Markdown

Resumo do Projeto

O Que Foi Implementado

Uma aplicação NestJS 11 completa e pronta para produção, com autenticação Better Auth e integração com Banco de Dados Oracle, seguindo todas as especificações do `NESTJS_SETUP_PROMPT.md`.

Estrutura Completa de Arquivos

```
api/
├── Arquivos de Configuração
│   ├── package.json              ☑ Todas as dependências configuradas
│   ├── tsconfig.json             ☑ TypeScript com módulos nodenext
│   ├── tsconfig.build.json       ☑ Configuração de Build
│   ├── nest-cli.json             ☑ NestJS CLI com SWC
│   ├── eslint.config.mjs         ☑ Configuração flat do ESLint v9
│   ├── .prettierrc               ☑ Configuração do Prettier
│   ├── .prettierignore           ☑ Padrões de ignore do Prettier
│   ├── .swcrc                    ☑ Configuração do compilador SWC
│   ├── .gitignore                ☑ Padrões de ignore do Git
│   └── typeorm.config.ts         ☑ Configuração da CLI do TypeORM
│
├── Documentação
│   ├── README.md                 ☑ Documentação abrangente
│   ├── SETUP.md                  ☑ Guia de configuração rápida
│   ├── GETTING_STARTED.md        ☑ Tutorial passo a passo
│   └── PROJECT_SUMMARY.md        ☑ Este arquivo
│
├── src/
│   ├── main.ts                   ☑ Entrada da aplicação com Swagger
│   ├── app.module.ts             ☑ Módulo raiz com todas as importações
│   │
│   ├── config/
│   │   └── oracle.config.ts      ☑ Configuração do banco de dados Oracle
│   │
│   ├── database/
│   │   ├── entities/             ☑ Todas as 8 entidades do Better Auth
│   │   │   ├── user.entity.ts
│   │   │   ├── session.entity.ts
│   │   │   ├── account.entity.ts
│   │   │   ├── verification.entity.ts
│   │   │   ├── organization.entity.ts
│   │   │   ├── member.entity.ts
│   │   │   ├── invitation.entity.ts
│   │   │   └── two-factor.entity.ts
│   │   │
│   │   └── migrations/
│   │       ├── 1760109600000-initial-schema.ts  ☑ Migração de exemplo
```

```
│   │       └── README.md                          ☑ Scripts SQL para configuração
manual
│   │
│   ├── modules/
│   │   ├── auth/
│   │   │   ├── auth.module.ts                     ☑ Módulo Better Auth
│   │   │   ├── auth-schema.ts                     ☑ Config do Better Auth com
init preguiçoso do DB
│   │   │   ├── decorators/
│   │   │   │   ├── public.decorator.ts            ☑ Decorator @Public()
│   │   │   │   ├── optional.decorator.ts          ☑ Decorator @Optional()
│   │   │   │   └── session.decorator.ts           ☑ Decorator @Session()
│   │   │   ├── guards/
│   │   │   │   └── auth.guard.ts                  ☑ Guard de Autenticação
│   │   │   └── hooks/
│   │   │       └── sign-up.hook.ts                ☑ Hook de validação de cadastro
│   │   │
│   │   └── users/
│   │       ├── users.module.ts                    ☑ Módulo de usuários
│   │       ├── users.controller.ts                ☑ CRUD + rotas protegidas
│   │       ├── users.service.ts                   ☑ Lógica de negócio de usuário
│   │       └── dto/
│   │           └── update-user.dto.ts             ☑ DTO com validação
│   │
│   └── common/
│       ├── health/
│       │   ├── health.module.ts                   ☑ Módulo de verificação de
saúde
│       │   └── health.controller.ts               ☑ Endpoint de saúde
│       ├── filters/
│       │   └── http-exception.filter.ts           ☑ Filtros de exceção
│       ├── interceptors/
│       │   └── transform.interceptor.ts           ☑ Transformador de resposta
│       └── pipes/
│           └── validation.pipe.ts                 ☑ Pipe de validação customizado
│
└── test/
    ├── app.e2e-spec.ts          ☑ Testes E2E
    └── jest-e2e.json            ☑ Config do Jest E2E
✨ Funcionalidades Implementadas
1. Framework Principal (Core)
☑ NestJS 11.x com TypeScript 5.x

☑ Suporte a Node.js 22.x

☑ Resolução de módulos moderna (nodenext)

☑ Suporte a Decorators e metadados

☑ Compilação SWC para builds rápidos

2. Integração de Banco de Dados
☑ Suporte a Oracle Database via TypeORM
```

☑ Kysely com OracleDialect para o Better Auth

☑ Pooling de conexão configurado

☑ Tratamento de fuso horário (UTC)

☑ 8 entidades TypeORM correspondentes ao schema do Better Auth

☑ Suporte a migrações

☑ Exemplo de migração e scripts SQL

3. Configuração do Better Auth
☑ Inicialização preguiçosa (lazy) do banco de dados (sem top-level await)

☑ Autenticação por Email/Senha

☑ Gerenciamento de sessão (7 dias, refresh de 1 dia)

☑ Todos os plugins necessários:

☑ openAPI - Documentação da API

☑ admin - Funcionalidade de Admin com impersonation (personificação)

☑ organization - Suporte a multi-tenancy (múltiplas organizações)

☑ twoFactor - Autenticação de dois fatores (2FA)

☑ (lastLoginMethod é nativo no núcleo do Better Auth)

4. Sistema de Autenticação
☑ AuthGuard para proteção de rotas

☑ Decorator @Public() para rotas públicas

☑ Decorator @Optional() para autenticação opcional

☑ Decorator @Session() para acessar a sessão do usuário

☑ Hook customizado de validação de cadastro (sign-up)

☑ Integração adequada com a Injeção de Dependência (DI) do NestJS

5. Funcionalidades da API
☑ Endpoints RESTful para gerenciamento de usuários

☑ Rotas protegidas e públicas

☑ Validação de entrada com class-validator

☑ DTOs para requisição/resposta

☑ Endpoints de verificação de saúde (Health check)

☑ Tratamento de erros adequado

☑ Filtros de exceção HTTP

6. Documentação da API
☑ Swagger UI em /api

☑ Especificação OpenAPI 3.0

☑ Documentação de autenticação Bearer

☑ Descrições e exemplos de rotas

☑ Endpoints organizados por tags

7. Qualidade de Código
☑ ESLint 9.x com configuração flat

☑ TypeScript ESLint v8.x

☑ Formatação Prettier 3.x

☑ Arquivos de ignore adequados

☑ Estilo de código consistente

8. Gerenciamento de Configuração
☑ @nestjs/config para variáveis de ambiente

☑ Suporte a arquivo .env

☑ Acesso type-safe às configurações

☑ Configurações específicas por ambiente

9. Configuração de Testes
☑ Configuração do Jest

☑ Exemplo de teste E2E

☑ Scripts de teste no package.json

☑ Relatório de cobertura (coverage)

10. Experiência do Desenvolvedor
☑ Hot-reload no desenvolvimento

☑ Suporte ao modo Debug

☑ Otimização de build de produção

☑ Mensagens de erro claras

☑ Documentação abrangente

🔌 Endpoints Disponíveis
Endpoints do Better Auth (Gerados automaticamente)
POST /api/auth/sign-up/email - Registro de usuário

POST /api/auth/sign-in/email - Login de usuário

POST /api/auth/sign-out - Logout

GET /api/auth/session - Obter sessão atual

POST /api/auth/two-factor/enable - Habilitar 2FA

POST /api/auth/two-factor/verify - Verificar 2FA

POST /api/auth/organization/create - Criar organização

GET /api/auth/organization/list - Listar organizações

E muito mais...

Endpoints Personalizados
GET /health - Verificação de saúde da aplicação

GET /users/public/health - Verificação de saúde pública

GET /users/me - Obter perfil do usuário atual (protegido)

GET /users - Listar todos os usuários (protegido)

GET /users/:id - Obter usuário por ID (protegido)

PATCH /users/:id - Atualizar usuário (protegido)

DELETE /users/:id - Deletar usuário (protegido)

📋 Comandos de Início Rápido
Bash

```
# Instalar dependências
pnpm install

# Iniciar servidor de desenvolvimento
pnpm start:dev

# Build para produção
pnpm build

# Iniciar servidor de produção
pnpm start:prod

# Formatar código
pnpm format
```

```
# Lint no código
pnpm lint

# Executar testes
pnpm test

# Executar migrações
pnpm migration:run
```

🔧 Configuração Necessária

Antes de executar, crie um arquivo .env com:

Snippet de código

```
NODE_ENV=development
PORT=3000
DB_HOST=127.0.0.1
DB_PORT=1521
DB_USERNAME=ROOT
DB_PASSWORD=sua_senha
DB_SERVICE_NAME=FPAPI
SESSION_SECRET=mude-isso-para-um-segredo-forte
```

🎓 Caminho de Aprendizado

Comece Aqui: Leia GETTING_STARTED.md para a configuração passo a passo.

Referência Rápida: Use SETUP.md para comandos comuns.

Mergulho Profundo: Leia README.md para documentação abrangente.

Exploração da API: Visite http://localhost:3000/api para a documentação Swagger.

Exemplos de Código: Revise src/modules/users para padrões de implementação.

🏗️ Destaques da Arquitetura

Inicialização Preguiçosa do Banco de Dados (Lazy Initialization)

A configuração do Better Auth usa um padrão de inicialização preguiçosa para evitar top-level await em CommonJS:

TypeScript

```
let dbInstance: Kysely<any> | null = null;

async function getDb() {
  if (!dbInstance) {
    // Inicializa o pool e a instância do Kysely
  }
  return dbInstance;
}
```

Configuração Dupla de Banco de Dados

TypeORM: Usado para gerenciamento de entidades e operações tradicionais de ORM.

Kysely: Usado exclusivamente pelo Better Auth para gerenciamento de sessão.

Ambos compartilham as mesmas credenciais de conexão Oracle.

Estrutura de Módulos
Módulos de Feature: Organizados por domínio (auth, users).

Módulo Common: Utilitários compartilhados (filtros, pipes, interceptors).

Módulo Config: Gerenciamento centralizado de configuração.

🔒 Funcionalidades de Segurança
☑ Hash de senha (gerenciado pelo Better Auth)

☑ Geração de token de sessão

☑ Configuração de CORS

☑ Validação de entrada em todos os endpoints

☑ Prevenção de injeção de SQL (queries parametrizadas)

☑ Proteção de variáveis de ambiente

☑ Suporte a 2FA

☑ Expiração de sessão

🚀 Funcionalidades Prontas para Produção
☑ Tratamento de erros e logging

☑ Endpoints de verificação de saúde

☑ Otimização de build de produção

☑ Configuração baseada em ambiente

☑ Pooling de conexão de banco de dados

☑ Tipagem TypeScript adequada em todo o projeto

☑ Documentação da API

☑ Suporte a migrações

☑ Middleware de validação

📊 Stack Tecnológico
Categoria Tecnologia Versão
Framework NestJS 11.x
Runtime Node.js 22.x
Linguagem TypeScript 5.x
Banco de Dados Oracle Database 19c+
ORM TypeORM Recente
Query Builder Kysely Recente
Auth Better Auth Recente
Validação class-validator 0.14.x

```
Docs    Swagger 11.x
Linter  ESLint  9.x
Formatador  Prettier    3.x

Exportar para as Planilhas

☑ Requisitos Atendidos
Todos os requisitos do NESTJS_SETUP_PROMPT.md foram implementados:

☑ NestJS 11 com TypeScript 5

☑ Integração com Oracle Database

☑ Configuração do TypeORM

☑ Better Auth com todos os plugins especificados

☑ Inicialização preguiçosa do banco de dados

☑ Todas as 8 entidades do Better Auth

☑ Guards e decorators de Auth

☑ Hooks customizados

☑ Documentação Swagger

☑ Configuração flat do ESLint 9

☑ Configuração do Prettier

☑ Gerenciamento de variáveis de ambiente

☑ Suporte a migrações

☑ Configuração de testes

☑ Estrutura de projeto completa

🎉 O Que Vem a Seguir?
Esta é uma fundação completa. Agora você pode:

Adicionar mais módulos de lógica de negócio

Implementar plugins adicionais do Better Auth (OAuth, etc.)

Adicionar mais entidades e relacionamentos

Implementar rate limiting (limite de requisições)

Adicionar logging com Winston ou Pino

Configurar pipeline de CI/CD
```

Adicionar testes mais abrangentes

Implementar cache com Redis

Adicionar funcionalidade de upload de arquivos

Deploy para produção

Notas

O arquivo .env está no gitignore - crie-o a partir do exemplo fornecido.

As tabelas do banco de dados precisam ser criadas antes da primeira execução.

O Oracle Instant Client pode ser necessário para desenvolvimento local.

Todo o código segue as melhores práticas do NestJS e TypeScript.

As melhores práticas de segurança estão documentadas no README.md.