CES Inventário

Manual de implantação

Revisão de 13/08/2026

CES Sistemas

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

Coletor de contagem de estoque para Android (`ces_inventario2`). O conferente bipa os produtos do lote no aparelho, o app guarda tudo em banco local e envia a contagem para a retaguarda CES ao encerrar.

> ***Repositório**: `ces.inventario.novo` (Azure DevOps) · pasta local `D:\Projetos\ces_inventario2` **Versão atual**: 1.0.5+4 · **Pacote**: `br.com.csistemas.ces_inventario`*

1. Como ele conversa com a retaguarda

```
aparelho / coletor Android
          │
          │  SOAP sobre HTTP
          ▼
http://<IP>:<porta>/Servicos.asmx      (web service CES)
          │
          ▼
SQL Server da retaguarda
```

Diferente do CES Cheff e do CES Despesas, **este app não usa o CES.Auth**. Não há token, não há `idServico`, não há login centralizado.

	Como é aqui
Autenticação do web service	código fixo `20401359`, enviado em toda chamada
Login do conferente	local, contra a lista de participantes baixada do servidor
Endereço do servidor	digitado no próprio aparelho, na primeira tela
Protocolo	**HTTP puro** (`usesCleartextTraffic="true"`); não há suporte a HTTPS

> *A senha do participante trafega e fica gravada **em texto puro** no SQLite do aparelho. Aparelho de conferente é item de chão de fábrica: se for extraviado, as senhas de contagem daquele inventário devem ser trocadas.*

2. Pré-requisitos na retaguarda

Nada disso é feito pelo app — se faltar, ele simplesmente não traz dados.

Precisa existir	Onde	Sintoma se faltar
Inventário **aberto** (`DT_Fechamento` nulo) para a filial	`tblInventario_conferencia`	"Nenhum inventário aberto"
Participante com nome e senha	`tblInventario_conferencia_participantes`	não aparece no dropdown de login
Lotes com `ID_Participante_Contagem` do participante e `TP_Excluido = 0`	`tblInventario_conferencia_lotes`	"Nenhum lote atribuído a você neste inventário"

Precisa existir	Onde	Sintoma se faltar
Produtos com `Status = 'S'`	`produtos` e `[Produtos Grade]`	produto não é reconhecido na bipagem

Web service publicado em HTTP, no caminho `/Servicos.asmx`, respondendo aos métodos `StatusWS`, `RequestDataSql` e `InsertDataSql` no namespace `http://comanda.com/Servicos/`, e aceitando o código de autenticação `20401359`.

Rede: o aparelho precisa enxergar o servidor pelo IP e porta informados. Como é HTTP puro, tem de ser rede interna ou VPN — não exponha esse serviço à internet.

3. Configuração no aparelho (primeira execução)

Na primeira abertura o app vai direto para a tela de configuração:

Campo	Exemplo	Observação
URL / IP	`192.168.0.10`	pode digitar com ou sem `http://`; o app limpa
Porta	`6655`	
Código da Filial	número inteiro maior que zero	é a chave de tenant de todo o sincronismo
Leitor físico	ligado/desligado	ligado, o campo de leitura **não abre o teclado virtual**

Ao gravar, o app testa a conexão (`StatusWS`) e só avança se o web service responder. Mensagem *"Não foi possível conectar ao WebService"* significa endereço, porta ou rede — não é problema de cadastro.

Nenhum desses campos tem valor padrão: **todo aparelho precisa ser configurado individualmente**. Há um botão **Reset** na tela que limpa tudo e volta ao estado de fábrica.

4. Uso no dia a dia

```
Configuração → Login (baixa os dados) → Seleção de lote → Leitura → Encerrar
```

Login. O conferente escolhe o nome numa lista e digita a senha. Ao entrar, o app baixa inventários, participantes, lotes e o **catálogo inteiro de produtos**, em páginas de 200, com a barra "Baixando Produtos…". É o passo demorado — em base grande, faça o primeiro login com o aparelho na rede boa e sem pressa.

Seleção de lote. Lista só os lotes atribuídos àquele conferente. Lote com contagem ainda não enviada aparece primeiro, com o selo *"N não enviado(s)"*.

Leitura. Bipa pelo leitor físico ou pela câmera (com lanterna). A quantidade vai de 1 a 9999 e volta para 1 depois de cada leitura. Leituras do mesmo produto somam na mesma linha.

> *Se o inventário estiver com filtro de categoria ligado, produto de outra categoria é recusado com *"Produto de outra categoria, não é possível fazer a leitura"*. É proposital: evita contar na praça errada.*

Encerrar. Envia a contagem em pacotes de 50 itens e depois marca o lote como lido, gravando início, fim e o modelo do aparelho que contou. **A leitura local só é apagada depois que o envio dá certo** — se cair a rede no meio, nada se perde: o lote volta para a lista com o selo de pendente.

5. Gerar o APK

5.1 Antes de tudo: a chave de assinatura

Dois arquivos precisam existir em `android/app/` — dentro de `app/`, não na pasta `android/`:

- `key.properties` (`storePassword`, `keyPassword`, `keyAlias=ces-alias`, `storeFile=ces-key.jks`)
- `ces-key.jks`

Os dois estão no `.gitignore` e **não vêm no clone**. Sem eles o Gradle quebra logo na configuração: **nenhum build funciona, nem o de debug**. É o primeiro erro de quem monta uma máquina nova.

5.2 Ambiente

- Flutter 3.41.3 (stable) · Dart 3.11.1
- NDK **27.0.12077973** (fixado no `build.gradle.kts`)
- JDK 11 · Gradle 8.12 · AGP 8.7.3 · Kotlin 2.1.0
- `flutter.sdk` preenchido em `android/local.properties`

5.3 Passos

```
flutter pub get
flutter test
flutter build apk --release
```

São 44 testes. O APK sai em `build\app\outputs\flutter-apk\app-release.apk`.

Não há `--dart-define`: **o endereço do servidor não entra no build**, é digitado no aparelho. O mesmo APK serve para todos os clientes.

5.4 Antes de distribuir

Subir o versionCode em `pubspec.yaml` (`version: 1.0.5+4` → o número depois do +), senão o Android recusa instalar por cima. E conferir a assinatura:

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

Se aparecer `CN=Android Debug`, o `key.properties` não foi lido — refaça.

5.5 Ícone

Só quando `lib/assets/logo.png` mudar:

```
dart run flutter_launcher_icons
```

O `pubspec.yaml` **não** configura adaptive icon, de propósito: sem ele o Android aplica o tratamento legado e o app fica com a mesma aparência dos outros da família CES na tela inicial. Não acrescente `adaptive_icon_background`.

6. Validação depois de instalar

1. Configurar IP, porta e filial; gravar. A conexão tem de ser aceita.
2. Fazer login com um participante real. O download de produtos precisa terminar.
3. A tela de lotes tem de listar os lotes daquele conferente.
4. Bipar um produto conhecido: a linha aparece com a quantidade.
5. Bipar um produto de outra categoria (se o inventário tiver filtro): tem de ser recusado.
6. Encerrar o lote. Conferir na retaguarda que `tblInventario_Conferencia_Lotes_Detalhe` recebeu os itens e que o lote ficou com `TP_Lido = 1`.

7. Pendências conhecidas

Item	Situação
Senha em texto puro — a senha do participante é gravada sem cifra no SQLite do aparelho e comparada em texto puro no login.	Risco aceito hoje; trocar senhas do inventário em caso de extravio do aparelho.
SQL montado no aplicativo — o app concatena os valores e manda o comando pronto para o servidor executar, sem escape de aspas.	Funciona porque os valores vêm do catálogo, mas é frágil. Vale trocar por parâmetros.
README do repositório ainda é o template padrão do Flutter.	Substituir pela documentação real, como foi feito no `ces.cheff.novo`.
Código morto — model e tabela `inventario_series` existem, mas nenhuma tela usa.	Remover ou concluir.
`minSdk`/`targetSdk` não fixados — vêm do Flutter SDK instalado.	Trocar de versão de Flutter muda o alvo sem ninguém decidir.