Otantist Developer Guide

Environment Setup & Onboarding

Version: 1.0

Last Updated: January 2026

Table of Contents

1. Prerequisites

Required Software

[Table - see markdown source]

Recommended Tools

[Table - see markdown source]

VS Code Extensions

Install the recommended extensions when you open the project:

```
dbaeumer.vscode-eslint
```

```
esbenp.prettier-vscode
bradlc.vscode-tailwindcss
prisma.prisma
mikestead.dotenv
```

2. Quick Start

Step 1: Clone the Repository

```
git clone https://github.com/your-org/otantist.git
cd otantist
```

Step 2: Install Dependencies

```
npm install
```

Step 3: Set Up Environment Variables

```
# Copy the example environment file
cp .env.example .env

# Edit .env with your local settings
# Most defaults work for local development
```

Step 4: Start Docker Services

```
# Start PostgreSQL, Redis, and Mailhog
npm run docker:up

# Verify services are running
docker ps
```

You should see:

- `otantist-postgres` on port 5432
- `otantist-redis` on port 6379
- `otantist-mailhog` on ports 1025 (SMTP) and 8025 (Web UI)

Step 5: Initialize the Database

```
# Generate Prisma client
cd apps/api
```

```
npx prisma generate

# Run migrations
npx prisma migrate dev

# (Optional) Seed with test data
npm run db:seed
```

Step 6: Start Development Servers

```
# From root directory
cd ../..

# Start API server
npm run dev:api

# In another terminal, start web app
npm run dev:web
```

Step 7: Verify Everything Works

- **API**: http://localhost:3001/api/docs (Swagger UI)
- **Web**: http://localhost:3000
- **Mailhog**: http://localhost:8025 (view sent emails)
- **pgAdmin**: http://localhost:5050 (database admin)

3. Project Structure

```
otantist/
■■■ apps/
■   ■■■ api/                     # NestJS backend
■   ■   ■■■ prisma/
■   ■   ■   ■■■ schema.prisma    # Database schema
■   ■   ■   ■■■ migrations/      # Database migrations
■   ■   ■■■ src/
■   ■   ■   ■■■ auth/            # Authentication module
■   ■   ■   ■■■ users/           # User management
■   ■   ■   ■■■ preferences/     # User preferences
■   ■   ■   ■■■ messaging/       # 1:1 messaging
■   ■   ■   ■■■ moderation/      # Moderation tools
■   ■   ■   ■■■ parent-dashboard/
■   ■   ■   ■■■ prisma/          # Prisma service
■   ■   ■   ■■■ common/          # Shared utilities
■   ■   ■   ■■■ app.module.ts
■   ■   ■   ■■■ main.ts
■   ■   ■■■ test/
■   ■
■   ■■■ web/                     # Next.js web app
■   ■   ■■■ app/              # App router pages
■   ■   ■■■ components/
■   ■   ■■■ lib/
■   ■   ■■■ public/
■   ■
■   ■■■ mobile/               # React Native + Expo
■       ■■■ app/              # Expo router
■       ■■■ components/
■       ■■■ lib/
■
■■■ packages/
■   ■■■ shared/               # Shared types & constants
■   ■   ■■■ src/
■   ■       ■■■ types/
■   ■       ■■■ constants/
■   ■
■   ■■■ ui/                   # Shared UI components
■       ■■■ src/
■
■■■ scripts/                  # Utility scripts
■■■ docs/                     # Documentation
■■■ docker-compose.yml
■■■ package.json              # Root package.json (workspaces)
■■■ .env.example
```

4. Development Workflow

Branch Naming

```
feature/OT-123-add-calm-mode
bugfix/OT-456-fix-message-queue
hotfix/OT-789-security-patch
chore/update-dependencies
```

Commit Messages

Follow Conventional Commits:

```
feat(messaging): add time boundary enforcement
fix(auth): resolve token refresh race condition
docs(api): update swagger documentation
chore(deps): upgrade nestjs to v10.3
```

Pull Request Process

1. Create feature branch from `main`

2. Make changes with meaningful commits
3. Run tests: `npm test`
4. Run linting: `npm run lint`
5. Create PR with description
6. Request review
7. Squash and merge

5. Database Management

Prisma Commands

```
cd apps/api

# Generate Prisma client after schema changes
npx prisma generate

# Create a new migration
npx prisma migrate dev --name add_user_preferences

# Apply migrations (production)
npx prisma migrate deploy

# Reset database (WARNING: deletes all data)
npx prisma migrate reset

# Open Prisma Studio (database GUI)
npx prisma studio
```

Schema Changes Workflow

1. Edit `apps/api/prisma/schema.prisma`
2. Run `npx prisma migrate dev --name descriptive_name`
3. Prisma generates migration SQL and updates client
4. Commit both schema and migration files

Connecting to Database

Via psql:

```
psql postgresql://otantist:otantist_dev@localhost:5432/otantist_dev
```

Via pgAdmin:

- URL: http://localhost:5050
- Login: admin@otantist.local / admin
- Add server: host=postgres, port=5432, user=otantist

6. API Development

Creating a New Module

```
cd apps/api

# Generate module scaffolding
npx nest generate module feature-name
npx nest generate controller feature-name
npx nest generate service feature-name
```

API Documentation

- Swagger UI: http://localhost:3001/api/docs
- Use decorators to document endpoints:

```
@ApiTags('messaging')
@ApiOperation({ summary: 'Send a message' })
@ApiResponse({ status: 201, description: 'Message sent' })
@ApiBearerAuth()
@Post()
async sendMessage(@Body() dto: SendMessageDto) {
  // ...
}
```

Authentication

Protected routes use JWT:

```
import { UseGuards } from '@nestjs/common';
import { JwtAuthGuard } from '../auth/guards/jwt-auth.guard';

@UseGuards(JwtAuthGuard)
@Get('me')
async getProfile(@Request() req) {
  return req.user;
}
```

7. Testing

Running Tests

```
# All tests
npm test

# API tests only
npm test -w @otantist/api

# Watch mode
npm test -- --watch

# Coverage report
npm test -- --coverage
```

Test Structure

```
apps/api/
■■■ src/
■   ■■■ auth/
■       ■■■ auth.service.ts
■       ■■■ auth.service.spec.ts  # Unit test
■■■ test/
    ■■■ auth.e2e-spec.ts          # E2E test
    ■■■ jest-e2e.json
```

Writing Tests

```
describe('AuthService', () => {
  let service: AuthService;
  let prisma: PrismaService;

  beforeEach(async () => {
    const module = await Test.createTestingModule({
      providers: [AuthService, PrismaService],
    }).compile();

    service = module.get<AuthService>(AuthService);
    prisma = module.get<PrismaService>(PrismaService);
  });

  it('should register a new user', async () => {
    const result = await service.register({
      email: 'test@example.com',
      password: 'SecureP@ss123',
      inviteCode: 'TEST123',
      language: 'en',
    });

    expect(result.accountId).toBeDefined();
  });
});
```

8. Coding Standards

TypeScript

- Strict mode enabled
- No `any` types (use `unknown` if needed)
- Explicit return types on functions
- Use interfaces over types when possible

Formatting

- Prettier handles formatting
- 2 space indentation
- Single quotes
- Trailing commas

Naming Conventions

[Table - see markdown source]

Bilingual Content

All user-facing strings must support FR/EN:

```
// ■ Bad
throw new BadRequestException('Invalid email');

// ■ Good
throw new BadRequestException({
  code: 'INVALID_EMAIL',
  message_en: 'Invalid email address',
  message_fr: 'Adresse courriel invalide',
});
```

9. Troubleshooting

Docker Issues

Containers won't start:

```
# Check logs
docker-compose logs postgres
docker-compose logs redis

# Restart containers
npm run docker:down
npm run docker:up
```

Port already in use:

```
# Find process using port 5432
lsof -i :5432

# Kill it or change port in docker-compose.yml
```

Database Issues

Migration fails:

```
# Reset database (development only!)
cd apps/api
npx prisma migrate reset

# Or fix manually
npx prisma migrate resolve --rolled-back migration_name
```

Prisma client out of sync:

```
npx prisma generate
```

Node/npm Issues

Dependency conflicts:

```
# Clear everything and reinstall
npm run clean
rm package-lock.json
npm install
```

Wrong Node version:

```
# Use nvm to switch
nvm use 20
```

10. Useful Commands

Quick Reference

```
# Start all services
npm run docker:up
npm run dev:api
npm run dev:web

# Database
npm run db:migrate      # Run migrations
npm run db:studio       # Open Prisma Studio
npm run db:seed         # Seed test data

# Testing
npm test               # Run all tests
npm run lint           # Lint all code

# Docker
npm run docker:up      # Start containers
npm run docker:down    # Stop containers
npm run docker:logs    # View logs

# Build
npm run build          # Build all apps
npm run build:api      # Build API only
```

Environment URLs

[Table - see markdown source]

Need Help?

- Check the `docs/` folder for additional documentation
- Review existing code for patterns
- Ask in the team Slack channel
- Create a GitHub issue for bugs

Happy coding! ■