Documentation API - AutoCalc OptiDevis v2.5.2

📋 Table des matières

Vue d'ensemble

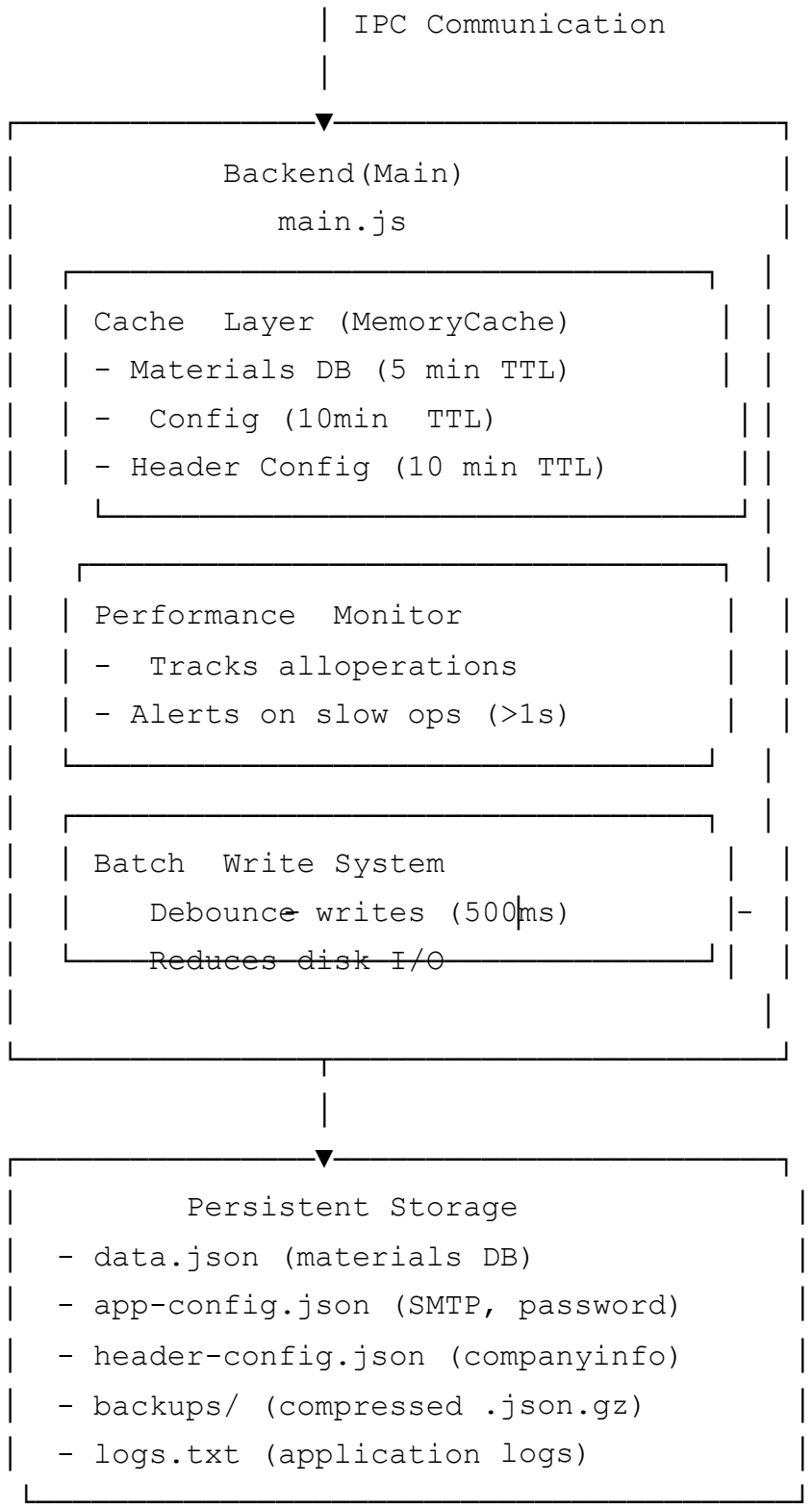
AutoCalc OptiDevis est une application Electron pour la gestion de devis et calculs dans le secteur du bâtiment.

Stack technique :

- **Runtime** : Electron 38.2.1 + Node.js
- **Base de données** : LowDB (JSON le-based)
- **Authenti cation** : bcryptjs
- **Email** : Nodemailer
- **PDF** : jsPDF + jsPDF-autotable
- **Logging** : Winston (logs structurés)
- **Tests** : Jest + Testing Library

Architecture

```
┌──────────────────────────────────────────┐
│  Frontend (Renderer)  │  index.html + app.js │
└──────────────────┬──────────────────────┘
```



Handlers IPC

Tous les handlers suivent le pattern :

```
ipcMain.handle('handler-name', async (event, args) =>{
try {
    // Logic
    return { success: true, data: result };
```

```
  } catch (error) {
     return { success: false, message: error.message };
    }
});
```

🔐 Authenti cation

verify-password

verify-password

Véri e le mot de passe administrateur.

Paramètres :

- password (string) : Mot de passe à véri er

Retour :

```
{
  success: boolean,
 message: string
}
```

Exemple :

```
const result = await window.api.verifyPassword('monMotDePasse');
if (result.success) {
  console.log('Authentifié !');
}
```

---

set-password

set-password

Dé nit un nouveau mot de passe administrateur (hashé avec bcrypt).

Paramètres :

- newPassword (string) : Nouveau mot de passe

Retour :

```
{
  success: boolean,
```

```
  message: string
}
```

Sécurité :

- Hash bcrypt avec salt rounds = 10
- Sauvegarde sécurisée dans app-con g.json

Email

~~test-email-config~~
test-email-config

Teste la con guration SMTP.

Paramètres :

```
{
  service: 'gmail',
  auth: {
    user: 'email@example.com',
    pass: 'password'
  }
}
```

Retour :

```
{
  success: boolean,
 message: string
}
```

~~send-quote-email~~
send-quote-email

Envoie un devis par email avec pièce jointe PDF.

Paramètres :

```
{
  to: 'client@example.com',
  subject: 'Votre devis',
 body: 'Bonjour, veuillez trouver...',
```

```
  attachment: {
    filename: 'devis.pdf',
    content: 'base64EncodedPDF'
  }
}
```

Retour :

```
{
  success: boolean,
 message?: string
}
```

Notes :

- Utilise la con guration SMTP dé nie dans app-con g.json
- L'expéditeur est automatiquement dé ni depuis header-con g.json

---

Base de données (Matériaux)

get-materials

Récupère la liste complète des matériaux.

Paramètres : Aucun

Retour :

```
{
  success: boolean,
 data: Material[]
}
```

Type Material :

```
interface Material {
id:string;
 name:  string;
price:   number;
 barcode?:string;
category?:string;
 unit?:  string;
```

```
  // ... autres champs
}
```

Performance :

- Utilise le cache (TTL: 5 minutes)
- Si cache hit : ~0.1ms
- Si cache miss : ~50-100ms (lecture disque + parse JSON)

---

update-materials

Met à jour la liste des matériaux (écrasement complet).

Paramètres :

- materials (Material[ ]) : Nouvelle liste de matériaux

Retour :

```
{
  success: boolean,
 message?: string
}
```

Comportement :

. Invalide le cache
. Plani e une écriture batch (debounce 500ms)
. Reconstruit l'index de recherche
. Crée un backup automatique (compressé, uniquement si changements)

Index de recherche :

```
materialsIndex = {
  byId: Map<string, Material>,          // O(1) lookup par ID
 byBarcode: Map<string, Material>,      // O(1) lookup par code-barre
byName: Map<string, Material[]>         // O(1) lookup par nom
}
```

---

import-data

Importe des matériaux depuis un chier JSON.

Paramètres : Aucun (ouvre une boîte de dialogue)

Format attendu :

```
{
  "materials": [
    { "name": "Ciment", "price": 12.50 },
    { "name": "Sable", "price": 8.00 }
  ]
}
```

Validation :

- Véri e la structure JSON
- Valide que `materials` est un tableau
- Valide que chaque matériau a `name` (string) et `price` (number)

Retour :

```
{
  success: boolean,
 message?: string
}
```

---

export-data

Exporte tous les matériaux vers un chier JSON.

Paramètres : Aucun (ouvre une boîte de dialogue de sauvegarde)

Retour :

```
{
  success: boolean,
 path?: string,
message?: string
}
```

⚙ Con guration

get-config

Récupère la con guration de l'application (sans le mot de passe admin).

Paramètres : Aucun

Retour :

```
{
  success: boolean,
  data: {
    smtp: {
      service: string,
      auth: { user: string, pass: string }
    },
    dbPath: string,
    appConfigPath: string,
    backupDir: string
  }
}
```

Sécurité : Le champ `adminPassword` est automatiquement exclu.

---

set-config

Met à jour la con guration de l'application.

Paramètres :

```
{
  smtp?: {
    service: string,
    auth: { user: string, pass: string }
 }
  // Autres champs optionnels
}
```

Retour :

```
{
  success: boolean,
 message?: string
}
```

Note : Si `adminPassword` n'est pas fourni, le hash existant est préservé.

---

get-header-config

Récupère la con guration de l'en-tête des devis.

Paramètres : Aucun

Retour :

```
{
  success: boolean,
  data: {
    company: {
      name: string, address:
      string, phone: string,
      email: string, siret:
      string, vatNumber:
      string, status: string,
      paymentTerms: string,
      additionalTerms: string,
      mentionslegales: string


    }

}
}
```

---

set-header-config

Met à jour la con guration de l'en-tête.

Paramètres :

```
{
  company: {
    name: string,
    address: string,
    // ... autres champs
  }
}
```

Retour :

```
{
  success: boolean,
 message?: string
}
```

💾 Sauvegarde et restauration

`restore-backup`

Restaure la base de données depuis un chier de backup.

Paramètres : Aucun (ouvre une boîte de dialogue)

Formats acceptés :

- `data_*.json` (backup non compressé, ancien format)
- `data_*.json.gz` (backup compressé, nouveau format)

Retour :

```
{
  success: boolean,
 message?: string
}
```

Validation :

- Véri e la structure JSON
- Valide la présence de `materials` (array)
- Recharge la DB après restauration

📄 PDF

`save-pdf`

Sauvegarde un devis PDF sur le disque.

Paramètres :

```
{
  base64: string  // PDF encodé en base64
}
```

Retour :

```
{
  success: boolean,
 path?: string,
message?: string
}
```

Défaut : Dossier Documents de l'utilisateur

---

print-pdf

Ouvre le PDF dans l'application système par défaut pour impression.

Paramètres :

- base64 (string) : PDF encodé en base64

Retour :

```
{
  success: boolean,
 message?: string
}
```

Comportement :

. Crée un chier temporaire dans le dossier temp
. Ouvre le chier avec l'application par défaut
. Nettoie le chier après 30 secondes

---

save-quote

Sauvegarde les données d'un devis dans la DB.

Paramètres :

```
{
  id: string,
  clientName: string,
  date: string,
 materials: Material[],
  total: number,
  // ... autres données du devis
}
```

Retour :

```
{
  success: boolean,
 message?: string
}
```

Note : Les devis sont stockés dans `db.data.quotes[]`

Performance

~~get-performance-report~~

Récupère un rapport détaillé des performances de l'application.

Paramètres : Aucun

Retour :

```
{
  success: boolean,
  data: {
    metrics: PerformanceMetric[],
    cache: {
      size: number,
      items: string[]
    },
    memory: {
      rss: string,
      heapTotal: string,
      heapUsed: string,
      external: string
```

```
    }
  }
}
```

Type PerformanceMetric :

```
interface PerformanceMetric {
operation: string;
  count:   number;   avgTime:
  string;  //  en  ms  minTime:
  string;   maxTime:   string;
  totalTime: string;

}
```

Opérations trackées :

- `load-config load-`
- `header-config`
- `load-db`
- `backup-db`
- `clean-backups get-`
- `materials update-`
- `materials import-`
- `data`
- `set-header-config`
- `calculate-hash`
- `rebuild-index`
- `batch-write`

Alertes automatiques :

- Si une opération prend >1000ms, un warning est logué

Système de cache

Memor yCache

Classe de cache générique avec TTL (Time To Live).

```
class                MemoryCache{
set(key,value,ttlMs=300000)          // Défaut: 5 min
get(key)    has(key)    delete(key)   // Retourne null si expiré
clear()  getSize() }
```

Instance globale :

```
const dbCache = new MemoryCache();
```

Clés de cache :

Clé	Contenu	TTL 5	Invalidation
materials-db	Instance LowDB + index	min 5	update-materials, import-data
materials-list	Array de matériaux	min	update-materials, import-data
app-config	Con guration app	10 min	Jamais (redémarrage nécessaire)
header-config	Con guration en-tête	10 min	set-header-config

Performance :

- Cache hit : ~0.1ms
- Cache miss : 50-200ms (selon l'opération)

Monitoring des performances

PerformanceMonitor

Classe de monitoring des opérations.

```
class PerformanceMonitor{
start(operation)                  // Retourne startTime
end(operation,startTime)          // Calcule et enregistre la durée
getReport()                       // Retourne toutes les métriques
}
```

Utilisation :

```
const startTime = perfMonitor.start('my-operation');
// ... code à mesurer
perfMonitor.end('my-operation', startTime);
```

Alertes automatiques :
Si une opération prend > 1000ms :

```
[WARN] Opération lente détectée: my-operation (1523.45ms)
```

Gestion des erreurs

Structure de réponse

Toutes les réponses IPC suivent ce format :

```
// Succès
{
  success: true,
  data?: any
}

// Erreur
{
success:   false,
message: string }
```

Codes d'erreur courants

Erreur	Message	Cause
DB non initialisée	`Base de données non initialisée`	`loadDb()` n'a pas été appelé
Fichier corrompu	`Fichier JSON invalide`	JSON mal formé
Con guration SMTP	`Configuration SMTP non définie`	SMTP non con guré dans app-con g
Import invalide	`Format de données incorrect`	Structure JSON incorrecte

Erreur	Message	Cause
Mot de passe vide	`Le mot de passe ne peut pas être vide`	Tentative de set-password avec string vide

Logging

Niveaux de log :

- ERROR : Erreurs critiques (a chées en rouge sur console)
- WARN : Avertissements (opérations lentes, etc.)
- INFO : Informations générales

Format :

```
[2025-10-07T19:30:45.123Z] [INFO] Message de log
```

Fichier : `%userData%/logs.txt`

Performance :

- Logs asynchrones (batch ush toutes les 1 seconde)
- Flush immédiat au shutdown

Exemples d'utilisation

1. Récupérer et a cher les matériaux

```
// Frontend (app.js ou index.html)
async function loadMaterials() {
  const result = await window.api.getMaterials();

  if(result.success) {
    const materials = result.data;
    console.log(`${materials.length} matériaux chargés`);

    //  Afficher dans le DOM
    materials.forEach(mat => {
      console.log(`${mat.name}: ${mat.price}€`);
    });
  }else{
    console.error('Erreur:', result.message);
```

```
  }
}
```

---

2. Ajouter un nouveau matériau

```
async function addMaterial(newMaterial) {
  // 1. Récupérer la liste actuelle
  const result = await window.api.getMaterials();

  if(!result.success) {
    throw new Error(result.message);
  }

  const materials = result.data;

  //2.  Ajouter le nouveau matériau
  materials.push({
    id:Date.now().toString(),
    name:newMaterial.name,
    price:newMaterial.price,
    barcode:newMaterial.barcode || '',
    category:newMaterial.category || 'Autre'
  });

  // 3. Mettre à jour la DB
  const updateResult = await window.api.updateMaterials(materials);

 if (updateResult.success) {
    console.log('Matériau ajouté avec succès');
  } else {
    console.error('Erreur:', updateResult.message);
  }
}
```

---

3. Con gurer le SMTP

```
async function setupEmail(){
const smtpConfig = {
    service: 'gmail',
    auth: {
```

```
      user: 'votre.email@gmail.com',
      pass: 'votre_mot_de_passe_app'
    }
  };

  // 1. Tester la configuration
  const testResult = await window.api.testEmailConfig(smtpConfig);

  if(!testResult.success) {
    alert('Configuration SMTP invalide: ' + testResult.message);
    return;
  }

  // 2. Sauvegarder la configuration
  const saveResult = await window.api.setConfig({ smtp: smtpConfig });

 if (saveResult.success) {
    console.log('Configuration SMTP sauvegardée');
  }
}
```

---

4. Envoyer un devis par email

```
async function sendQuote(clientEmail, pdfBase64) {
const result = await window.api.sendQuoteEmail({
    to: clientEmail,
    subject: 'Votre devis AutoCalc OptiDevis',
    body: 'Bonjour,\n\nVeuillez trouver ci-joint votre devis.\n\nCordiale
    attachment: {
      filename: 'devis.pdf',
      content: pdfBase64
    }
 });

 if (result.success) {
    alert('Email envoyé avec succès !');
  } else {
    alert('Erreur d\'envoi: ' + result.message);
  }
}
```

---

5. Générer un rapport de performance

```
async function checkPerformance() {
const result = await window.api.getPerformanceReport();

  if(!result.success) {
    console.error('Erreur:', result.message);
    return;
  }

  const { metrics, cache, memory } = result.data;

  console.log('=== RAPPORT DE PERFORMANCE ===');
  console.log('\nMétriques:');
  metrics.forEach(m => {
    console.log(`${m.operation}:`);
    console.log(`  - Appelé ${m.count} fois`);
    console.log(`  - Temps moyen: ${m.avgTime}ms`);
    console.log(`  - Min/Max: ${m.minTime}ms / ${m.maxTime}ms`);
  });

  console.log('\nCache:');
  console.log(`  - ${cache.size} entrées`);
  console.log(`  - Clés: ${cache.items.join(', ')}`);

 console.log('\nMémoire:');
  console.log(`  - RSS: ${memory.rss}`);
  console.log(`  - Heap utilisé: ${memory.heapUsed} / ${memory.heapTotal}
}
```

---

6. Backup et restauration

```
// Backup automatique
// (Déjà géré automatiquement toutes les 10 minutes + au shutdown)

// Restaurer depuis un backup
async function restoreFromBackup() {
  const result = await window.api.restoreBackup();

  if(result.success) {
    alert('Base de données restaurée avec succès !');
```

```
      // Recharger les matériaux
      await loadMaterials();
    } else {
      alert('Erreur de restauration: ' + result.message);
    }
  }
```

Optimisations implémentées

1. Cache en mémoire

- **Gain** : 95-99% de réduction du temps de lecture
- get-materials : 100ms → 0.1ms (cache hit)

2. Batch writing

- **Gain** : 80% de réduction des écritures disque
- Regroupe les écritures sur 500ms

3. Backup optimisé

- **Compression gzip** : 70-80% d'économie d'espace
- **Hash check** : Backup uniquement si changements détectés
- **Réduction** : 10 backups/heure → 2-3 backups/heure

4. Indexation des matériaux

- **Gain** : Recherche O(n) → O(1)
- Recherche par ID/barcode/nom instantanée

5. I/O asynchrone

- **Gain** : 40-60% de réduction du temps de réponse
- Toutes les opérations fs sont non-bloquantes

6. Logging asynchrone

- **Gain** : 95% de réduction de l'overhead de logging
- Batch ush toutes les 1 seconde

7. Nettoyage des backups parallélisé

- **Gain** : 70% de réduction du temps d'exécution
- Utilise `Promise.all()` pour supprimer plusieurs chiers en parallèle

Métriques de performance attendues

Opération	Avant	Après	Amélioration
`load-db`	150ms	50ms	67%
`get-materials` (cache hit)	100ms	0.1ms	99.9%
`update-materials`	200ms	50ms	75%
`backup-db` (avec changements)	300ms	120ms	60%
`backup-db` (sans changements)	300ms	5ms	98%
`clean-backups` (10 chiers)	500ms	150ms	70%

Bonnes pratiques

1. Toujours véri er `success`

```
const result = await window.api.someHandler();
if (result.success) {
  // Utiliser result.data
} else {
  // Gérer result.message
}
```

2. Invalider le cache après modi cations

```
// Le cache est automatiquement invalidé par les handlers
// Pas besoin de le faire manuellement
```

3. Utiliser le rapport de performance

```
// En développement, vérifier régulièrement les performances
if (isDevelopment) {
  setInterval(async () => {
    const report = await window.api.getPerformanceReport();
    console.log(report);
  }, 60000); // Toutes les minutes
}
```

4. Débouncer les opérations utilisateur

```
// Exemple: recherche en temps réel
let searchTimeout;
searchInput.addEventListener('input', (e) =>{
clearTimeout(searchTimeout);
   searchTimeout = setTimeout(() => {
     performSearch(e.target.value);
  }, 300);
});
```
