



Configurare Backend CAD - Ghid Complet

[npm start](#)

Cuprins

1. [Pregătirea Mediului](#)
 2. [Instalarea Node.js](#)
 3. [Download și Instalare Teigha](#)
 4. [Setup Proiect Backend](#)
 5. [Configurare și Testare](#)
 6. [Rularea cu npm start](#)
 7. [Integrarea cu Frontend](#)
 8. [Troubleshooting](#)
-



1. Pregătirea Mediului

Pentru Windows:

bash

Verifică dacă ai Git instalat

git --version

Dacă nu ai Git, downloadează de la:

<https://git-scm.com/download/win>

Verifică PowerShell sau Command Prompt

Deschide PowerShell ca Administrator

Pentru macOS:

bash

```
# Verifică dacă ai Homebrew
```

```
brew --version
```

```
# Dacă nu ai Homebrew, instalează:
```

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

```
# Instalează Git dacă nu e instalat
```

```
brew install git
```

Pentru Linux (Ubuntu/Debian):

```
bash
```

```
# Update sistem
```

```
sudo apt update && sudo apt upgrade -y
```

```
# Instalează utilitățile necesare
```

```
sudo apt install -y curl wget git build-essential
```

2. Instalarea Node.js

Windows:

```
bash
```

```
# Opțiunea 1: Download direct
```

```
# Mergi la: https://nodejs.org/
```

```
# Download versiunea LTS (18.x sau 20.x)
```

```
# Rulează installer-ul
```

```
# Opțiunea 2: Cu Chocolatey (dacă ai)
```

```
choco install nodejs
```

```
# Verifică instalarea
```

```
node --version
```

```
npm --version
```

macOS:

```
bash
```

```
# Cu Homebrew (recomandat)
```

```
brew install node
```

```
# Sau cu installer direct de pe nodejs.org
```

```
# Verifică instalarea
```

```
node --version
```

```
npm --version
```

Linux:

```
bash
```

```
# Ubuntu/Debian - NodeSource repository
```

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
```

```
sudo apt-get install -y nodejs
```

```
# Verifică instalarea
```

```
node --version
```

```
npm --version
```

```
# Dacă ai probleme cu permisiunile npm
```

```
sudo chown -R $(whoami) ~/.npm
```

3. Download și Instalare Teigha Teigha

Step 1: Download Teigha File Converter

```
bash
```

```
# Mergi la: https://www.opendesign.com/guestfiles/TeighaFileConverter
```

```
# Înregistrează-te gratuit (doar email)
```

```
# Download versiunea pentru sistemul tău:
```

```
# - Windows: TeighaFileConverter_QnxX64_4.3.2_vc16.zip
```

```
# - Mac: TeighaFileConverter_Mac_4.3.2.dmg
```

```
# - Linux: TeighaFileConverter_lnxX64_4.3.2.tgz
```

Windows - Instalare Teigha:

```
bash
```

```
# 1. Extrage ZIP-ul în C:\Program Files\ODA\  
# 2. Structura finală:  
# C:\Program Files\ODA\TeighaFileConverter\TeighaFileConverter.exe  
  
# 3. Adaugă în PATH (optional):  
# - Deschide System Properties > Environment Variables  
# - Adaugă în PATH: C:\Program Files\ODA\TeighaFileConverter  
  
# 4. Test instalare  
"C:\Program Files\ODA\TeighaFileConverter\TeighaFileConverter.exe" --help
```

macOS - Instalare Teigha:

```
bash  
  
# 1. Montează DMG-ul și copiază aplicația  
sudo cp -R "/Volumes/TeighaFileConverter/TeighaFileConverter.app" /Applications/  
  
# 2. Creează symlink pentru command line  
sudo ln -sf "/Applications/TeighaFileConverter.app/Contents/MacOS/TeighaFileConverter" /usr/local/bin/TeighaFileCon  
  
# 3. Test instalare  
TeighaFileConverter --help
```

Linux - Instalare Teigha:

```
bash  
  
# 1. Extrage arhiva  
tar -xzf TeighaFileConverter_InxX64_4.3.2.tgz  
  
# 2. Mută în /opt/  
sudo mv TeighaFileConverter_InxX64_4.3.2 /opt/teigha  
  
# 3. Creează symlink  
sudo ln -sf /opt/teigha/TeighaFileConverter /usr/local/bin/TeighaFileConverter  
  
# 4. Test instalare  
TeighaFileConverter --help
```



4. Setup Proiect Backend

Creează structura proiectului:

```
bash

# Creează directorul principal
mkdir cad-converter-backend
cd cad-converter-backend

# Creează structura de foldere
mkdir -p {src,config,uploads,converted,logs,public}
```

Inițializează proiectul Node.js:

```
bash

# Inițializează package.json
npm init -y

# Instalează dependențele necesare
npm install express multer cors helmet express-rate-limit winston uuid fs-extra
npm install --save-dev nodemon
```

Creează package.json optimizat:

```
json
```

```
{
  "name": "cad-converter-backend",
  "version": "1.0.0",
  "description": "Backend pentru CAD Converter cu Teigha File Converter",
  "main": "server.js",
  "scripts": {
    "start": "node server.js",
    "dev": "nodemon server.js",
    "test": "node test.js"
  },
  "dependencies": {
    "express": "^4.18.2",
    "multer": "^1.4.5",
    "cors": "^2.8.5",
    "helmet": "^6.1.5",
    "express-rate-limit": "^6.7.0",
    "winston": "^3.8.2",
    "uuid": "^9.0.0",
    "fs-extra": "^11.1.1"
  },
  "devDependencies": {
    "nodemon": "^2.0.22"
  },
  "engines": {
    "node": ">= 16.0.0"
  }
}
```

Creează fișierul principal server.js:

javascript

```
const express = require('express');
const cors = require('cors');
const helmet = require('helmet');
const rateLimit = require('express-rate-limit');
const multer = require('multer');
const path = require('path');
const fs = require('fs-extra');
const { spawn } = require('child_process');
const { v4: uuidv4 } = require('uuid');

const app = express();
const PORT = process.env.PORT || 3000;

// Configurarea Teigha
const TEIGHA_PATHS = {
  win32: 'C:\\Program Files\\ODA\\TeighaFileConverter\\TeighaFileConverter.exe',
  darwin: '/usr/local/bin/TeighaFileConverter',
  linux: '/usr/local/bin/TeighaFileConverter'
};

const TEIGHA_PATH = process.env.TEIGHA_PATH || TEIGHA_PATHS[process.platform];

// Middleware
app.use(helmet());
app.use(cors());

// Rate limiting
const limiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minute
  max: 20, // max 20 cereri per IP
  message: { error: 'Prea multe cereri. Încearcă din nou în 15 minute.' }
});

app.use('/api/', limiter);
app.use(express.json({ limit: '10mb' }));
app.use(express.urlencoded({ extended: true, limit: '10mb' }));

// Configurare multer pentru upload
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, './uploads');
  },
  filename: function (req, file, cb) {
```


```
    const uniqueName = uuidv4() + path.extname(file.originalname);
    cb(null, uniqueName);
  }
});
```

```
const upload = multer({
  storage: storage,
  fileFilter: (req, file, cb) => {
    const allowedTypes = ['.dwg', '.dxf'];
    const fileExt = path.extname(file.originalname).toLowerCase();
    if (allowedTypes.includes(fileExt)) {
      cb(null, true);
    } else {
      cb(new Error(`Tipul de fișier ${fileExt} nu este suportat`), false);
    }
  },
  limits: {
    fileSize: 100 * 1024 * 1024 // 100MB
  }
});
```

// Asigură că directoarele există

```
async function ensureDirectories() {
  const dirs = ['./uploads', './converted', './logs', './public'];
  for (const dir of dirs) {
    await fs.ensureDir(dir);
  }
}
```

// Funcție pentru conversie DWG cu Teigha

```
async function convertDWGtoDXF(inputPath, outputDir) {
  return new Promise((resolve, reject) => {
    console.log( Încep conversia cu Teigha: ${inputPath});

    const args = [
      path.dirname(inputPath), // input directory
      outputDir,               // output directory
      'ACAD2018',              // output version
      'DXF',                   // output format
      '0',                     // recurse subdirectories
      '1',                     // audit and fix
      '*.dwg'                  // input filter
    ];
```



```

console.log(`Comanda Teigha: ${TEIGHA_PATH} ${args.join(' ')}`);

const teighaProcess = spawn(TEIGHA_PATH, args, {
  stdio: ['ignore', 'pipe', 'pipe'],
  timeout: 120000 // 2 minute timeout
});

let stdout = "";
let stderr = "";

teighaProcess.stdout.on('data', (data) => {
  stdout += data.toString();
  console.log('Teigha stdout:', data.toString());
});

teighaProcess.stderr.on('data', (data) => {
  stderr += data.toString();
  console.log('Teigha stderr:', data.toString());
});

teighaProcess.on('close', (code) => {
  console.log(`Teigha închis cu codul: ${code}`);
  if (code === 0) {
    resolve({ stdout, stderr });
  } else {
    reject(new Error(`Teigha a eșuat cu codul ${code}: ${stderr}`));
  }
});

teighaProcess.on('error', (error) => {
  console.error('Eroare pornire Teigha:', error);
  reject(new Error(`Nu s-a putut porni Teigha: ${error.message}`));
});
}

// Rută de health check
app.get('/api/health', async (req, res) => {
  try {
    // Test dacă Teigha este disponibil
    const testProcess = spawn(TEIGHA_PATH, ['--help'], {
      stdio: 'pipe',
      timeout: 5000
    });
  }

```

```

testProcess.on('close', (code) => {
  res.json({
    success: true,
    status: 'healthy',
    teigha: {
      installed: code === 0,
      path: TEIGHA_PATH,
      version: 'ODA 4.3.2'
    },
    timestamp: new Date().toISOString()
  });
});

testProcess.on('error', () => {
  res.status(503).json({
    success: false,
    status: 'unhealthy',
    error: 'Teigha File Converter nu este disponibil',
    teigha: {
      installed: false,
      path: TEIGHA_PATH
    },
    timestamp: new Date().toISOString()
  });
});

} catch (error) {
  res.status(503).json({
    success: false,
    status: 'unhealthy',
    error: error.message,
    timestamp: new Date().toISOString()
  });
}
});

// Rută pentru conversie DWG la DXF
app.post('/api/convert/dwg-to-dxf', upload.single('dwgFile'), async (req, res) => {
  let tempFilePath = null;
  let outputDir = null;

  try {
    if (!req.file) {

```

```
return res.status(400).json({
  success: false,
  error: 'Nu a fost încărcat niciun fișier'
});
}

tempFilePath = req.file.path;
outputDir = path.join('./converted', uuidv4());

console.log(` 📁 Procesez fișierul: ${req.file.originalname}`);
console.log(` 📁 Calea temporară: ${tempFilePath}`);
console.log(` 📁 Directorul de output: ${outputDir}`);

await fs.ensureDir(outputDir);

// Conversie cu Teigha
await convertDWGtoDXF(tempFilePath, outputDir);

// Găsește fișierul DXF generat
const files = await fs.readdir(outputDir);
console.log(` 📁 Fișiere generate:`, files);

const dxfFile = files.find(f => f.toLowerCase().endsWith('.dxf'));

if (!dxfFile) {
  throw new Error('Fișierul DXF nu a fost generat de Teigha');
}

const dxfPath = path.join(outputDir, dxfFile);
const outputFilename = req.file.originalname.replace(/\.dwg$/i, '.dxf');

console.log(` ✅ DXF generat cu succes: ${dxfPath}`);

// Trimite fișierul DXF
res.download(dxfPath, outputFilename, async (err) => {
  if (err) {
    console.error(' ❌ Eroare download:', err);
  } else {
    console.log(` 📁 DXF trimis cu succes: ${outputFilename}`);
  }
});

// Cleanup
try {
  await fs.remove(tempFilePath);
}
```

```

        await fs.remove(outputDir);
        console.log(' 🗑 Cleanup complet');
    } catch (cleanupError) {
        console.error(' ⚠ Eroare cleanup:', cleanupError);
    }
});

} catch (error) {
    console.error(' ❌ Eroare conversie:', error);

    // Cleanup la eroare
    if (tempFilePath) {
        try { await fs.remove(tempFilePath); } catch(e) {}
    }
    if (outputDir) {
        try { await fs.remove(outputDir); } catch(e) {}
    }

    res.status(500).json({
        success: false,
        error: error.message
    });
}

});

// Servire fișiere statice (frontend)
app.use(express.static('./public'));

// Catch-all pentru SPA
app.get('*', (req, res) => {
    if (req.path.startsWith('/api/')) {
        return res.status(404).json({ error: 'API endpoint nu a fost găsit' });
    }
    res.sendFile(path.join(__dirname, 'public', 'index.html'));
});

// Error handler global
app.use((error, req, res, next) => {
    console.error(' ❌ Server error:', error);

    if (error instanceof multer.MulterError) {
        if (error.code === 'LIMIT_FILE_SIZE') {
            return res.status(400).json({
                success: false,

```

```

        error: 'Fișierul este prea mare. Dimensiunea maximă: 100MB'
    });
}
}

res.status(500).json({
  success: false,
  error: process.env.NODE_ENV === 'production'
    ? 'Eroare internă server'
    : error.message
});
});

// Pornire server
async function startServer() {
  try {
    await ensureDirectories();
    console.log('📁 Directoarele au fost create/verificate');

    app.listen(PORT, () => {
      console.log('🚀 =====');
      console.log(`🚀 CAD Converter Backend PORNIT!`);
      console.log(`🚀 Port: ${PORT}`);
      console.log(`🚀 Teigha: ${TEIGHA_PATH}`);
      console.log(`🚀 API Health: http://localhost:${PORT}/api/health`);
      console.log(`🚀 Frontend: http://localhost:${PORT}`);
      console.log('🚀 =====');
    });

  } catch (error) {
    console.error('❌ Eroare pornire server:', error);
    process.exit(1);
  }
}

// Cleanup la închidere
process.on('SIGINT', () => {
  console.log(`\n👋 Închidere server...`);
  process.exit(0);
});

startServer();

```

Creează fișier de test (test.js):

javascript

```
const { spawn } = require('child_process');
const path = require('path');

// Testează dacă Teigha este instalat și funcțional
function testTeigha() {
  const TEIGHA_PATHS = {
    win32: 'C:\\Program Files\\ODA\\TeighaFileConverter\\TeighaFileConverter.exe',
    darwin: '/usr/local/bin/TeighaFileConverter',
    linux: '/usr/local/bin/TeighaFileConverter'
  };

  const TEIGHA_PATH = process.env.TEIGHA_PATH || TEIGHA_PATHS[process.platform];

  console.log('🔧 Testare Teigha File Converter...');
  console.log('📍 Calea: ${TEIGHA_PATH}');
  console.log('🖥 Platform: ${process.platform}');

  const testProcess = spawn(TEIGHA_PATH, ['--help'], {
    stdio: 'pipe'
  });

  testProcess.on('close', (code) => {
    if (code === 0) {
      console.log('✅ Teigha File Converter este instalat și funcțional!');
      console.log('🚀 Poți rula: npm start');
    } else {
      console.log('❌ Teigha a returnat codul: ${code}');
      console.log('📋 Verifică instalarea Teigha File Converter');
    }
  });

  testProcess.on('error', (error) => {
    console.log('❌ Eroare testare Teigha:', error.message);
    console.log('📋 Soluții posibile:');
    console.log('  1. Verifică dacă Teigha este instalat la calea corectă');
    console.log('  2. Setează variabila TEIGHA_PATH cu calea corectă');
    console.log('  3. Adaugă Teigha în PATH-ul sistemului');
  });

  testProcess.stdout.on('data', (data) => {
    console.log('📄 Teigha output:', data.toString().substring(0, 200) + '...');
  });
}
```


```
testTeigha();
```

5. Configurare și Testare

Testează instalarea Teigha:

```
bash

# În directorul proiectului
npm run test

# Output așteptat:
# ☒ Teigha File Converter este instalat și funcțional!
#  Poți rula: npm start
```

Dacă ai erori cu Teigha:

```
bash

# Windows - setează calea custom
set TEIGHA_PATH="C:\path\to\your\TeighaFileConverter.exe"

# macOS/Linux - setează calea custom
export TEIGHA_PATH="/path/to/your/TeighaFileConverter"

# Testează din nou
npm run test
```

Copiază frontend-ul în directorul public:

```
bash

# Salvează aplicația frontend ca index.html în directorul public/
# Sau copiază din aplicația noastră anterioară
```

6. Rularea cu `npm start`

Pornește backend-ul:

```
bash
```



```
# În directorul cad-converter-backend
```

```
npm start
```

```
# Output așteptat:
```

```
# 🚀 =====
```

```
# 🚀 CAD Converter Backend PORNIT!
```

```
# 🚀 Port: 3000
```

```
# 🚀 Teigha: /usr/local/bin/TeighaFileConverter
```

```
# 🚀 API Health: http://localhost:3000/api/health
```

```
# 🚀 Frontend: http://localhost:3000
```

```
# 🚀 =====
```

Pentru development cu auto-restart:

```
bash
```

```
npm run dev
```

```
# Backend-ul va reporni automat la modificări
```

Testează backend-ul:

```
bash
```

```
# Test health check
```

```
curl http://localhost:3000/api/health
```

```
# Output așteptat:
```

```
# {
```

```
#   "success": true,
```

```
#   "status": "healthy",
```

```
#   "teigha": {
```

```
#     "installed": true,
```

```
#     "path": "/usr/local/bin/TeighaFileConverter",
```

```
#     "version": "ODA 4.3.2"
```

```
#   }
```

```
# }
```



7. Integrarea cu Frontend

Copiază frontend-ul optimizat:

html

```
<!-- public/index.html -->
<!DOCTYPE html>
<html lang="ro">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CAD Converter - Backend Ready</title>
  <style>
    /* Include CSS-ul din aplicația noastră anterioare */
  </style>
</head>
<body>
  <!-- Include HTML-ul din aplicația noastră anterioare -->

  <!-- JavaScript actualizat pentru backend real -->
  <script>
    // Frontend-ul va detecta automat că rulează pe server
    // și va folosi backend-ul real pentru conversii DWG
  </script>
</body>
</html>
```

Testează aplicația completă:

bash

1. Asigură-te că backend-ul rulează

npm start

2. Deschide browser la:

http://localhost:3000

3. Verifică statusul în colțul din dreapta sus:

Ar trebui să vezi: "  Teigha [versiune] Online"

4. Testează cu un fișier DWG real



8. Troubleshooting

Probleme comune și soluții:

Eroarea: "Cannot find module"

```
bash

# Reinstalează dependențele
rm -rf node_modules package-lock.json
npm install
```

Eroarea: "Teigha File Converter nu este disponibil"

```
bash

# Verifică calea Teigha
which TeighaFileConverter # Linux/Mac
where TeighaFileConverter # Windows

# Setează calea manual
export TEIGHA_PATH="/calea/ta/custom/TeighaFileConverter"
npm start
```

Eroarea: "Permission denied"

```
bash

# Linux/Mac - setează permisiuni
chmod +x /usr/local/bin/TeighaFileConverter

# Windows - rulează ca Administrator
```

Eroarea: "Port 3000 already in use"

```
bash

# Schimbă portul
PORT=3001 npm start

# Sau omoară procesul existent
# Windows:
netstat -ano | findstr :3000
taskkill /PID [PID] /F

# Linux/Mac:
lsof -ti:3000 | xargs kill
```

Backend-ul pornește dar nu convertește:

```
bash

# Verifică logs-urile în timp real
npm run dev

# Încearcă o conversie și urmărește output-ul
# Logs-urile vor arăta exact ce se întâmplă cu Teigha
```

Debug pas cu pas:

```
bash

# 1. Verifică Node.js
node --version # Ar trebui >= 16.0.0

# 2. Verifică Teigha
npm run test

# 3. Verifică dependențele
npm list

# 4. Pornește cu logging detaliat
DEBUG=* npm start

# 5. Testează API-ul manual
curl -X GET http://localhost:3000/api/health
```

Structura finală a proiectului:

```
cad-converter-backend/
├── package.json
├── server.js
├── test.js
├── uploads/      (generat automat)
├── converted/    (generat automat)
├── logs/         (generat automat)
├── public/
│   └── index.html (frontend-ul tău)
```

Verificare Finală

Checklist complet:

- ☐ ☒ Node.js instalat (16.0.0+)
- ☐ ☒ Teigha File Converter instalat și funcțional
- ☐ ☒ `npm run test` returnează succes
- ☐ ☒ `npm start` pornește server-ul fără erori
- ☐ ☒ `http://localhost:3000/api/health` returnează `"installed": true`
- ☐ ☒ Frontend-ul se încarcă la `http://localhost:3000`
- ☐ ☒ Status indicator arată "🌐 Backend Online"
- ☐ ☒ Conversiile DWG → DXF funcționează real

Success! 🎉

bash

echo "🚀 FELICITĂRI! Backend-ul CAD cu Teigha funcționează!"

echo "🌐 Aplicația ta este disponibilă la: http://localhost:3000"

echo "🔄 Pentru auto-restart în development: npm run dev"

echo "📊 Pentru monitoring: urmărește logs-urile în terminal"

Acum ai un backend complet funcțional cu Teigha File Converter! Încearcă să pornești cu `npm start` și spune-mi la ce pas ai nevoie de ajutor. 🚀