# 1. Résumé très synthétique pour Wagz

**Principe**

- L'abonnement K Bour Crypto devient un **NFT d'adhésion**.
- Posséder le NFT = être membre; les **droits** (membre, pro, modérateur…) sont dérivés des NFTs détenus.[^1][^2]
- L'accès reste **wallet-based** (connexion + signature), pas de rupture avec l'existant web/app.[^3][^1]

**Parcours utilisateur**

- CTA actuel "Rejoindre / S'abonner" ⇒ connexion wallet.
- Si le wallet possède déjà le NFT d'adhésion ⇒ on crée/charge le compte existant et on l'authentifie.
- Sinon ⇒ redirection vers une page "Acheter le NFT d'abonnement" (mint ou lien marketplace), puis re-check et accès une fois le NFT obtenu.[^4][^1]
- En cas d'interruption d'abonnement :
  - soit le NFT a une **expiration** (on-chain ou en base) et n'est plus valable après la date,
  - soit `status_abonnement` est géré off-chain et doit être "actif" pour que le NFT donne réellement accès.[^5][^6]

**Technique (intégration avec l'existant)**

- Garder la couche **user / abonnement actuelle** mais changer la condition "abonné actif" en :
  - `wallet possède le NFT requis` **ET** `subscription_status = actif` (ou `now < expires_at`).[^7][^8]
- Ajouter :
  - un login wallet + signature,
  - une route `GET /api/membership/status` qui renvoie `hasNft`, `roles`, `subscriptionStatus`,
  - un middleware d'API/UI qui protège les zones payantes en fonction de ces infos.[^9][^1]

---

# 2. Mini cahier de specs (.md) pour brainstorm technique

```
# Feature: Abonnement NFT K Bour Crypto (intégration avec l'existant)

## Objectif
Remplacer/compléter l'abonnement actuel par un **NFT d'adhésion** :
- Le NFT est la preuve on-chain de membership.
- L'accès au contenu payant utilise ce NFT + le status d'abonnement existant (pour transition douce).
[web:56][web:72]

---

## Parcours utilisateur

### 1. Rejoindre / S'abonner

- CTA existant "Rejoindre la communauté" / "S'abonner".
- Nouveau flow :
  1. Si pas connecté wallet : prompt connexion wallet (Metamask, WalletConnect, etc.).
  2. Appel `GET /api/membership/status?address=<wallet>`:
     - Retourne:
       - `hasNft: boolean`
       - `roles: string[]` (dérivés des NFTs)
       - `subscriptionStatus: 'none' | 'active' | 'expired'`
  3. Cas:
     - `hasNft && subscriptionStatus === 'active'` → accès direct à l'espace membre.
     - `hasNft && subscriptionStatus === 'expired'` → page "Renouveler" (paiement + éventuellement MAJ expiration).
     - `!hasNft` → redirection vers `/membership/nft` (page d'achat/mint du NFT).
[web:56][web:75]

### 2. Page /membership/nft

- Affiche:
  - Explication "Ce NFT = ton abonnement K Bour Crypto".
  - Bouton "Acheter / Mint NFT".
- Action:
  - Déclenche une transaction vers le smart contract NFT (mint) ou lien vers une marketplace.
  - Après confirmation, re-check `GET /api/membership/status`:
    - Si `hasNft = true`, on met/maj `subscriptionStatus` à `active` et on redirige vers l'espace membre.
[web:59][web:72]

### 3. Connexion ultérieure

- Bouton "Se connecter" :
  - Connexion wallet + signature (SIWE-like).
  - Backend vérifie:
    - signature,
    - possession NFT,
    - `subscriptionStatus`.
  - Si OK, création d'une session (JWT ou cookie) contenant :
    - `walletAddress`
    - `roles`
    - `subscriptionStatus`.
[web:64][web:32]

---

## Modèle de données (côté serveur)

### Table `users`
- `id`
```

```
- `wallet_address` (nullable au début, obligatoire ensuite)
- autres champs existants (profil, email si déjà là, etc.)
- lien éventuel vers `subscriptions` existante

### Table `subscriptions` (existant + adaptation)
- `id`
- `user_id` ou `wallet_address`
- `status` : `active` | `expired` | `cancelled`
- `expires_at` : datetime (gère l'interruption / fin de période)
- autres colonnes business déjà en place

### Mapping NFT → rôles (en mémoire ou table)

- Config statique (ex YAML/JSON) ou table `nft_roles`:

  - `contract_address`
  - `token_id` (optionnel, pour rôles spécifiques)
  - `trait_key` / `trait_value` (optionnel, si rôles via metadata)
  - `role` (ex: `ROLE_MEMBER`, `ROLE_PRO`, `ROLE_MODERATOR`)

[web:55][web:56]

---

## API / Backend

### 1. Auth wallet

#### `POST /api/auth/siwe/prepare`
- Input: `address`
- Output: `nonce`, message à signer

#### `POST /api/auth/siwe/verify`
- Input: `address`, `message`, `signature`
- Actions:
  - Vérifier la signature.
  - Récupérer ou créer l'utilisateur lié à `address`.
  - Appeler `checkMembership(address)` (voir ci-dessous).
  - Générer un JWT/cookie avec `walletAddress`, `roles`, `subscriptionStatus`.
[web:64][web:32]

### 2. Membership

#### `GET /api/membership/status?address=<wallet>`
- Vérifie:
  1. Possession NFT sur la/les chaînes ciblées.
  2. Statut d'abonnement en base (`subscriptions`).
  3. Rôles associés au(x) NFT(s).
- Retour:
  ```json
  {
    "hasNft": true,
    "roles": ["ROLE_MEMBER", "ROLE_PRO"],
    "subscriptionStatus": "active"
  }
```

[web:56][web:79]

### 3. Fonctions internes

**`checkMembership(address) -> { hasNft, roles, subscriptionStatus }`**

- `hasNft`:
  - ERC-721/1155 `balanceOf(address)` sur le/les contrats de la commu.
  - Optionnellement, récupération des tokenIds + metadata pour traits → mapping vers rôles.
- `roles`:
  - Via table/config `nft_roles`.
- `subscriptionStatus`:
  - Cherche en base la subscription lié à `address`:
    - Si `expires_at` < now → `expired`.
    - Sinon, `active` ou `none`. [web:56][web:77]

---

# Middlewares d'accès

- Middleware API (et éventuellement côté SSR) qui vérifie:
  - JWT/session valide.
  - `subscriptionStatus === 'active'`.
  - Rôles requis pour la ressource (ex: `ROLE_MEMBER` pour espace privé, `ROLE_PRO` pour outils avancés). [web:56][web:80]

---

# Smart contract NFT (V1 simple)

- ERC-721 ou 1155 standard.
- Pas d'expiration on-chain dans un premier temps (gérée via `subscriptions.expires_at`).
- Champs indispensables pour le backend:
  - `contract_address`
  - standard ERC (721/1155)
```

    - event `Transfer` pour historisation (optionnel pour V1).

Évolutions possibles:

- Ajouter une logique d'**expiration on-chain** plus tard (membership NFT avec durée limitée).
- Permettre différents "tiers" via plusieurs collections ou traits. [web:98][web:105]

---

# Contraintes / Intégration avec l'existant

- Ne pas casser:
    - les comptes déjà créés,
    - le système de subscriptions existant.
- Transition:
    - Pour les abonnés actuels: lier leur compte à un wallet + (optionnel) distribuer gratuitement ou à prix réduit un NFT.
    - Pendant une phase de migration, autoriser:
        - accès via abonnement "classique" OU via NFT+subscriptionStatus.
- Petit scope prioritaire:
    - Auth wallet + signature.
    - Endpoint `membership/status`.
    - Middleware d'accès basé sur `hasNft` + `subscriptionStatus`.


```
<style type="text/css">@media print {
    *, :after, :before {background: 0 0 !important;color: #000 !important;box-shadow: none !important;text-shadow: none !im
    a, a:visited {text-decoration: underline}
    a[href]:after {content: " (" attr(href) ")"}
    abbr[title]:after {content: " (" attr(title) ")"}
    a[href^="#"]:after, a[href^="javascript:"]:after {content: ""}
    blockquote, pre {border: 1px solid #999;page-break-inside: avoid}
    thead {display: table-header-group}
    img, tr {page-break-inside: avoid}
    img {max-width: 100% !important}
    h2, h3, p {orphans: 3;widows: 3}
    h2, h3 {page-break-after: avoid}
}
html {font-size: 12px}
@media screen and (min-width: 32rem) and (max-width: 48rem) {
    html {font-size: 15px}
}
@media screen and (min-width: 48rem) {
    html {font-size: 16px}
}
body {line-height: 1.85}
.air-p, p {font-size: 1rem;margin-bottom: 1.3rem}
.air-h1, .air-h2, .air-h3, .air-h4, h1, h2, h3, h4 {margin: 1.414rem 0 .5rem;font-weight: inherit;line-height: 1.42}
.air-h1, h1 {margin-top: 0;font-size: 3.998rem}
.air-h2, h2 {font-size: 2.827rem}
.air-h3, h3 {font-size: 1.999rem}
.air-h4, h4 {font-size: 1.414rem}
.air-h5, h5 {font-size: 1.121rem}
.air-h6, h6 {font-size: .88rem}
.air-small, small {font-size: .707em}
canvas, iframe, img, select, svg, textarea, video {max-width: 100%}
body {color: #444;font-family: 'Open Sans', Helvetica, sans-serif;font-weight: 300;margin: 0;text-align: center}
img {border-radius: 50%;height: 200px;margin: 0 auto;width: 200px}
a, a:visited {color: #3498db}
a:active, a:focus, a:hover {color: #2980b9}
pre {background-color: #fafafa;padding: 1rem;text-align: left}
blockquote {margin: 0;border-left: 5px solid #7a7a7a;font-style: italic;padding: 1.33em;text-align: left}
li, ol, ul {text-align: left}
p {color: #777}</style>
```