CES Cheff

Manual de implantação

Revisão de 14/08/2026

CES Sistemas

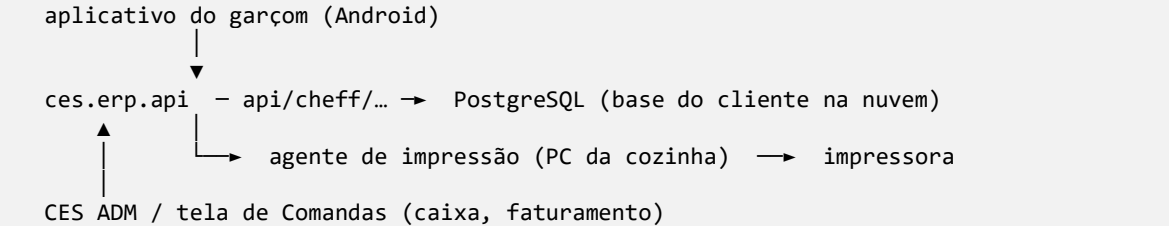
Francisco Beltrão · PR · www.csistemas.com.br · 0800 505 0035

Como colocar em produção o conjunto CES Cheff: API, aplicativo do garçom, agente de impressão da cozinha e a tela de comandas do CES ADM.

A ordem importa — cada passo depende do anterior.

> ***Chamados**: 0062772 (agente de impressão) · 0062787 (tela de comandas do ADM)*

0. As peças e como se ligam



O CES Cheff não tem mais API própria. Os endpoints passaram a viver dentro da `ces.erp.api`, que já está em produção, sob o prefixo **`api/cheff/`**. Foi a mesma stack dos dois lados — net8.0, CES.Auth pelo `ces.web` e `ces.orm.core` — então nada precisou ser reescrito.

> ***Durante a transição a `ces.cheff.api` continua no ar.** Isso é proposital: dá para migrar um cliente por vez, trocando só o endereço, e voltar atrás se algo não fechar. Ela só é desligada quando todos estiverem no ERP.*

Todo mundo fala com a API; **ninguém fala com o banco da nuvem direto**.

A API descobre em qual base gravar pelo **token** do usuário (DS_ChaveConexao), e descobre a **filial** pelo CNPJ que vem no mesmo token, consultando `Filiais`.

> *Consequência prática: usuário provisionado errado no CES.Auth grava na base ou na filial errada — e isso não aparece como erro, aparece como "sumiu o pedido".*

1. CES.Auth — pré-requisito de tudo

Item	Valor
idServico do CES Cheff	**8**
Auth de produção	`https://api.auth.databi.com.br`
Auth de homologação	`https://hom-auth.ceserp.com.br`

1. O serviço precisa existir no painel do CES.Auth com **idServico 8**. Esse

número é usado pelo aplicativo, pelo agente de impressão e pela tela de comandas do ADM. Número divergente autentica contra outro serviço e a API recusa o token — o sintoma é um **401 sem explicação**.

2. Cada garçom precisa de usuário com acesso ao serviço 8 e vínculo com a

empresa do cliente.

3. A empresa precisa estar com o **CNPJ correto**, e esse CNPJ tem de existir em

`Filiais` no ERP. É por ele que a API resolve o código de filial — o `ID_Empresa` do CES.Auth **não** é o código de filial do ERP.

2. Banco da nuvem

A base do cliente é a mesma que o restante do SaaS usa; a API não cria banco. Precisam existir `tblCartao`, `tblComanda`, `tblComandaItens`, `Produtos` e `Filiais`.

Conferir que os cartões têm `ID_Empresa` preenchido. Cartão com `ID_Empresa` nulo ou zero aparece para **todas** as filiais — a API trata `= 0` como compatibilidade com versões antigas, de propósito, para o registro não sumir da tela do operador.

3. A API — agora é a do ERP

Os endpoints do Cheff vivem em `ces.erp.api`, em `Features/Cheff`, branch `master`, commit `9a4756d` ou posterior. **Não há mais servidor separado para publicar**: se o ERP do cliente já está no ar, a API do Cheff já está junto.

3.1 As rotas

Todas sob o prefixo da área:

```
api/cheff/Comanda      api/cheff/Cartao      api/cheff/Produto
api/cheff/Categoria    api/cheff/CategoriaSub    api/cheff/Impressao
```

O prefixo existe porque `Produto`, `Categoria` e `CategoriaSub` **já existiam** na API do ERP. `GET api/Produto` sairia dos dois lados, e rota duplicada não quebra a compilação — quebra na primeira chamada, em runtime.

3.2 Publicar

É a publicação normal do ERP:

```
dotnet publish -c Release -o C:\inetpub\ces.erp.api
```

Servidor com IIS e o **ASP.NET Core Hosting Bundle 8** (ele instala o `AspNetCoreModuleV2`; sem ele o site sobe e devolve **500.19** ou **502.5**), pool em *Sem código gerenciado*, `AlwaysRunning`, `Idle Time-out = 0` e preload habilitado — senão o pool dorme e a primeira chamada do dia leva minutos.

Nada muda no `appsettings` por causa do Cheff: ele usa a mesma autenticação, o mesmo tenant e os mesmos bancos que o ERP já inicializa.

3.3 Conferir que subiu

Uma rota inexistente tem de devolver 404 e uma rota real tem de devolver 401 sem token:

```
GET {erp}/api/cheff/Comanda/RotaQueNaoExiste   ->  404 esperado
GET {erp}/api/cheff/Comanda/Todas              ->  401 esperado
```

Se as duas derem 404, o site não carregou a aplicação — ou o publish saiu sem o slice do Cheff.

3.4 Endereço

Anotar a base final, **com o `/api/cheff`**:

```
https://<servidor-do-erp>/api/cheff
```

Ela é usada em **três** lugares: parâmetro do ADM, configuração do agente de impressão e build do aplicativo. Os três recebem exatamente esse mesmo valor.

> ***Enquanto a `ces.cheff.api` estiver no ar**, o endereço antigo (`http://servidor:49982`) continua funcionando. É o que permite migrar um cliente por vez: troca o endereço de um, valida, passa para o próximo.*

4. CES ADM — tela de comandas

1. **DLL**: `ces.adm.comercial.comanda.dll` **5.0.0.17** ou superior, em `C:\CES Sistemas\ces_adm.Net`. Conferir pelo rodapé da tela de Comandas.
2. **Parâmetros**, pela ferramenta de Manutenção de parâmetros — nunca por SQL direto, porque a sincronização por MD5 apaga parâmetro órfão:

Parâmetro	Tipo	Valor
TP_USA_CES_CHEFF	Boolean (default False)	**liga** a integração
DS_URL_API_CHEFF	texto	`https://<servidor-do-erp>/api/cheff`

Com `TP_USA_CES_CHEFF` desligado a tela se comporta exatamente como antes — é o que garante que cliente sem o aplicativo não muda de comportamento.

> ***Migrar para o ERP é só trocar este parâmetro** — não precisa de DLL nova. A partir da 5.0.0.17 o `DS_URL_API_CHEFF` carrega a base **completa, com caminho**, e o módulo emenda `/Comanda`, `/Cartao` direto nela. Instalação antiga, com só o host no parâmetro (`http://servidor:49982`), continua funcionando sem ninguém tocar em nada: quando o valor não traz caminho, o `/api` é completado sozinho. A mesma DLL atende as duas APIs.*

3. O usuário do caixa precisa das **credenciais de API** no cadastro de usuário (`DS_Usuario_api` / `DS_Senha_api`) e de acesso ao serviço 8. Sem isso a tela mostra *"Sem acesso à API do CES Cheff"*.
4. A tela cria sozinha `C:\CS\LOG\COMANDA`. Terminal sem permissão em `C:\` cai no perfil do usuário.

5. Agente de impressão — PC da cozinha

1. Levar o `dist\CesCheffImpressao-setup-<versão>.zip`, **extrair o ZIP inteiro** e rodar `INSTALAR.bat`. Não exige Python na máquina. Para saber a versão instalada depois, o ícone da bandeja → **Status** mostra na primeira linha.
2. Na primeira execução a configuração abre sozinha:

Aba	O que informar
1. Conexão	**e-mail e senha** do CES, e a **empresa**
2. Impressora	impressora de saída, largura da bobina, vias
3. O que imprimir	tudo (padrão) ou só as categorias/subcategorias da praça
4. Geral	iniciar com o Windows, começar pausado, nível de log

3. Os **tempos** da aba 4 (verificar a cada 5 s; 120 min até economizar; 60 min em economia) são **fixos** e aparecem só como informação. Para alterar, editar `%APPDATA%\ces.cheff.impressao\config.json` — é decisão de suporte, não do usuário da cozinha.

4. **Um agente por impressora.** Duas praças (cozinha e churrasqueira) são dois terminais, cada um com o filtro da sua praça na aba 3.

> ***A partir da 1.0.2 não há endereço para preencher.*** *Os dois endereços são fixos no programa, porque são iguais em todo cliente: | | | |---|---| | API | `http://apierp.databi.com.br/api/cheff` | | CES.Auth | `https://api.auth.databi.com.br/api` | Quem já tinha o agente instalado recebe os novos endereços ao atualizar: eles sobrescrevem o que estava gravado. Trocar de ambiente é alterar `URL_API` e `URL_AUTH` em `ces_cheff_impressao/config.py` e reempacotar.*

6. Aplicativo do garçom — APK

Os endereços entram em **tempo de build**; o mesmo código gera o APK de qualquer cliente:

```
flutter build apk --release --dart-define=CES_API_APP=https://servidor-do-erp/api/cheff
--dart-define=CES_API_AUTH=https://api.auth.databi.com.br/api
```

Repare no `/api/cheff` no fim do `CES_API_APP`: o app monta as chamadas como `{base}/Comanda`, `{base}/Cartao`, então a base precisa incluir o prefixo da área. É por isso que a migração para o ERP não exigiu mudar uma linha do aplicativo.

> *APK gerado **sem** esses dois parâmetros aponta para a rede interna da CES e não funciona no cliente.*

Assinatura. A chave vai em `android/app/key.properties` — dentro de `app/`, não na pasta `android/`. Sem esse arquivo o build **não falha**: o APK sai assinado com a chave de *debug* e só aparece um aviso no meio da saída do Gradle.

```
apksigner verify --print-certs app-release.apk
```

Se aparecer `CN=Android Debug`, refazer com a chave certa. Não use `keytool -printcert -jarfile`: ele responde *"não é um arquivo jar assinado"* mesmo num APK bem assinado, porque só enxerga assinatura v1.

Subir o `versionCode` do `pubspec.yaml` a cada APK distribuído, senão o Android recusa instalar por cima. Passo a passo completo no README do `ces.cheff.novo`.

6A. Publicar na Google Play — assinatura passo a passo

O APK acima serve para instalar direto no aparelho. Para a **loja** muda duas coisas: o formato é `.aab` e a assinatura precisa ser de verdade.

6A.1 Criar a chave (uma vez para o app, para sempre)

```
keytool -genkeypair -v -keystore ces-cheff-upload.jks -storetype JKS -keyalg RSA -
keysize 2048 -validity 10000 -alias ces-cheff
```

Ele pergunta uma senha e os dados do certificado (nome, organização, cidade, estado, país — use os da CES Sistemas).

> ***Guarde o `.jks` e as senhas fora da máquina de build*** *— cofre de senhas da empresa, não a pasta do projeto. É o mesmo arquivo para* ***todas*** *as versões futuras deste app.*

6A.2 Apontar o projeto para a chave

Copie o `.jks` para `android/app/` e crie `android/app/key.properties`:

```
storePassword=<senha do keystore>
keyPassword=<senha da chave>
keyAlias=ces-cheff
storeFile=ces-cheff-upload.jks
```

Os dois arquivos estão no `.gitignore` e **não vão para o repositório**, de propósito.

6A.3 Gerar o pacote da loja

```
flutter build appbundle --release --dart-
define=CES_API_APP=http://apierp.databi.com.br/api/cheff --dart-
define=CES_API_AUTH=https://api.auth.databi.com.br/api
```

Sai em `build\app\outputs\bundle\release\app-release.aab`.

> *A Play* ***não aceita mais APK*** *para app novo — é `.aab`. É por isso que o comando é `appbundle` e não `apk`.*

6A.4 Antes de enviar

Conferir	Como
Não está com chave de debug	`apksigner verify --print-certs` no APK equivalente
`versionCode` subiu	`pubspec.yaml`, número depois do `+`
Pacote correto	`br.com.csistemas.ces_cheff` — não muda nunca mais depois do 1º envio

6A.5 Na Play Console

1. **Criar app** → nome, idioma, tipo (aplicativo), gratuito.
2. Em **Versões → Produção** (ou *Teste interno*, recomendado para a primeira),

enviar o `.aab`.

3. Aceitar a **assinatura de apps do Google Play**. A partir daí o `.jks` que você criou passa a ser a **chave de upload**, e o Google guarda a chave de assinatura final.

4. Preencher o que a loja exige antes de publicar: ícone 512×512, capturas de tela, descrição, política de privacidade, classificação indicativa e o questionário de segurança de dados.

> ***Sobre perder a chave:*** *com a assinatura do Google ligada, perder o `.jks` de upload é recuperável — o suporte da Play redefine. Se você optar por *não* usar a assinatura do Google, perder a chave significa **nunca mais atualizar o app**; só restaria publicar outro, com outro pacote, e pedir para todos reinstalarem.*

6A.6 Nas próximas versões

Subir o `versionCode`, gerar o `.aab` com a **mesma** chave e enviar. Nada dos passos 6A.1 e 6A.2 se repete.

7. Validação ponta a ponta

1. Garçom entra no aplicativo e vê as categorias e os produtos da empresa dele.
2. Abre um cartão, digita o nome e lança um item.
3. O papel sai na impressora da cozinha em até 5 segundos.
4. Na tela de Comandas do ADM, clicar em **Atualizar**: a comanda aparece com o nome digitado e o valor correto.
5. Lançar mais um item no aplicativo e clicar em Atualizar de novo: o rodapé tem de informar *"1 atualizada(s) com 1 item(ns) novo(s)"* e o total tem de subir.
6. Fechar os cartões no ADM (Fechamento de cartões, com senha de gerente).
7. Clicar em Atualizar: a comanda tem de sumir do aplicativo **e** da base da nuvem.

Falhou algum passo, o motivo está em:

- `C:\CS\LOG\COMANDA\comanda_erro_AAAA-MM-DD.log` — lado ADM
- `%APPDATA%\ces.cheff.impressao\logs\` — lado agente de impressão

8. Migrar um cliente que já usa a API antiga

Como a `ces.cheff.api` continua no ar, a virada é por cliente e reversível. Faça **um de cada vez** e valide antes de passar ao próximo — assim, se algo falhar, você sabe qual peça foi.

Ordem	O que fazer	Como voltar atrás
1	Publicar a `ces.erp.api` e conferir o 401 em `api/cheff/Comanda/Todas`	nada mudou ainda
2	Atualizar a DLL do ADM para 5.0.0.17	volta a DLL anterior

Ordem	O que fazer	Como voltar atrás
3	Trocar `DS_URL_API_CHEFF` para `.../api/cheff` e testar o Atualizar	volta o valor antigo
4	Apontar o agente de impressão para a nova base (aba 1)	volta o valor antigo
5	Gerar e instalar o APK novo com o `--dart-define`	reinstala o APK anterior

Só depois que **todos** os clientes estiverem no ERP é que a `ces.cheff.api` é desligada.

> *A ordem não é arbitrária: o ADM é o primeiro porque é onde o erro aparece escrito em arquivo (`C:\CS\LOG\COMANDA`). O APK é o último porque é o mais caro de reverter — precisa reinstalar em cada aparelho.*

9. Pendências conhecidas

9.1 Credencial no repositório. O `appsettings.Development.json` está versionado com senha de banco em texto puro, desde o commit `4011d72` (anterior a este trabalho). Precisa sair do repositório e entrar no `.gitignore`; como já está no histórico, a senha também deve ser trocada. Não afeta produção — o arquivo Development não vai para o servidor — mas é exposição real de credencial.

9.2 Carga manual. Pedido lançado no aplicativo depois que a tela de comandas já estava aberta só aparece ao clicar em Atualizar; o ciclo automático de 5 s lê apenas o SQL local. Há proposta de tornar a carga da nuvem automática a cada 30 s, usando o contador de itens que a API já devolve para só descer ao detalhe da comanda que mudou. **Aguardando definição.**

9.3 Imagens de produto. O aplicativo tem o espaço da imagem, mas não existe imagem cadastrada nem definição de onde armazenar (proposta: MinIO). **Aguardando definição.**