NewSoft NextGen

Stack Modernizada e API COM do motor VFP

Documento técnico · 2026-04-22

Este documento descreve a arquitectura técnica da plataforma NewSoft NextGen — a modernização web da aplicação legacy de gestão de clínicas escrita em Visual FoxPro 9. Mantém-se o VFP como motor de regras de negócio, exposto através de COM Server; a nova camada web oferece UI moderna, API GraphQL e um adaptador (bridge) que invoca o COM.

Conteúdo:

1. Visão geral da stack

A NewSoft NextGen combina um front-end web servido por nginx, uma API GraphQL em Node.js (Bun) que lê/escreve directamente em SQL Server via Prisma, e uma bridge dedicada que invoca a lógica de negócio existente no VFP através de COM. A base de dados é partilhada.

1.1 Componentes

Frontend web	HTML/CSS/JS estático servido por nginx (prototipos). Entry point: /prototipos/
nginx	Reverse proxy para /graphql !’ backend:4000. Serve /prototipos e /js estáticos. TLS com certificado de nextgen.newsoftds.pt.
Backend GraphQL	Bun + GraphQL Yoga + Pothos + Prisma. Porta 4000. Queries/mutations de CRUD e leituras complexas.
Admin framework	Camada genérica adicionada ao backend (src/admin + src/schema/admin.ts) que auto-serve CRUD de tabelas/configurações com metadata declarativa.
SQL Server	srv-clientscloud2.imaginasoft.pt:50036 — BD ns_196196197 (principal) e nsc_196196197 (clínica). Partilhada entre a web e o VFP.
VFP Bridge (Node)	Serviço HTTP na porta 5000 (em C:\newsoft\app\vfp-bridge). Pool de workers que mantêm instâncias COM vivas e fazem proxy JSON.
VFP COM Server	NewsoftAPI.dll (32-bit). ProgID newsoftapi.NewSoftAPI. Registada via regsvr32. Contém a lógica de negócio portada da aplicação legacy.
Worker de COM	Subprocesso PowerShell 32-bit por slot do pool — mantém o objecto COM aberto e aceita comandos via stdin/stdout (usado porque winax precisa de Python+VC++ que não estão neste VPS).

1.2 Fluxos principais

Fluxo A — Leitura/escrita de tabelas CRUD

```
Browser
  % % > HTTPS /prototipos/tables.html?t=bancos
        (serve-se do disco via nginx)
        % % > fetch /graphql (adminList / adminUpsert / adminDelete)
              % % > nginx proxy_pass !’ http://127.0.0.1:4000/graphql
                    % % > Bun + Pothos + Prisma
                          % % > SQL Server (ns_196196197.bancos)
```

Fluxo B — Regras de negócio (agenda, validações, orçamento)

```
Browser
  % % > fetch /graphql (ex. hora-livres, pode-gravar)
        % % > GraphQL resolver
              % % > HTTP fetch http://localhost:5000/horas-livres (VFP Bridge)
                    % % > Pool de PSCOMWorker (PowerShell 32-bit)
                          % % > Invoke-Member "HorasLivresPeriodos" no objecto COM
                                % % > NewsoftAPI.dll executa algoritmo (adaptado
                                    de agenda.scx::horaslivresperiodos, 491 LOC)
                                      % % > ODBC !’ SQL Server
                                            % % > Retorna JSON (slots + total)
```

1.3 Topologia de portas

Porta	Serviço	Estado	Observações
80	nginx HTTP	Serviço Windows (Automatic)	Redirect 301 para HTTPS.
443	nginx HTTPS	Serviço Windows (Automatic)	TLS 1.2/1.3, HSTS. Redirect / !' / prototipos/.
4000	Backend GraphQL	Serviço Windows newsoft-backend (Automatic)	Bun src/server.ts. Proxy via /graphql.
5000	VFP Bridge (opc.)	Processo PM2 / manual	Invoca COM VFP. Só usado para regras que vivem no VFP.

2. Servidor web e GraphQL

2.1 nginx — configuração relevante

O ficheiro C:\nginx\conf\nginx.conf define dois server blocks (80!'HTTPS e 443). No HTTPS:

```
# Estáticos
root C:/newsoft/app/public;
location /prototipos/ { try_files $uri $uri/ =404; }
location /js/         { try_files $uri =404; expires 1h; }

# GraphQL proxy com CORS
location /graphql {
  proxy_pass http://127.0.0.1:4000/graphql;
  proxy_set_header Host  $host;
  proxy_set_header X-Real-IP $remote_addr;
  add_header Access-Control-Allow-Origin "https://nextgen.newsoftds.pt" always;
  if ($request_method = OPTIONS) { return 204; }
}

# Redirects
location = / { return 301 /prototipos/; }        # root !' menu
location /   { return 301 /prototipos/login.html; }
```

Rate-limit: zone login 5r/m; zone api 30r/s com burst 50 nodelay.

2.2 Backend Bun + GraphQL Yoga

Stack: Bun 1.x + GraphQL Yoga + Pothos (schema builder) + Prisma (574 modelos introspectados). Entry: C:\newsoft\app\backend\src\server.ts.

- Schema modular em src/schema/{agenda,pacientes,medicos,clinica,admin}.ts
- Scalars custom: DateTime, Decimal, JSON
- Autenticação JWT (ver schema/auth.ts)
- CORS "*" para aceitar chamadas a partir de file:// ou outros hosts em dev

2.3 Connection string SQL Server

Prisma lê DATABASE_URL de backend/.env:

```
sqlserver://srv-clientscloud2.imaginasoft.pt:50036
  ;database=ns_196196197
  ;user=newsoft_196196197
  ;password=…
  ;encrypt=true;trustServerCertificate=true
```

A mesma BD está disponível ao VFP COM via ODBC com o driver "ODBC Driver 17 for SQL Server" (32-bit):

```
Driver={ODBC Driver 17 for SQL Server}
 ;Server=srv-clientscloud2.imaginasoft.pt,50036    <-- virgula, nao dois-pontos
 ;Database=ns_196196197
 ;UID=newsoft_196196197;PWD=…
 ;Encrypt=yes;TrustServerCertificate=yes
```

3. Framework admin — Tabelas e Configurações

As ~170 forms tbl*/config* da aplicação VFP (menus "Tables" e "Configuration") são modernizadas por uma camada admin genérica. Em vez de escrever uma página por tabela, cada tabela declara metadata (colunas, labels, validações) e a UI auto-renderiza list + edit modal.

3.1 Componentes

- backend/src/admin/metadata.ts — registo declarativo das tabelas (11 registadas inicialmente).
- backend/src/schema/admin.ts — resolvers genéricos sobre Prisma.
- public/prototipos/tables-menu.html — página de entrada (agrupa Tabelas/Configurações/RGPD).
- public/prototipos/tables.html?t=<key> — edição genérica da tabela.

3.2 API GraphQL admin

```
# Listagem
query { adminTables { key label menu vfpOrigin readonly } }

# Metadata de uma tabela
query { adminTableInfo(key: "bancos") { label fields { name type required } } }

# CRUD
query { adminList(key: "bancos", search: "", page: 1, pageSize: 50) { total rows } }
query { adminGet(key: "bancos", id: "1") }                        # JSON

mutation { adminCreate(key: "bancos", data: {nome:"BCP"}) }      # JSON
mutation { adminUpdate(key: "bancos", id: "1", data: {…}) }
mutation { adminDelete(key: "bancos", id: "1") }
```

3.3 Exemplo de metadata

```
// backend/src/admin/metadata.ts
caixas: {
  key: 'caixas',
  prismaModel: 'caixas',
  label: 'Seguradoras',
  menu: 'tabelas', menuOrder: 1,
  vfpOrigin: 'TblCaixas',
  fields: [
    { name: 'cod_caixa',  label: 'Código', type: 'int',    pk: true, readOnly: true },
    { name: 'nome_caixa', label: 'Nome',   type: 'string', required: true,
searchable: true, maxLength: 100 },
    { name: 'activo',     label: 'Ativo',  type: 'bool' },
  ],
  orderBy: 'nome_caixa',
}
```

3.4 Tabelas já registadas

Chave	Label	Menu	Origem VFP
caixas	Seguradoras	tabelas	TblCaixas
espec	Especialidades	tabelas	TblEspec
catEntidade	Categoria de Entidade	tabelas	TblCatEntidade
t_salas	Salas	tabelas	TblSalas
estcons	Estados de Consulta	tabelas	—

tipocons	Tipos de Consulta	tabelas	TblTipotrat
cattrat	Categorias de Tratamento	tabelas	Tblcattrat
bancos	Bancos	tabelas	—
moeda	Moedas	tabelas	TblMoedas
ConfigAPP	Configuração APP	configuracoes	configAPP
configRGPD	Configuração RGPD	rgpd	CONFIGRGPD

4. API COM NewSoft

ProgID	newsoftapi.NewSoftAPI
Tipo	In-proc DLL 32-bit (OLEPUBLIC, base Session)
Ficheiro	C:\VFP\newsoftapi\NewsoftAPI.dll
Source VFP	C:\VFP\newsoftapi*.prg (projecto newsoftapi.pjx gerado programaticamente)
Dependências	VFP 9 Runtime 32-bit, ODBC Driver 17 for SQL Server
Registo	regsvr32 (via SysWOW64 para 32-bit)

4.1 Lista completa de métodos

Método	Parâmetros	Retorno
TestConnection	—	JSON { success, data: { version, database, server_time_utc } }
Ping	—	boolean .T.
GetVersion	—	string "1.0.0"
ListMethods	—	JSON array de nomes de métodos expostos
Diagnostics	—	JSON { classes: {…}, lConnected, cLastError, nSQLHandle }
HorasLivresPeriodos	(tcData "YYYY-MM-DD", tnDentista int, tnDuracao min)	JSON { success, data: { total, slots: [{hora, hora_min, duracao, sala, tipocons, idposto}] } }
PodeGravar	(tnEpisodio, tcData, tnHora, tnDentista, tnPaciente, tnDuracao, tcSala)	JSON { can_save: bool, reasons: [{code, message}] }
GravaConsulta	(tcJSON com data/hora/dentista/paciente/sala/…)	JSON { episodio, criado }
Facturar	(tnPaciente, tcTratamentosJSON, tcMetodoPag)	JSON (delegado a stored proc up_factura_emitir)
ValidarNIF	(tcNIF)	JSON { valid: bool, nif }
CalcularOrcamento	(tnPaciente, tcCodTratamentosJSON)	JSON { total_base, total_desconto, total_iva, total, linhas: [...] }
CalcularHonorarios	(tnMedico, tcDataInicio, tcDataFim)	JSON { medico, total, linhas }
GerarRecibo	(tnFactura, tcDadosPagamentoJSON)	JSON (delegado a stored proc up_recibo_gerar)
SubmeterADSE	(tcClaimIDsJSON, tcLote)	JSON { lote, total, enviados, falhados }
SQLQuery	(tcSQL — só SELECTs)	JSON { data: [… rows …] } (para debug)

4.2 Envelope de resposta uniforme

```
// Sucesso
{ "success": true,  "data": { … } }

// Erro (nunca lança excepção COM — intercepta e retorna JSON)
{ "success": false, "error": "mensagem legivel", "code": "OPT_CODE" }
```

4.3 Chamada a partir do Node.js (via Bridge HTTP)

```
// POST http://localhost:5000/horas-livres
{
  "data":     "2026-04-22",
  "dentista": 1,
  "duracao":  30          // minutos
}

// Resposta
{
  "success": true,
  "data": {
    "data":         "2026-04-22",
    "dentista":     1,
    "duracao_min": 30,
    "total":        16,
    "slots": [
      { "hora": "09:00", "hora_min": 540, "duracao": 30, "sala": "S1", "tipocons": 0,
"idposto": 0 },
      …
    ]
  }
}
```

5. Regras de negócio extraídas do VFP

O código legacy vive dentro de ficheiros .scx (forms binários do VFP). A análise prévia extraiu o código de cada método para forms_extraction.json (1 MB). Os métodos críticos da agenda.scx foram depois portados para classes standalone (C:\VFP\newsoftapi\business\), com as dependências de cursor-form substituídas por cursores carregados via SQL.

5.1 HorasLivresPeriodos (491 LOC originais)

- Objectivo: calcular slots livres para (data, dentista, duração pretendida).
- Carrega tabelas de actividade (agdactiv) e inactividade (agdinact) do dentista.
- Chama DataDisponivel (180 LOC) para obter períodos de trabalho do dia-da-semana, aplicando as inactividades como "cortes" que podem dividir períodos.
- Carrega consultas existentes do dia (m_e_t) para esse dentista.
- Itera os períodos em incrementos de (duração escolhida). Para cada posição verifica sobreposição com consultas existentes via SlotOcupadoMin.
- Devolve array de slots com hora (HH:MM), duração em minutos, sala, tipo de consulta, posto.

5.1.1 Convenção HH.MM do VFP legacy

As tabelas agdactiv, agdinact e m_e_t guardam "hora" como decimal onde a parte inteira são horas e os primeiros 2 dígitos decimais são minutos. Exemplos: 9.30 = 9h30min (não 9,5 h); 12.3 = 12h30; 0.3 = 30 minutos. A aritmética do algoritmo converte tudo para minutos absolutos no início (via HhmmToMin) e só reconverte para HH:MM na apresentação (MinToStr).

```
FUNCTION HhmmToMin(tnHhmm)           && 12.30 -> 750
    LOCAL lnH, lnM
    lnH = INT(tnHhmm)
    lnM = INT((tnHhmm - lnH) * 100 + 0.5)
    RETURN lnH * 60 + lnM
ENDFUNC

FUNCTION MinToStr(tnMin)              && 750 -> "12:30"
    RETURN PADL(INT(tnMin/60),2,"0") + ":" + PADL(tnMin%60,2,"0")
ENDFUNC
```

5.2 DataDisponivel (180 LOC)

- Pesquisa agdactiv por (dentista, dia-da-semana) filtrando pela vigência do período (DtIni/DtFim).
- Ordena os períodos por hora de início e materializa em array [hora_ini, hora_fim, duração, sala, tipocons, idposto].
- Para cada período de inactividade do dia no agdinact, aplica uma das 3 transformações:
- A inactividade começa depois do período !’ corta terminus do período OU cria dois sub-períodos.
- A inactividade envolve todo o período !’ elimina o período.
- A inactividade sobrepõe o início !’ avança hora de início.

Se não sobrar nenhum período ou o dentista não trabalha no dia, retorna .F. (sem disponibilidade).

5.3 PodeGravar (244 LOC originais)

Validações aplicadas antes de gravar uma consulta. Retorna can_save + lista de reasons (code + message). Razões "hard" bloqueiam; razões "soft" apenas avisam.

Código	Significado	Severidade
PACIENTE_INEXISTENTE	Paciente não existe em pessoais	bloqueia
DENTISTA_INVALIDO	Dentista em falta ou inválido	bloqueia

DENTISTA_INEXISTENTE	Dentista não existe em pessoais	bloqueia
HORA_INVALIDA	Hora < 0 ou >= 24	bloqueia
SOBREPOSICAO	Conflito com consulta existente (m_e_t) — mesmo dentista ou mesma sala no intervalo	bloqueia
DATA_PASSADO_AVISO	Data no passado — requer permissão especial	aviso (soft)

5.4 GravaConsulta (246 LOC originais)

- Parseia JSON de entrada (episodio, data, hora, dentista, paciente, sala, duração, estado, obs, …).
- Chama PodeGravar primeiro. Se não pode, devolve os reasons directamente.
- Abre transacção SQL Server.
- Se episodio == 0 ou ausente !’ INSERT em m_e_t; caso contrário UPDATE.
- Para INSERT, obtém próximo episódio com SELECT ISNULL(MAX(episodio),0)+1 (replica a lógica legacy — idealmente seria IDENTITY).
- Commit ou rollback. Retorna { episodio, criado }.

5.5 ValidarNIF

Validação de NIF português de 9 dígitos. Primeiro dígito válido { 1,2,3,5,6,7,8,9 }. Checksum: soma cada dígito × peso (10 " posição) da posição 1 à 8. check = 11 " (soma mod 11). Se check "e 10 !’ check = 0. Válido sse check == dígito 9. Originalmente delega em ValidarNIFPT do PIFAP legacy se disponível (via EVALUATE para esconder a referência ao compilador); fallback inline sempre funcional.

5.6 CalcularOrcamento

- Lê a caixa (convenção) do paciente (pessoais.caixa).
- Para cada código de tratamento, JOIN tratesc + descpercent (percentagem de desconto por caixa).
- Aplica desconto por convenção ao preço; calcula IVA (23% default, sobrescrito pelo tratesc.iva).
- Soma totais: base, desconto, IVA, total.
- Retorna linhas detalhadas + totais.

5.7 CalcularHonorarios

Agrega tratamentos efectuados (m_e_t_tratamentos.efectuado = 1) por médico no intervalo dado. Retorna total e breakdown por código de tratamento.

5.8 Facturar e GerarRecibo

Wrappers que delegam em stored procedures SQL Server (up_factura_emitir, up_recibo_gerar). Se a SP não existir, retornam erro informativo — a lógica full do emiterecibo.prg legacy (1000+ LOC, dependente de MESSAGEBOX, cursor Pessoais do form principal e variáveis PRIVATE) não foi portada directamente porque está fortemente acoplada ao contexto UI. Recomenda-se criar as SPs ou criar um serviço VFP standalone separado.

5.9 SubmeterADSE

Marca claims na tabela adse_claims como enviadas (estado "E", data_envio, lote). A submissão real via NS_ADSE.dll (existente no VFP) ainda não está ligada — pode ser adicionada num cls_adse.prg dedicado.

6. Adaptações técnicas e gotchas do VFP 9

6.1 Sessão de dados isolada

Classes com base AS Session recebem a sua própria DataSession. Cursores criados dentro de uma instância Session não são visíveis ao chamador (o EVAL/SELECT falha). Por isso todos os helpers (clsJson, clsSql, clsAgendaRules, clsFactura, clsPaciente, clsConfig) foram definidos AS Custom. Só a classe OLEPUBLIC raiz (NewSoftAPI) permanece AS Session — para isolar clientes COM concorrentes.

6.2 RETURN dentro de TRY/CATCH

Em VFP 9, FUNCTION OLEPUBLIC não aceita RETURN dentro de TRY/CATCH — lança em runtime "RETURN/RETRY statement not allowed in TRY/CATCH". Solução: atribuir o valor a uma variável local lcRet e fazer RETURN lcRet depois do ENDTRY.

6.3 SET PROCEDURE TO com literal puxa cadeia inteira

SET PROCEDURE TO "C:\PRGS\x.prg" faz o compilador incluir essa .prg e todos os seus #INCLUDE e referências cruzadas no DLL. Contornar com variável: lcP = "C:\..."; SET PROCEDURE TO (lcP). Funções chamadas por nome (ex.: ENCRYPT) são também resolvidas em compile-time; esconder com EVALUATE("ENCRYPT(x)") quando não queremos forçar a inclusão.

6.4 Convenção HH.MM

Já explicado em 5.1.1 — sempre converter para minutos absolutos antes de fazer aritmética.

6.5 CREATE PROJECT NOSHOW bloqueia em modo headless

CREATE PROJECT em VFP9 via vfp9.exe /t ou via COM automation cria uma janela escondida que aguarda eventos que nunca chegam. Alternativa usada: criar o ficheiro .pjx directamente como DBF com o layout conhecido (28 campos; primeiro record é o header TYPE="H", depois um record por ficheiro TYPE="P"). Depois BUILD MTDLL ... FROM newsoftapi.pjx RECOMPILE funciona standalone. Ver C:\VFP\newsoftapi\build2.prg.

6.6 VFP é 32-bit — bridge para Node 64-bit

O VFP só compila DLLs 32-bit. Como o Node.js neste VPS é 64-bit, winax (binding nativo para COM) não consegue instanciar o objecto in-process. Também não foi possível instalar winax aqui (requer Python + VC++ Build Tools). Solução usada: um worker PowerShell 32-bit por slot do pool, que mantém a instância COM viva e aceita chamadas JSON por stdin/stdout. ~10-20% mais lento que winax puro mas 100% pure-JS + PowerShell nativo do Windows.

6.7 Build MT-DLL falha silenciosamente

Em modo headless, BUILD MTDLL não produz o ficheiro sem lançar erro explícito. O fallback usado é BUILD DLL (single-thread) que compila OK mas serializa chamadas. Com o pool de 3 workers PowerShell (cada um com a sua instância COM independente), a concorrência prática fica aceitável.

7. Como operar

7.1 Serviços Windows

Serviço	Arranca	Restart	Logs
nginx	C:\nginx\nginx.exe	Automático (NSSM)	C:\nginx\logs\access.log, error.log, svc-stderr.log
newsoft-backend	bun src/server.ts em C:\newsoft\app\backend	Automático (NSSM)	C:\newsoft\logs\backend-stdout.log, backend-stderr.log

7.2 Comandos úteis

```
# Estado
Get-Service nginx, newsoft-backend

# Reiniciar
Restart-Service nginx
Restart-Service newsoft-backend

# Logs em tempo real
Get-Content -Wait -Tail 50 C:\newsoft\logs\backend-stdout.log
Get-Content -Wait -Tail 50 C:\nginx\logs\error.log

# Recompilar DLL VFP (após alteracoes em C:\VFP\newsoftapi\*.prg)
#    requer 1 processo vfp9 livre
Remove-Item C:\VFP\newsoftapi\NewsoftAPI.dll,*.pjx,*.pjt -ErrorAction SilentlyContinue
& "C:\Program Files (x86)\Microsoft Visual FoxPro 9\vfp9.exe" /t /nologo /cC:
\VFP\newsoftapi\config.fpw C:\VFP\newsoftapi\build2.prg
& C:\Windows\SysWOW64\regsvr32.exe /s C:\VFP\newsoftapi\NewsoftAPI.dll
```

7.3 Checklist de sanidade

- curl -I https://nextgen.newsoftds.pt/ deve responder 301 para /prototipos/
- curl -X POST https://nextgen.newsoftds.pt/graphql -d '{"query":"{ adminTables { key }}"}' deve devolver a lista de tabelas admin
- curl http://localhost:4000/graphql responde directamente (backend local)
- Get-Service nginx, newsoft-backend deve mostrar Running em ambos
- C:\newsoft\logs\backend-stderr.log deve estar vazio ou com mensagens INFO

7.4 Ficha de contacto

Produto	NewSoft NextGen
VPS produção	nextgen.newsoftds.pt (135.125.79.108)
BD principal	ns_196196197 @ srv-clientscloud2.imaginasoft.pt:50036
Contacto	dolphinmargin@gmail.com
Versão doc.	1.0 — 2026-04-22