Fiche Mémo : Architecture MVC & Express

Le Flux d'une Requête (Request Lifecycle)

1. **Client** : Envoie `GET /users/1`
2. **App (app.js)** : Reçoit la requête, passe par les middlewares (Logger, BodyParser).
3. **Router (routes/)** : Dirige vers `UserController.getOne`.
4. **Controller (controllers/)** : Appelle `UserService.findById(1)`.
5. **Service (services/)** : Appelle `UserModel.findOne(1)`.
6. **Model (models/)** : Renvoie la donnée brute `{ id: 1, name: "Alex" }`.
7. **Service** : Renvoie l'objet utilisateur (peut-être transformé).
8. **Controller** : Renvoie la réponse HTTP `res.json(user)`.

Les Bonnes Pratiques (Code Pro)

Structure des Dossiers

```
src/
 ├── config/       # Variables d'env, connexion DB
 ├── controllers/  # Gestion HTTP (req, res)
 ├── errors/       # Classes d'erreurs personnalisées
 ├── middlewares/  # Logger, AuthGuard, ErrorHandler
 ├── models/       # Schémas de données (ou tableaux mock)
 ├── routes/       # Définition des URLs
 ├── services/     # Logique métier pure
 ├── utils/        # Fonctions utiles (AsyncHandler)
 └── app.js        # Configuration Express
```

Règles d'Or

- **Jamais de logique métier dans le Controller.** Le Controller ne doit faire que du passe-plat.
- **Jamais de SQL/Data dans le Controller.** C'est le rôle du Service ou du Model.
- **Gestion d'erreur centralisée.** Utilisez un middleware `errorHandler` à la fin de `app.js` et le wrapper `asyncHandler` pour éviter les `try/catch` partout.
- **Variables d'environnement.** Les secrets (API Keys, Mots de passe) vont dans `.env`, jamais dans le code.

Concepts Clés

- **Middleware :** Une fonction qui s'exécute entre la requête et la réponse. Elle peut modifier la requête, la logger, ou l'arrêter (Auth). N'oubliez pas `next()` !
- **Async/Await :** Permet d'écrire du code asynchrone (attente de BDD) qui ressemble à du code synchrone. Ne bloque pas le serveur.
- **Injection de Dépendance (Light) :** On importe les services dans les contrôleurs (`require`). C'est simple et modulaire.

Commandes Utiles

- `npm init -y` : Créer un projet.
- `npm install express dotenv` : Installer le framework et la gestion de config.
- `npm install --save-dev nodemon` : Outil de dev pour redémarrer automatiquement.
- `npm run dev` : Lancer le serveur en mode dev (si configuré dans `package.json`).