SRVGOLD - DOCUMENTAï¿‰ï¿‰O Tï¿‰CNICA

VISï¿‰O GERAL
O projeto SRVGold ï¿‰ uma API multi-tenant construï¿‰da com NestJS (https://nestjs.com/) e Prisma (https:/

ARQUITETURA E TECNOLOGIAS
ï¿‰ Runtime: Node.js 20 (requer >= 18).
ï¿‰ Framework: NestJS 11 com organizaï¿‰ï¿‰o modular (controllers, services, DTOs).
ï¿‰ ORM: Prisma 6 com dois schemas:
ï¿‰ prisma/schema.prisma: banco "main" com a tabela Tenant.
ï¿‰ prisma/erp.prisma: espelha a base legada (centenas de tabelas) para cada tenant.
ï¿‰ Banco de Dados: PostgreSQL multi-base. Hï¿‰ um banco principal com metadata (Tenant) e um banco p
ï¿‰ Configuraï¿‰ï¿‰o: Variï¿‰veis de ambiente carregadas via @nestjs/config. Conexï¿‰es construï¿‰da
ï¿‰ Middlewares: TenantContextMiddleware injeta automaticamente o cliente Prisma do tenant com base no h

ESTRUTURA DE PASTAS
Cï¿‰digo:

```
src/
+-- app.module.ts              // Registro global de mï¿‰dulos, providers e middleware
+-- main.ts                    // Bootstrap NestJS com CORS habilitado
+-- prisma.service.ts          // PrismaClient para o banco "main"
+-- tenant-prisma.manager.ts   // Factory de Prisma clients por tenant (lazy + cache)
+-- middleware/
ï¿‰   +-- tenant-context.middleware.ts
+-- tenants/
ï¿‰   +-- tenants.controller.ts   // POST /tenants
ï¿‰   +-- tenants.module.ts
ï¿‰   +-- tenants.service.ts      // Clona banco template + registra tenant
+-- t_cli/ ...                 // CRUD para clientes (exemplo de mï¿‰dulo funcional)
+-- t_emp/ ...                   // DTOs validados para empresas
+-- t_usere/ ...                 // CRUD da entidade ‘t_usere‘
+-- t_users/ ...                 // CRUD da entidade ‘t_users‘
```

Fim do cï¿‰digo
Outras pastas relevantes:
ï¿‰ prisma/: schemas Prisma e cï¿‰digo gerado (prisma/generated/erp).
ï¿‰ scripts/: automaï¿‰ï¿‰es em Node/TypeScript (por exemplo migrate-tenants.ts).
ï¿‰ docs/: artefatos de documentaï¿‰ï¿‰o (este documento + PDF gerado).

CONFIGURAï¿‰ï¿‰O DE AMBIENTE
1. Instalar dependï¿‰ncias
Cï¿‰digo:

```
npm install
```

Fim do cï¿‰digo
2. Gerar clientes Prisma (necessï¿‰rio apï¿‰s alterar schemas ou instalar o projeto):
Cï¿‰digo:

npx prisma generate
Fim do cï¿‰digo
3. Variï¿‰veis de ambiente (arquivo .env na raiz):
ï¿‰ DATABASE_URL: conexï¿‰o com o banco principal (main).
ï¿‰ ERP_TEMPLATE_URL: conexï¿‰o com o banco template padrï¿‰o usados em scripts.
ï¿‰ BASE_PG_URL: prefixo da URL PostgreSQL para montar conexï¿‰es dos tenants.
ï¿‰ PG_HOST, PG_PORT, PG_USER, PG_PASSWORD: credenciais administrativas usadas para criar/clona
ï¿‰ PG_SUPER_DB (opcional, padrï¿‰o postgres): banco acessado pelo usuï¿‰rio administrativo.
ï¿‰ PG_TEMPLATE (opcional, padrï¿‰o goldpdv): nome do banco template clonado.
ï¿‰ PORT (opcional): porta HTTP exposta pelo Nest (padrï¿‰o 3000).
> Seguranï¿‰a: mantenha credenciais fora do controle de versï¿‰o (adicionar .env no .gitignore) e rotacion

EXECUï¿‰ï¿‰O E SCRIPTS ï¿‰TEIS
ï¿‰ Desenvolvimento (watch mode):
Cï¿‰digo:
npm run start:dev
Fim do cï¿‰digo
ï¿‰ Execuï¿‰ï¿‰o simples (sem hot-reload):
Cï¿‰digo:
npm run start
Fim do cï¿‰digo
ï¿‰ Build de produï¿‰ï¿‰o + execuï¿‰ï¿‰o:
Cï¿‰digo:
npm run build && npm run start:prod
Fim do cï¿‰digo
ï¿‰ Testes:
Cï¿‰digo:
npm run test # unit
npm run test:e2e # end-to-end
npm run test:cov # cobertura
Fim do cï¿‰digo
ï¿‰ Migraï¿‰ï¿‰o dos tenants existentes (sincroniza schema ERP em todos os bancos):
Cï¿‰digo:
npm run migrate:tenants
Fim do cï¿‰digo
scripts/migrate-tenants.ts lï¿‰ todos os registros de Tenant no banco main, monta URLs com BASE_PG_UR

MULTI-TENANCY
1. Registro de tenant: POST /tenants chama TenantsService.createTenant, que:
ï¿‰ Valida duplicidade de slug (Tenant no banco main).
ï¿‰ Cria um banco novo via pg.Pool clonando o template (CREATE DATABASE ... WITH TEMPLATE ...).
ï¿‰ Registra name, slug e db_name na tabela Tenant.
2. Resoluï¿‰ï¿‰o do tenant em runtime:
ï¿‰ TenantContextMiddleware lï¿‰ x-tenant ou subdomï¿‰nio, busca o registro no banco main (PrismaServi
ï¿‰ tenantId (ID numï¿‰rico do tenant).

ï¿‰ tenantClient (instï¿‰ncia PrismaClient da base do tenant, providenciada pelo TenantPrismaManager).
ï¿‰ Os controllers recebem RequestWithPrisma e delegam aos services que usam o client apropriado.
3. Gerenciamento de conexï¿‰es:
ï¿‰ TenantPrismaManager memoiza Promise<PrismaClient> por dbName, evitando condiï¿‰ï¿‰es de corrid
ï¿‰ Conexï¿‰es sï¿‰o fechadas em onModuleDestroy, garantindo limpeza controlada na parada da aplicaï¿

Mï¿‰DULOS DE DOMï¿‰NIO (EXEMPLOS)
ï¿‰ t_users (src/t_users)
ï¿‰ CRUD completo sobre t_users (entidade legada com atributos como cdusu, deusu, email).
ï¿‰ DTOs (CreateTUsersDto, UpdateTUsersDto) com validaï¿‰ï¿‰es class-validator.

ï¿‰ t_usere (src/t_usere)
ï¿‰ Controlador e serviï¿‰o expï¿‰em operaï¿‰ï¿‰es CRUD sobre t_usere (relaciona usuï¿‰rios a empre
ï¿‰ DTOs atualizados para refletir restriï¿‰ï¿‰es do schema Prisma: id ï¿‰ UUID opcional, codusu e codem

ï¿‰ t_cli, t_emp, t_comanda, etc.
ï¿‰ Cada mï¿‰dulo segue padrï¿‰o Nest: Module registra controller/service, controllers recebem RequestW
ï¿‰ DTOs cobrem dezenas de campos herdados do ERP e utilizam decoradores class-validator para garantir

CAMADA DE DADOS
ï¿‰ Banco principal (Tenant): gerenciado por PrismaService (extends PrismaClient). Schema simples em prisr
ï¿‰ Banco ERP: schema extenso (prisma/erp.prisma) com centenas de modelos. Codegen disponï¿‰vel em
ï¿‰ Acesso: services nï¿‰o instanciam Prisma diretamente; recebem PrismaClient do middleware, garantindo

OBSERVABILIDADE E LOGS
ï¿‰ TenantPrismaManager configura logs de nï¿‰vel warn para o Prisma multi-tenant (pode ser ajustado).
ï¿‰ Sugestï¿‰o: integrar nestjs-pino ou interceptors de logging para rastrear tenant em cada requisiï¿‰ï¿‰o

BOAS PRï¿‰TICAS E PRï¿‰XIMOS PASSOS
ï¿‰ Validaï¿‰ï¿‰o adicional: adicionar guards/interceptors para verificar permissï¿‰es por tenant e auditoria
ï¿‰ Migrations vs db push: considerar o uso de prisma migrate para versionamento formal das mudanï¿‰as n
ï¿‰ Testes: expandir testes unitï¿‰rios/e2e cobrindo middleware e fluxo multi-tenant.
ï¿‰ Automaï¿‰ï¿‰o: incluir pipelines CI/CD que executem build, testes e script de migraï¿‰ï¿‰o em ambie
ï¿‰ Seguranï¿‰a: usar secrets manager (AWS Secrets Manager, Vault, etc.) e restringir acesso ao banco ten

---
Documento gerado automaticamente por Codex CLI.