dados no formato JSON
- Simples e Rápido
- Em produção
 - Delphi
 - C#
 - Java
 - Configuração para Replicador de Banco de Dados
 - Documentação do Banco de Dados

Transformer Code Generator

- Modelagem do Banco de Dados - Enterprise Architect
- Geração/Sincronização do Banco de Dados Físico
- Importar os Metadados no formato JSON
- Selecionar Template Mustache
- Gerar Código

Transformer Code Generator

Database Builder

Start Page Database Builder

Central NFe (PostgreSQL)

- Tables
 - centraldf.cad_config
 - fk_cad_config_cad_empresa
 - centraldf.cad_empresa
 - centraldf.df_service_doc_fiscal
 - fk_df_service_doc_fiscal_df_service_evento_doc_fiscal
 - centraldf.df_service_evento_doc_fiscal
 - fk_df_service_evento_doc_fiscal_df_service_doc_fiscal
 - centraldf.sys_log
 - centraldf.sys_usuario
 - centraldf.web_certificado
 - centraldf.web_proxy
 - centraldf.web_web_service
 - fk_web_web_service_web_proxy
 - public.centraldf_doc_fiscal
 - association to: public.centraldf_evento_doc_fiscal
 - public.centraldf_evento_doc_fiscal
 - fk_centraldf_evento_doc_fiscal_centraldf_doc_fiscal
 - public.centraldf_integracao

Columns Constraints SQL Scratch Pad Database Compare Execute DDL

Name	Type	Len...	Scale	PK	Not Null
service_doc_fiscal_id	bigserial			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
service_evento_doc_fisca...	bigint			<input type="checkbox"/>	<input type="checkbox"/>
centraldf_doc_fiscal_id	bigint			<input type="checkbox"/>	<input type="checkbox"/>
doc_fiscal_id_erp	bigint			<input type="checkbox"/>	<input type="checkbox"/>
cnpj_empresa	varchar	18		<input type="checkbox"/>	<input checked="" type="checkbox"/>
chave	varchar	44		<input type="checkbox"/>	<input type="checkbox"/>
razao_social	varchar	120		<input type="checkbox"/>	<input checked="" type="checkbox"/>
nome	varchar	120		<input type="checkbox"/>	<input type="checkbox"/>
cnpj_cpf	varchar	18		<input type="checkbox"/>	<input checked="" type="checkbox"/>
inscricao_estadual	varchar	30		<input type="checkbox"/>	<input type="checkbox"/>
inscricao_municipal	varchar	30		<input type="checkbox"/>	<input type="checkbox"/>
data_emissao	date			<input type="checkbox"/>	<input checked="" type="checkbox"/>
tipo_doc_fiscal	smallint			<input type="checkbox"/>	<input checked="" type="checkbox"/>
tipo_operacao	smallint			<input type="checkbox"/>	<input checked="" type="checkbox"/>
numero	varchar	9		<input type="checkbox"/>	<input checked="" type="checkbox"/>
serie	varchar	3		<input type="checkbox"/>	<input checked="" type="checkbox"/>
valor	money			<input type="checkbox"/>	<input checked="" type="checkbox"/>

@ cesarliws@gmail.com

 @cesarliws

 /cesarliws

[OBRIGADO]

Embarcadero conference



Embarcadero conference

embarcadero®