

# Plan de Modifications

Adaptation de votre projet existant

Cache BDD + IMAP IDLE + WebSocket

|                   |                                     |
|-------------------|-------------------------------------|
| <b>Projet :</b>   | WebMail Laravel existant            |
| <b>Serveur :</b>  | BlueMind (mailpar.centrech.fr)      |
| <b>Approche :</b> | Modifier l'existant, ne rien casser |
| <b>Date :</b>     | 17 March 2026                       |

# 1. Audit de vos tables existantes

Voici l'analyse de chacune de vos tables par rapport à l'architecture cache + IMAP IDLE + WebSocket :

## Tables existantes — Rien à changer

| Statut | Table              | Champ / Index                                                               | Action                                     |
|--------|--------------------|-----------------------------------------------------------------------------|--------------------------------------------|
| OK     | imap_configs       | host, port, encryption, username, password, validate_cert, type, society_id | Configuration serveur IMAP/SMTP. Complète. |
| OK     | user_imap_configs  | username, fullname, password, user_id                                       | Identifiants par utilisateur. Complet.     |
| OK     | scheduled_webmails | data, scheduled_at, sent, is_cancelled, user_id, society_id                 | Emails programmés. Aucun changement.       |
| OK     | email_signatures   | name, content, enabled, user_id                                             | Signatures email. Aucun changement.        |
| OK     | webmail_addresses  | email, fullname, user_id                                                    | Carnet d'adresses. Aucun changement.       |
| OK     | mail_histories     | imap_mail_id, model (morph)                                                 | Historique. Aucun changement.              |

## Table à modifier — imap\_mails

| Statut | Table      | Champ / Index     | Action                                     |
|--------|------------|-------------------|--------------------------------------------|
| OK     | imap_mails | uid               | Existe. UID IMAP du message.               |
| OK     | imap_mails | subject           | Existe. Sujet du message.                  |
| OK     | imap_mails | body              | Existe. Corps du message (HTML).           |
| OK     | imap_mails | to, from, cc, bcc | Existe. Mais stocké en TEXT (pas optimal). |
| OK     | imap_mails | folder            | Existe. Nom du dossier IMAP.               |
| OK     | imap_mails | listen            | Existe. Flag custom.                       |
| OK     | imap_mails | attachments       | Existe (ajouté en juillet). TEXT.          |
| MANQUE | imap_mails | is_read           | Flag \Seen IMAP. Essentiel pour l'UI.      |
| MANQUE | imap_mails | is_flagged        | Flag \Flagged (email important/étoilé).    |
| MANQUE | imap_mails | is_answered       | Flag \Answered (email répondu).            |

| Statut | Table      | Champ / Index                 | Action                                           |
|--------|------------|-------------------------------|--------------------------------------------------|
| MANQUE | imap_mails | is_draft                      | Flag \Draft (brouillon).                         |
| MANQUE | imap_mails | sent_at                       | Date d'envoi. Essentiel pour le tri par date.    |
| MANQUE | imap_mails | message_id                    | Header Message-ID. Essentiel pour le threading.  |
| MANQUE | imap_mails | in_reply_to                   | Header In-Reply-To. Pour les threads.            |
| MANQUE | imap_mails | preview                       | Aperçu 150 caractères. Évite de charger le body. |
| MANQUE | imap_mails | thread_id                     | Identifiant de thread calculé.                   |
| MANQUE | imap_mails | size                          | Taille du message en octets.                     |
| MANQUE | imap_mails | body_text                     | Corps texte brut (pour recherche full-text).     |
| MANQUE | imap_mails | synced_at                     | Date de dernière synchronisation.                |
| MANQUE | imap_mails | user_imap_config_id           | FK vers user_imap_configs (lier au compte).      |
| MANQUE | imap_mails | INDEX (folder, sent_at)       | Index pour lister les mails triés.               |
| MANQUE | imap_mails | INDEX (uid, folder)           | Index pour sync incrémentale rapide.             |
| MANQUE | imap_mails | FULLTEXT (subject, body_text) | Index pour recherche rapide.                     |

### Tables à créer — N'existent pas encore

| Statut | Table            | Champ / Index                                                                     | Action                                               |
|--------|------------------|-----------------------------------------------------------------------------------|------------------------------------------------------|
| MANQUE | mail_mailboxes   | account_id, name, full_name, uid_validity, last_uid, unread_count, total_messages | Dossiers IMAP avec métadonnées de sync.              |
| MANQUE | mail_recipients  | imap_mail_id, type (to/cc/bcc), email, name                                       | Destinataires extraits pour recherche indexée.       |
| MANQUE | mail_attachments | imap_mail_id, filename, mime_type, size, content_id, part_number                  | Métadonnées des pièces jointes (fetch à la demande). |

Table à modifier — imap\_configs

| Statut | Table        | Champ / Index  | Action                                        |
|--------|--------------|----------------|-----------------------------------------------|
| MANQUE | imap_configs | last_synced_at | Timestamp de dernière sync globale.           |
| MANQUE | imap_configs | is_active      | Pour désactiver un serveur sans le supprimer. |

## 2. Migrations à créer

Voici les 5 fichiers de migration à ajouter dans votre projet. Ils utilisent vos conventions existantes (UUID, foreignUuid, etc).

### Migration 1 : Modifier imap\_mails

```
php artisan make:migration add_cache_fields_to_imap_mails_table
```

```
Schema::table('imap_mails', function (Blueprint $table) { // Flags IMAP
    $table->boolean('is_read')->default(false)->after('listen');
    $table->boolean('is_flagged')->default(false)->after('is_read');
    $table->boolean('is_answered')->default(false)->after('is_flagged');
    $table->boolean('is_draft')->default(false)->after('is_answered'); // Metadonnees
    essentielles $table->timestamp('sent_at')->nullable()->after('is_draft');
    $table->string('message_id', 500)->nullable()->after('sent_at');
    $table->string('in_reply_to', 500)->nullable()->after('message_id');
    $table->string('preview', 500)->nullable()->after('in_reply_to');
    $table->string('thread_id', 255)->nullable()->after('preview');
    $table->unsignedInteger('size')->default(0)->after('thread_id');
    $table->longText('body_text')->nullable()->after('body');
    $table->timestamp('synced_at')->nullable()->after('size'); // FK vers le compte
    utilisateur $table->foreignUuid('user_imap_config_id')->nullable()->after('synced_at')
    ->references('id')->on('user_imap_configs'); // INDEX pour la performance
    $table->index(['folder', 'sent_at'], 'idx_folder_sent'); $table->index(['uid',
    'folder'], 'idx_uid_folder'); $table->index('message_id', 'idx_message_id');
    $table->index('thread_id', 'idx_thread_id'); $table->index('sent_at', 'idx_sent_at');
});
```

**Attention :** Lancez d'abord cette migration sur une copie de la BDD pour vérifier qu'il n'y a pas de conflit avec vos données existantes.

## Migration 2 : Créer mail\_mailboxes

```
php artisan make:migration create_mail_mailboxes_table
```

```
Schema::create('mail_mailboxes', function (Blueprint $table) {
    $table->uuid('id')->primary(); $table->foreignUuid('user_imap_config_id')
    ->references('id')->on('user_imap_configs') ->cascadeOnDelete(); $table->string('name');
    // INBOX, Sent, Drafts... $table->string('full_name', 500); // Chemin IMAP complet
    $table->string('delimiter', 5)->default('.');
    $table->integer('total_messages')->default(0);
    $table->integer('unread_count')->default(0);
    $table->unsignedInteger('uid_validity')->nullable();
    $table->unsignedInteger('last_uid')->default(0);
    $table->boolean('selectable')->default(true); $table->timestamps();
    $table->unique(['user_imap_config_id', 'full_name']); });
```

**Note :** `uid_validity` est une valeur que le serveur IMAP assigne à chaque dossier. Si elle change, tous les UUIDs de ce dossier sont invalidés et il faut tout re-synchroniser. `last_uid` est le dernier UID synchronisé : la sync incrémentale ne télécharge que les UUIDs supérieurs.

## Migration 3 : Créer mail\_recipients

```
php artisan make:migration create_mail_recipients_table
```

```
Schema::create('mail_recipients', function (Blueprint $table) {
    $table->uuid('id')->primary(); $table->foreignUuid('imap_mail_id')
    ->references('id')->on('imap_mails') ->cascadeOnDelete(); $table->enum('type', ['to',
    'cc', 'bcc']); $table->string('email'); $table->string('name')->nullable();
    $table->index(['imap_mail_id', 'type']); $table->index('email'); });
```

**Note :** Cette table remplace les champs TEXT `to`, `cc`, `bcc` de `imap_mails`. Gardez les anciens champs pour compatibilité, mais utilisez cette table pour les nouvelles requêtes.

## Migration 4 : Créer mail\_attachments

```
php artisan make:migration create_mail_attachments_table
```

```
Schema::create('mail_attachments', function (Blueprint $table) {
    $table->uuid('id')->primary(); $table->foreignUuid('imap_mail_id')
    ->references('id')->on('imap_mails') ->cascadeOnDelete(); $table->string('filename',
    500); $table->string('mime_type')->nullable();
    $table->unsignedInteger('size')->default(0); $table->string('content_id')->nullable();
    // CID images inline $table->string('disposition', 50)->default('attachment');
    $table->string('part_number', 50)->nullable(); // Pour fetch IMAP $table->timestamps();
    $table->index('imap_mail_id'); });
```

**Note :** Le champ `part_number` stocke la référence IMAP de la pièce jointe. Le contenu binaire n'est PAS stocké en BDD : il est fetché depuis IMAP à la demande quand l'utilisateur clique sur 'Télécharger'.

## Migration 5 : Modifier imap\_configs

```
php artisan make:migration add_sync_fields_to_imap_configs_table
```

```
Schema::table('imap_configs', function (Blueprint $table) {
    $table->boolean('is_active')->default(true)->after('validate_cert');
    $table->timestamp('last_synced_at')->nullable()->after('is_active'); });
```

### 3. Ordre d'exécution des migrations

Les migrations doivent être exécutées dans cet ordre précis pour respecter les dépendances de clés étrangères :

| Ordre | Fichier                               | Dépendance                      |
|-------|---------------------------------------|---------------------------------|
| 1     | add_sync_fields_to_imap_configs_table | Aucune                          |
| 2     | create_mail_mailboxes_table           | user_imap_configs (existe déjà) |
| 3     | add_cache_fields_to_imap_mails_table  | user_imap_configs (existe déjà) |
| 4     | create_mail_recipients_table          | imap_mails (existe déjà)        |
| 5     | create_mail_attachments_table         | imap_mails (existe déjà)        |

```
php artisan migrate
```

**Attention :** Faites un backup de votre base de données avant de lancer les migrations. La migration 3 (modify imap\_mails) ajoute des colonnes à une table existante avec des données.

## 4. Migration des données existantes (backfill)

Après les migrations de structure, il faut remplir les nouveaux champs pour les emails déjà présents dans `imap_mails`. Créez une commande Artisan pour cela :

```
php artisan make:command BackfillImapMails
```

### 4.1 Extraire les destinataires

Vos champs `to`, `cc`, `bcc` sont stockés en TEXT. Le backfill parse ces champs et insère dans `mail_recipients` :

```
ImapMail::chunk(200, function ($mails) { foreach ($mails as $mail) { foreach
(['to','cc','bcc'] as $type) { $raw = $mail->$type; if (!$raw) continue; $addresses =
$this->parseAddresses($raw); foreach ($addresses as $addr) { MailRecipient::create([
'imap_mail_id' => $mail->id, 'type' => $type, 'email' => $addr['email'], 'name' =>
$addr['name'] ?? null, ]); } } } });
```

### 4.2 Générer les previews

```
ImapMail::whereNull('preview')->chunk(200, function ($mails) { foreach ($mails as $mail)
{ $text = strip_tags($mail->body); $mail->update([ 'preview' => Str::limit($text, 150),
'body_text' => $text, ]); } });
```

### 4.3 Créer les mailboxes

```
// Extraire les dossiers uniques de imap_mails $folders = ImapMail::select('folder')
->distinct()->pluck('folder'); foreach ($folders as $folderName) {
MailMailbox::firstOrCreate( ['user_imap_config_id' => $configId, 'full_name' =>
$folderName], ['name' => $folderName, 'last_uid' => 0] ); }
```

## 5. Fichiers PHP à créer

Voici la liste complète des fichiers à ajouter à votre projet :

| Type      | Fichier                                    | Rôle                             |
|-----------|--------------------------------------------|----------------------------------|
| Migration | add_cache_fields_to_imap_mails_table       | Ajouter les champs manquants     |
| Migration | create_mail_mailboxes_table                | Table des dossiers IMAP          |
| Migration | create_mail_recipients_table               | Table des destinataires          |
| Migration | create_mail_attachments_table              | Table des pièces jointes         |
| Migration | add_sync_fields_to_imap_configs_table      | Champs sync sur imap_configs     |
| Model     | app/Models/MailMailbox.php                 | Modèle Eloquent mailboxes        |
| Model     | app/Models/MailRecipient.php               | Modèle Eloquent recipients       |
| Model     | app/Models/MailAttachment.php              | Modèle Eloquent attachments      |
| Job       | app/Jobs/SyncMailboxJob.php                | Sync incrémentale IMAP → BDD     |
| Job       | app/Jobs/SyncFlagsJob.php                  | Sync des flags (lu, important)   |
| Command   | app/Console/Commands/ImapIdleCommand.php   | Worker IMAP IDLE temps réel      |
| Command   | app/Console/Commands/BackfillImapMails.php | Migration des données existantes |
| Event     | app/Events/NewEmailReceived.php            | Event broadcast WebSocket        |
| Config    | config/supervisor/mail-idle.conf           | Supervisor pour IDLE worker      |

## Fichiers existants à modifier

| Fichier                       | Modification                                                                                                           |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------|
| app/Models/ImapMail.php       | Ajouter les relations : recipients(), attachments(), mailbox(). Ajouter les \$casts pour is_read, is_flagged, sent_at. |
| app/Models/UserImapConfig.php | Ajouter la relation : mailboxes()                                                                                      |
| app/Console/Kernel.php        | Ajouter les jobs au scheduler (everyTwoMinutes, everyFiveMinutes)                                                      |
| routes/channels.php           | Ajouter le canal privé : Broadcast::channel('mail.{userId}', ...)                                                      |
| .env                          | Ajouter BROADCAST_CONNECTION=ververb + config Redis                                                                    |

| Fichier                 | Modification             |
|-------------------------|--------------------------|
| config/broadcasting.php | Configurer Reverb/Pusher |

## 6. Packages à installer

| Package              | Commande                              | Rôle                                                     |
|----------------------|---------------------------------------|----------------------------------------------------------|
| webklex/laravel-imap | composer require webklex/laravel-imap | Client IMAP optimisé pour Laravel (si pas déjà installé) |
| laravel/reverb       | composer require laravel/reverb       | Serveur WebSocket officiel Laravel                       |
| laravel-echo         | npm install laravel-echo pusher-js    | Client WebSocket côté navigateur                         |
| supervisor           | apt install supervisor                | Garder le worker IDLE actif                              |
| redis                | apt install redis-server              | Queue pour les jobs de sync                              |

### Vérifier ce qui est déjà installé

```
# Verifier si webklex/laravel-imap est deja installe composer show | grep imap # Verifier si Redis est disponible redis-cli ping # Doit repondre PONG # Verifier si Supervisor est installe supervisorctl status
```

## 7. Impact sur le code existant

Le principe est de ne rien casser. Voici comment les modifications s'intègrent :

### 7.1 Ce qui ne change PAS

**OK** : Les anciens champs to, cc, bcc (TEXT) restent dans imap\_mails. Votre code existant continue de fonctionner.

**OK** : Le champ attachments (TEXT) reste. Le code qui l'utilise ne casse pas.

**OK** : Les tables scheduled\_webmails, email\_signatures, webmail\_addresses, mail\_histories ne sont pas touchées.

**OK** : Vos routes, controllers et vues existants continuent de fonctionner.

### 7.2 Ce qui s'améliore

Une fois les modifications déployées, vous pouvez progressivement adapter votre code :

| Avant (lent)                                    | Après (rapide)                  | Gain             |
|-------------------------------------------------|---------------------------------|------------------|
| Chaque page fait une connexion IMAP             | Lecture depuis MySQL local      | 50x plus rapide  |
| Recherche via IMAP SEARCH                       | Recherche FULLTEXT locale       | 100x plus rapide |
| Rafraîchir la page pour voir les nouveaux mails | Push WebSocket automatique      | Temps réel       |
| Pas de flags (lu/non lu)                        | is_read, is_flagged dans la BDD | UI moderne       |
| Pas de threading                                | Groupement par thread_id        | Gmail-like       |

### 7.3 Ordre de migration recommandé

| Semaine | Action                                           | Risque                                       |
|---------|--------------------------------------------------|----------------------------------------------|
| 1       | Migrations BDD + backfill données existantes     | Faible (ajout de colonnes seulement)         |
| 2       | Job SyncMailboxJob (cron toutes les 2 min)       | Nul (nouveau code, ne touche pas l'existant) |
| 3       | Adapter l'UI pour lire depuis BDD au lieu d'IMAP | Moyen (modifier les controllers)             |

| Semaine | Action                                  | Risque                           |
|---------|-----------------------------------------|----------------------------------|
| 4       | Worker IMAP IDLE + Supervisor           | Faible (nouveau process séparé)  |
| 5       | WebSocket (Reverb) + JS côté client     | Faible (ajout de fonctionnalité) |
| 6       | Tests, optimisation, mise en production | Standard                         |

## 8. Checklist de déploiement

Cochez chaque étape au fur et à mesure :

### Préparation

- Backup complet de la base de données
- Vérifier que Redis est installé et actif
- Vérifier que Supervisor est installé
- Installer webklex/laravel-imap (si pas déjà fait)
- Installer laravel/reverb

### Migrations

- Exécuter les 5 migrations dans l'ordre
- Vérifier que les tables sont créées correctement
- Lancer la commande de backfill (BackfillImapMails)
- Vérifier que les données existantes sont intactes

### Synchronisation

- Créer le Job SyncMailboxJob
- Configurer le scheduler dans Kernel.php
- Tester : `php artisan schedule:run`
- Vérifier que les nouveaux mails arrivent dans la BDD

### IMAP IDLE

- Créer la commande `ImapIdleCommand`
- Tester : `php artisan mail:idle-listen`
- Configurer Supervisor pour garder le worker actif
- Envoyer un email test et vérifier qu'il apparaît en ~1s

### WebSocket

- Configurer Reverb dans `.env` et `broadcasting.php`
- Créer l'événement `NewEmailReceived`
- Ajouter le canal privé dans `routes/channels.php`

- Intégrer Laravel Echo côté JavaScript
- Tester la notification en temps réel dans le navigateur

## Production

- Configurer le cron : `* * * * * php artisan schedule:run`
- Vérifier Supervisor : `supervisorctl status`
- Démarrer Reverb : `php artisan reverb:start`
- Monitoring des logs : `/var/log/supervisor/mail-idle.log`

---

**Ce plan de modifications est conçu pour s'intégrer à votre projet existant sans rien casser.** Chaque étape est indépendante : si vous ne faites que les migrations + le job de sync (semaines 1-3), vous aurez déjà une amélioration massive de performance. IMAP IDLE et WebSocket sont des bonus pour le temps réel.