TP DealExpress - Synthèse Rapide

Contexte

API REST inspirée de Dealabs. Utilisateurs partagent des bons plans, votent (hot/cold), commentent. Système de modération avec 3 rôles.

Stack : Express.js • MongoDB • JWT • express-validator • bcryptjs

Modèles de Données

Modèle	Champs Clés
User	username, email, password (hashé), role (user/moderator/admin), createdAt
Deal	title, description, price, originalPrice, url, category, status (pending/approved/rejected), temperature, authorId, createdAt
Comment	content, dealId, authorId, createdAt
Vote	type (hot/cold), userId, dealId, createdAt

Routes à Implémenter

Authentification (Phase 1 - 1h)

Route	Méthode	Authentification	Rôle
/api/auth/register	POST	Non	Créer compte (role: user par défaut)
/api/auth/login	POST	Non	Générer JWT avec role
/api/auth/me	GET	✅ Requise	Récupérer profil connecté

Middlewares à créer : authenticate (vérifier JWT), requireRole(roles) (vérifier rôle)

Deals (Phase 2 - 1h30)

Route	Méthode	Auth	Détail
/api/deals	GET	Non	Liste deals "approved" + pagination
/api/deals/search	GET	Non	Recherche par q (title + description)
/api/deals/:id	GET	Non	Détails + auteur + commentaires + votes
/api/deals	POST	✅	Créer deal (status: "pending")
/api/deals/:id	PUT	✅	Modifier son deal (si pending + user ou admin)
/api/deals/:id	DELETE	✅	Supprimer son deal (user ou admin)

Validation : title (5-100), description (10-500), price >= 0, category ∈ [High-Tech, Maison, Mode, Loisirs, Autre]

Filtre selon rôle :

- User/Guest : voir uniquement deals status="approved"
- Moderator/Admin : voir tous les deals

Votes (Phase 3 - 45 min)

Route	Méthode	Auth	Détail
/api/deals/:id/vote	POST	✅	Voter hot/cold (remplacer si existe)
/api/deals/:id/vote	DELETE	✅	Retirer son vote

Logique : Pas de doublon • Température = (votes hot) - (votes cold)

Commentaires (Phase 4 - 45 min)

Route	Méthode	Auth	Détail
/api/deals/:dealId/comments	GET	Non	Liste commentaires triée
/api/deals/:dealId/comments	POST	✅	Ajouter commentaire (3-500 char)
/api/comments/:id	PUT	✅	Modifier son commentaire
/api/comments/:id	DELETE	✅	Supprimer son commentaire ou admin

Populate : auteur (username uniquement)

Modération & Admin (Phase 5 - 1h)

Routes Moderator/Admin :

Route	Méthode	Rôle	Détail
/api/admin/deals/pending	GET	moderator/admin	Liste deals "pending"
/api/admin/deals/:id/moderate	PATCH	moderator/admin	Approver ou rejeter (body: status)

Routes Admin Uniquement :

Route	Méthode	Rôle	Détail
/api/admin/users	GET	admin	Liste utilisateurs + pagination
/api/admin/users/:id/role	PATCH	admin	Changer rôle (body: role)

Matrice de Permissions

```
                    User    Moderator   Admin
──────────────────────────────────────────────
Créer deal          ✅ (pending)  ✅ (pending)  ✅ (pending)
Modifier deal       Son deal     Son deal     Tous
Supprimer deal      Son deal     Son deal     Tous
Approver/Rejeter    ❌           ✅           ✅
```

```
Supprimer comment   Son cmnt     Son cmnt     Tous
Voter               ✓            ✓            ✓
Gérer users         ✗            ✗            ✓
```

Validation Obligatoire

✓ Passwords hashés bcrypt
✓ JWT valide + role dans payload
✓ Ownership vérifié (PUT/DELETE)
✓ Status filtré selon le rôle
✓ Pas de doublon vote
✓ Données validées (express-validator)
✓ Populate auteurs (sans password)

Structure Projet

```
dealexpress/
├── src/
│   ├── config/database.js
│   ├── models/ (User, Deal, Comment, Vote)
│   ├── routes/ (auth, deals, comments, admin)
│   ├── controllers/
│   ├── middlewares/ (auth, validate)
│   ├── validators/
│   └── app.js
├── .env
└── package.json
```

Timeline (5h)

Heure	Phase
H1	Config + Auth (register, login, /me, middleware)
H2	CRUD Deals + Recherche + filtrage status
H3	Votes + température
H4	Commentaires + ownership
H5	Modération + Admin users + tests

Points Clés

Calcul température : COUNT(votes.type="hot") - COUNT(votes.type="cold")

Pas de doublon vote : Avant créer, chercher findOne(userId, dealId). Si existe → update type. Sinon → create.

Filtre status :

```
let filter = {};
if (!user || user.role === "user") {
  filter.status = "approved";
}
```

Ownership check : Comparer resource.authorId === req.user._id, ou vérifier role === "admin"

Populate : Recipe.populate("authorId", "-password") pour exclure password

Livrables

✓ Code Git avec commits réguliers
✓ README avec instructions + endpoints
✓ Collection Postman avec tests
✓ 3 comptes test documentés (user, moderator, admin)
✓ .env.example

Barème (20 pts)

Architecture (3) • Auth (3) • RBAC (4) • CRUD+Search (4) • Votes (2) • Comments+Moderation (2) • Qualité (2)

Ressources

mongoosejs.com/docs/populate.html • express-validator.github.io/docs • jwt.io • postman.com