










-  Documentation API Ecologis
 -  Vue d'ensemble
 -  Authentification
 - Système de rôles
 - Mécanisme JWT
 -  Routes disponibles
 - 1.  Authentification (/auth)
 - POST /auth/register
 - POST /auth/login
 - POST /auth/refresh
 - POST /auth/logout
 - POST /auth/reset-password
 - POST /auth/change-password
 - 2.  Gestion des maisons (/maisons)
 - POST /maisons
 - GET /maisons
 - GET /maisons/:id
 - PUT /maisons/:id
 - DELETE /maisons/:id
 - PATCH /maisons/:id/tarif
 - POST /maisons/residents/ajouter
 - POST /maisons/residents/retirer
 - 3.  Gestion des résidents (/residents)
 - POST /residents
 - GET /residents
 - GET /residents/:id
 - PUT /residents/:id
 - DELETE /residents/:id
 - 4.  Gestion des consommations (/consommations)
 - POST /consommations
 - GET /consommations/resident/:residentId
 - GET /consommations/maison/:maisonId
 - PUT /consommations/:id
 - DELETE /consommations/:id
 - 5.  Gestion des factures (/factures)
 - POST /factures/generer/:residentId
 - GET /factures/resident/:residentId

- GET /factures/maison/:maisonId
- GET /factures/:id
- PUT /factures/:id/payer
- 6. 🏠 Gestion des abonnements (/abonnements)
 - GET /abonnements
 - POST /abonnements/souscrire
 - POST /abonnements/renouveler
 - GET /abonnements/actuel
 - POST /abonnements/annuler
 - GET /abonnements/historique
- 💰 Logique métier
 - Calcul des factures
 - Gestion des abonnements
 - Notifications automatiques
- 🛠 Variables importantes
 - Tarifs par défaut
 - Limites d'abonnement
- 📱 Exemples de requêtes Postman
 - 1. Créer un compte propriétaire
 - 2. Se connecter
 - 3. Créer une maison (avec token)
 - 4. Ajouter un résident
 - 5. Enregistrer une consommation
 - 6. Générer une facture
- 🚀 Guide d'utilisation pour Flutter
 - 1. Authentification
 - 2. Gestion des requêtes
 - 3. Gestion des erreurs
 - 4. Exemple d'utilisation
- 📋 Résumé pour le développeur Flutter
 - Points clés à retenir :
 - Workflow recommandé :
 - Sécurité :
- 📞 Support





Vue d'ensemble

L'API Ecoglis est une solution de gestion de consommation électrique qui permet aux propriétaires de gérer leurs maisons, résidents et factures d'électricité. Elle gère automatiquement le calcul des factures basé sur les relevés de consommation et les tarifs personnalisés.

Base URL: <http://localhost:3000> (développement) / <https://votre-domaine.com> (production)



Authentification

Système de rôles

- **proprietaire** : Peut gérer maisons, résidents, abonnements et voir toutes les données
- **resident** : Peut voir ses propres consommations et factures

Mécanisme JWT

- **Access Token** : Valide 15 minutes, utilisé pour toutes les requêtes authentifiées
 - **Refresh Token** : Valide 7 jours, utilisé pour renouveler l'access token
 - **Format** : `Authorization: Bearer <access_token>`
-



Routes disponibles

1. Authentification (</auth>)

POST </auth/register>

Créer un compte propriétaire

Body JSON:

```
{
  "nom": "Doe",
  "prenom": "John",
  "email": "john.doe@example.com",
  "telephone": "+22890123456",
  "motDePasse": "MotDePasse123!"
}
```

Réponse succès (201):

```
{
  "message": "Compte propriétaire créé avec succès",
  "user": {
    "_id": "64f1a2b3c4d5e6f7g8h9i0j1",
    "nom": "Doe",
    "prenom": "John",
    "email": "john.doe@example.com",
    "telephone": "+22890123456",
    "role": "proprietaire",
    "firstLogin": false
  },
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

POST /auth/login

Connexion utilisateur

Body JSON:

```
{
  "email": "john.doe@example.com",
  "motDePasse": "MotDePasse123!"
}
```

Réponse succès (200):

```
{
  "message": "Connexion réussie",
  "user": {
    "_id": "64f1a2b3c4d5e6f7g8h9i0j1",
    "nom": "Doe",
    "prenom": "John",
    "email": "john.doe@example.com",

```

```
    "role": "proprietaire",
    "abonnementId": "64f1a2b3c4d5e6f7g8h9i0j2"
  },
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "abonnement": {
    "type": "premium",
    "prix": 29.99,
    "nbResidentsMax": 10
  }
}
```

POST /auth/refresh

Renouveler l'access token

Body JSON:

```
{
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

POST /auth/logout

Déconnexion (*Authentication requise*)

Headers: Authorization: Bearer <access_token>

POST /auth/reset-password

Changer le mot de passe (premier login) (*Authentication requise*)

Body JSON:

```
{
  "nouveauMotDePasse": "NouveauMotDePasse123!"
}
```

POST /auth/change-password

Changer le mot de passe normal (*Authentication requise*)

Body JSON:

```
{
  "motDePasseActuel": "AncienMotDePasse123!",
  "nouveauMotDePasse": "NouveauMotDePasse123!"
}
```

2. 🏠 Gestion des maisons (/maisons)

POST /maisons

Créer une maison (*Authentication + Propriétaire requis*)

Body JSON:

```
{
  "nomMaison": "Villa Sunshine",
  "adresse": {
    "rue": "123 Avenue de la Paix",
    "ville": "Lomé",
    "codePostal": "01BP1234",
    "pays": "Togo"
  },
  "description": "Belle villa avec jardin",
  "tarifKwh": 0.1740
}
```

Réponse succès (201):

```
{
  "message": "Maison créée avec succès",
  "maison": {
    "_id": "64f1a2b3c4d5e6f7g8h9i0j3",
    "nomMaison": "Villa Sunshine",
    "proprietaireId": "64f1a2b3c4d5e6f7g8h9i0j1",
    "adresse": {
      "rue": "123 Avenue de la Paix",
      "ville": "Lomé",
      "codePostal": "01BP1234",
      "pays": "Togo"
    },
    "tarifKwh": 0.1740,
    "statut": "active"
  }
}
```

GET /maisons

Lister les maisons (*Authentication requise*)

Réponse succès (200):

```
{
  "maisons": [
    {
      "_id": "64f1a2b3c4d5e6f7g8h9i0j3",
      "nomMaison": "Villa Sunshine",
      "adresse": {
        "rue": "123 Avenue de la Paix",
        "ville": "Lomé"
      },
      "tarifKwh": 0.1740,
      "nbResidents": 3,
      "statut": "active"
    }
  ]
}
```

GET /maisons/:id

Obtenir une maison spécifique (*Authentication requise*)

PUT /maisons/:id

Mettre à jour une maison (*Authentication + Propriétaire requis*)

DELETE /maisons/:id

Supprimer une maison (*Authentication + Propriétaire requis*)

PATCH /maisons/:id/tarif

Mettre à jour le tarif kWh (*Authentication + Propriétaire requis*)

Body JSON:

```
{
  "tarifKwh": 0.1850
}
```

POST /maisons/residents/ajouter

Ajouter un résident à une maison (*Authentication + Propriétaire requis*)

Body JSON:

```
{
  "maisonId": "64f1a2b3c4d5e6f7g8h9i0j3",
  "residentId": "64f1a2b3c4d5e6f7g8h9i0j4"
}
```

POST /maisons/residents/retirer

Retirer un résident d'une maison (*Authentication + Propriétaire requis*)

3. Gestion des résidents (/residents)

POST /residents

Ajouter un résident (*Authentication + Propriétaire + Vérification quota requis*)

Body JSON:

```
{
  "nom": "Smith",
  "prenom": "Alice",
  "email": "alice.smith@example.com",
  "telephone": "+22890123457"
}
```

Réponse succès (201):

```
{
  "message": "Résident ajouté avec succès",
  "resident": {
    "_id": "64f1a2b3c4d5e6f7g8h9i0j4",
    "nom": "Smith",
    "prenom": "Alice",
    "email": "alice.smith@example.com",
    "telephone": "+22890123457",
    "role": "resident",
    "idProprietaire": "64f1a2b3c4d5e6f7g8h9i0j1",
  }
}
```



```
"motDePasse": "MotDePasseTemporaire123!"
}
```

GET /residents

Lister les résidents (*Authentication + Propriétaire requis*)

GET /residents/:id

Obtenir un résident spécifique (*Authentication + Propriétaire requis*)

PUT /residents/:id

Mettre à jour un résident (*Authentication + Propriétaire requis*)

DELETE /residents/:id

Supprimer un résident (*Authentication + Propriétaire requis*)

4. ⚡ Gestion des consommations (/consommations)

POST /consommations

Enregistrer une consommation (*Authentication requise*)

Body JSON:

```
{
  "residentId": "64f1a2b3c4d5e6f7g8h9i0j4",
  "maisonId": "64f1a2b3c4d5e6f7g8h9i0j3",
  "kwh": 125.5,
  "mois": 12,
  "annee": 2025,
  "commentaire": "Relevé du compteur principal"
}
```

Réponse succès (201):

```
{
  "message": "Consommation enregistrée avec succès",
  "consommation": {
    "_id": "64f1a2b3c4d5e6f7g8h9i0j5",
    "residentId": "64f1a2b3c4d5e6f7g8h9i0j4",
    "maisonId": "64f1a2b3c4d5e6f7g8h9i0j3",
    "kwh": 125.5,
    "mois": 12,
    "annee": 2025,
    "montant": 21.84,
    "statut": "enregistree"
  }
}
```

GET /consommations/resident/:residentId

Historique des consommations d'un résident (*Authentication requise*)

Query params:

- **annee** (optionnel) : Année spécifique
- **mois** (optionnel) : Mois spécifique

Réponse succès (200):

```
{
  "consommations": [
    {
      "_id": "64f1a2b3c4d5e6f7g8h9i0j5",
      "kwh": 125.5,
      "mois": 12,
      "annee": 2025,
      "montant": 21.84,
      "maisonId": {
        "_id": "64f1a2b3c4d5e6f7g8h9i0j3",
        "nomMaison": "Villa Sunshine"
      }
    }
  ],
  "statistiques": {
    "totalKwh": 125.5,
    "totalMontant": 21.84,
    "moyenneKwh": 125.5,
    "nombreRelevés": 1
  }
}
```

GET /consommations/maison/:maisonId

Consommations d'une maison (*Authentication requise*)

PUT `/consommations/:id`

Mettre à jour une consommation (*Authentication requise*)

DELETE `/consommations/:id`

Supprimer une consommation (*Authentication + Propriétaire requis*)

5. Gestion des factures (`/factures`)

POST `/factures/generer/:residentId`

Générer une facture pour un résident (*Authentication requise*)

Body JSON:

```
{
  "mois": 12,
  "annee": 2025,
  "fraisFixes": 2.50
}
```

Réponse succès (201):

```
{
  "message": "Facture générée avec succès",
  "facture": {
    "_id": "64f1a2b3c4d5e6f7g8h9i0j6",
    "numeroFacture": "FACT-202512-0001",
    "residentId": "64f1a2b3c4d5e6f7g8h9i0j4",
    "montant": 24.34,
    "statut": "non payée",
    "dateEmission": "2025-12-01T10:00:00.000Z",
    "dateEcheance": "2025-12-31T10:00:00.000Z",
    "details": {
      "kwh": 125.5,
      "prixKwh": 0.1740,
      "fraisFixes": 2.50
    }
  }
}
```

GET /factures/resident/:residentId

Factures d'un résident (*Authentication requise*)

Query params:

- **statut** (optionnel) : payée, non payée, en retard
- **annee** (optionnel) : Année spécifique

GET /factures/maison/:maisonId

Factures d'une maison (*Authentication requise*)

GET /factures/:id

Obtenir une facture spécifique (*Authentication requise*)

PUT /factures/:id/payer

Marquer une facture comme payée (*Authentication requise*)

6. Gestion des abonnements (/abonnements)

GET /abonnements

Liste des offres disponibles (*Public*)

Réponse succès (200):

```
{
  "offres": [
    {
      "type": "basic",
      "prix": 9.99,
      "nbResidentsMax": 3,
      "description": "Pour petits propriétaires"
    },
    {
      "type": "premium",
      "prix": 29.99,
      "nbResidentsMax": 10,
    }
  ]
}
```

```
    "description": "Pour propriétaires moyens"
  },
  {
    "type": "enterprise",
    "prix": 99.99,
    "nbResidentsMax": 50,
    "description": "Pour gros propriétaires"
  }
]
```

POST /abonnements/souscrire

Souscrire à un abonnement (*Authentication + Propriétaire requis*)

Body JSON:

```
{
  "type": "premium",
  "dureeMois": 12
}
```

POST /abonnements/renouveler

Renouveler un abonnement (*Authentication + Propriétaire requis*)

GET /abonnements/actuel

Obtenir l'abonnement actuel (*Authentication + Propriétaire requis*)

POST /abonnements/annuler

Annuler un abonnement (*Authentication + Propriétaire requis*)

GET /abonnements/historique

Historique des abonnements (*Authentication + Propriétaire requis*)



Logique métier

Calcul des factures

1. **Tarif personnalisé** : Chaque maison a son propre tarif kWh (défini par le propriétaire)
2. **Tarif par défaut** : Si aucun tarif n'est défini, utilisation du tarif standard (0.1740 FCFA/kWh)
3. **Frais fixes** : Possibilité d'ajouter des frais fixes (maintenance, etc.)
4. **Formule** : $\text{Montant} = (\text{kWh} \times \text{Tarif kWh}) + \text{Frais fixes}$

Gestion des abonnements

- **Quota résidents** : Limite le nombre de résidents selon le type d'abonnement
- **Expiration automatique** : Notifications 7 jours avant expiration
- **Renouvellement** : Extension automatique de la durée

Notifications automatiques

- **Consommation élevée** : Si la consommation dépasse la moyenne des 3 derniers mois
- **Nouvelle facture** : Envoi WhatsApp automatique lors de la génération
- **Factures en retard** : Rappels automatiques après 30 jours
- **Expiration abonnement** : Alertes 7 jours avant expiration



Variables importantes

Tarifs par défaut

- **Prix kWh standard** : 0.1740 FCFA/kWh
- **Frais fixes** : 0 FCFA (configurable par propriétaire)
- **Échéance facture** : 30 jours après émission

Limites d'abonnement

- **Basic** : 3 résidents max
 - **Premium** : 10 résidents max
 - **Enterprise** : 50 résidents max
-



Exemples de requêtes Postman

1. Créer un compte propriétaire

```
POST http://localhost:3000/auth/register
Content-Type: application/json
```

```
{
  "nom": "Doe",
  "prenom": "John",
  "email": "john.doe@example.com",
  "telephone": "+22890123456",
  "motDePasse": "MotDePasse123!"
}
```

2. Se connecter

```
POST http://localhost:3000/auth/login
Content-Type: application/json
```

```
{
  "email": "john.doe@example.com",
  "motDePasse": "MotDePasse123!"
}
```

3. Créer une maison (avec token)

```
POST http://localhost:3000/maisons
Authorization: Bearer <access_token>
Content-Type: application/json
```

```
{
  "nomMaison": "Villa Sunshine",
  "adresse": {
```

```
"rue": "123 Avenue de la Paix",
"ville": "Lomé",
"codePostal": "01BP1234"
},
"tarifKwh": 0.1740
}
```

4. Ajouter un résident

```
POST http://localhost:3000/residents
Authorization: Bearer <access_token>
Content-Type: application/json

{
  "nom": "Smith",
  "prenom": "Alice",
  "email": "alice.smith@example.com",
  "telephone": "+22890123457"
}
```

5. Enregistrer une consommation

```
POST http://localhost:3000/consommations
Authorization: Bearer <access_token>
Content-Type: application/json

{
  "residentId": "64f1a2b3c4d5e6f7g8h9i0j4",
  "maisonId": "64f1a2b3c4d5e6f7g8h9i0j3",
  "kwh": 125.5,
  "mois": 12,
  "annee": 2025
}
```

6. Générer une facture

```
POST http://localhost:3000/factures/generer/64f1a2b3c4d5e6f7g8h9i0j4
Authorization: Bearer <access_token>
Content-Type: application/json

{
  "mois": 12,
```



```
"annee": 2025,  
"fraisFixes": 2.50  
}
```



Guide d'utilisation pour Flutter

1. Authentification

```
// Stocker les tokens de manière sécurisée  
class AuthService {  
  static String? accessToken;  
  static String? refreshToken;  
  
  static Future<bool> login(String email, String password) async {  
    // Appel POST /auth/login  
    // Stocker les tokens  
    // Retourner true si succès  
  }  
  
  static Future<String?> getValidToken() async {  
    // Vérifier si access token est valide  
    // Si expiré, utiliser refresh token  
    // Retourner token valide  
  }  
}
```

2. Gestion des requêtes

```
class ApiService {  
  static const String baseUrl = 'http://localhost:3000';  
  
  static Future<Map<String, String>> getHeaders() async {  
    final token = await AuthService.getValidToken();  
    return {  
      'Content-Type': 'application/json',  
      'Authorization': 'Bearer $token',  
    };  
  }  
  
  static Future<http.Response> get(String endpoint) async {  
    final headers = await getHeaders();  
    return await http.get(  
      Uri.parse('$baseUrl$endpoint'),  
      headers: headers,  
    );  
  }  
}
```

```
        headers: headers,
    );
}
}
```

3. Gestion des erreurs

```
class ApiException implements Exception {
    final String message;
    final int statusCode;

    ApiException(this.message, this.statusCode);

    static ApiException fromResponse(http.Response response) {
        final body = json.decode(response.body);
        return ApiException(body['message'] ?? 'Erreur inconnue', response.statusCode);
    }
}
```

4. Exemple d'utilisation

```
class ConsommationService {
    static Future<List<Consommation>> getConsommationsResident(String residentId)
    async {
        try {
            final response = await ApiService.get('/consommations/resident/$residentId');

            if (response.statusCode == 200) {
                final data = json.decode(response.body);
                return (data['consommations'] as List)
                    .map((json) => Consommation.fromJson(json))
                    .toList();
            } else {
                throw ApiException.fromResponse(response);
            }
        } catch (e) {
            // Gérer l'erreur
            rethrow;
        }
    }
}
```



Résumé pour le développeur Flutter

Points clés à retenir :

1. **Authentification obligatoire** : Toutes les routes (sauf `/auth/register`, `/auth/login`, `/auth/refresh`) nécessitent un token JWT valide
2. **Gestion des rôles** : Vérifier le rôle utilisateur avant d'afficher certaines fonctionnalités
3. **Refresh automatique** : Implémenter la logique de refresh token pour maintenir la session
4. **Gestion des erreurs** : Traiter les codes de statut HTTP et les messages d'erreur
5. **Notifications push** : L'API envoie automatiquement des notifications FCM pour les événements importants
6. **Calculs automatiques** : Les montants des factures sont calculés automatiquement par l'API

Workflow recommandé :

1. Authentification → Récupération des tokens
2. Vérification du rôle utilisateur
3. Chargement des données selon les permissions
4. Gestion des erreurs et des tokens expirés
5. Mise à jour en temps réel via WebSocket (optionnel)

Sécurité :

- Stocker les tokens de manière sécurisée (SecureStorage)
- Ne jamais exposer les tokens dans les logs
- Vérifier les permissions côté client ET côté serveur
- Implémenter la déconnexion automatique en cas d'erreur 401



Support

Pour toute question technique ou problème d'intégration, contactez l'équipe de développement backend.

Version API : 1.0.0

Dernière mise à jour : Décembre 2025