Voici notre fichier *schemas.py* qui définit les schémas Pydantic :

```python
from pydantic import BaseModel
from typing import Optional, List


# --- Schémas secondaires ---

class RatingBase(BaseModel):
    userId: int
    movieId: int
    rating: float
    timestamp: int

    class Config:
        orm_mode = True


class TagBase(BaseModel):
    userId: int
    movieId: int
    tag: str
    timestamp: int

    class Config:
        orm_mode = True


class LinkBase(BaseModel):
    imdbId: Optional[str]
    tmdbId: Optional[int]

    class Config:
        orm_mode = True


# --- Schéma principal pour Movie ---
class MovieBase(BaseModel):
    movieId: int
    title: str
    genres: Optional[str] = None

    class Config:
        orm_mode = True


class MovieDetailed(MovieBase):
    ratings: List[RatingBase] = []
    tags: List[TagBase] = []
    link: Optional[LinkBase] = None
```

```python
# --- Schéma pour liste de films (sans détails imbriqués) ---
class MovieSimple(BaseModel):
    movieId: int
    title: str
    genres: Optional[str]

    class Config:
        orm_mode = True


# --- Pour les endpoints de /ratings et /tags si appelés seuls ---
class RatingSimple(BaseModel):
    userId: int
    movieId: int
    rating: float
    timestamp: int

    class Config:
        orm_mode = True


class TagSimple(BaseModel):
    userId: int
    movieId: int
    tag: str
    timestamp: int

    class Config:
        orm_mode = True


class LinkSimple(BaseModel):
    movieId: int
    imdbId: Optional[str]
    tmdbId: Optional[int]

    class Config:
        orm_mode = True
```

Ce fichier définit les **schémas de données** utilisés dans ton API pour :

- structurer ce que l'API **envoie** (en sortie, dans les réponses),
- valider ce que l'API **reçoit** (en entrée, dans les requêtes POST/PUT).

Ces schémas utilisent **Pydantic**, une bibliothèque de validation de données très utilisée avec FastAPI.

Ci-dessous l'explication du fichier :

- **Importations**

```
from pydantic import BaseModel
from typing import Optional, List
```

- `BaseModel` : classe de base fournie par Pydantic. Tous tes schémas doivent en hériter.
- `Optional[X]` : signifie que le champ peut être `None` (il est facultatif).
- `List[X]` : signifie qu'on attend une **liste d'éléments** de type `X`.

- **Schémas secondaires (associés aux films)**

Ces classes représentent les **données associées** à un film : notes, tags, et liens.

```
***`RatingBase`***
```

```
class RatingBase(BaseModel):
    userId: int
    movieId: int
    rating: float
    timestamp: int

    class Config:
        orm_mode = True
```

- Représente une **note** donnée par un utilisateur à un film.
- `orm_mode = True` indique à FastAPI qu'on peut créer ce schéma à partir d'un objet SQLAlchemy (ce qui est le cas dans ton projet).

```
***`TagBase`***
```

Même principe que `RatingBase`, mais pour les **tags** (mots-clés ajoutés par les utilisateurs).

```
***`LinkBase`***
```

Représente les **liens externes** d'un film :

- `imdbId` (IMDb)
- `tmdbId` (The Movie DB)

- **Schéma principal pour les films**

MovieBase

```
class MovieBase(BaseModel):
    movieId: int
    title: str
    genres: Optional[str] = None

    class Config:
        orm_mode = True
```

- Schéma de base pour un film.
- Contient juste les **infos essentielles** : ID, titre et genres.

```
***`MovieDetailed`***
```

```
class MovieDetailed(MovieBase):
    ratings: List[RatingBase] = []
    tags: List[TagBase] = []
    link: Optional[LinkBase] = None
```

- Hérite de `MovieBase`.
- Ajoute les **détails imbriqués** :
  - `ratings` : liste des notes
  - `tags` : liste des tags
  - `link` : les identifiants externes (IMDb, TMDb)

➡ C'est ce schéma que tu utiliseras pour des **détails complets sur un film**, ex: `/movies/1`.

- **Schéma simplifié pour les listes de films**

```
class MovieSimple(BaseModel):
    movieId: int
    title: str
    genres: Optional[str]

    class Config:
        orm_mode = True
```

- Identique à `MovieBase` mais utilisé **sans détails imbriqués**, pour les **listes de films** (ex: endpoint `/movies`).

- **Schémas "simples" pour utiliser indépendamment**

Si tu exposes un jour des endpoints `/ratings` ou `/tags` seuls, tu pourras utiliser ces schémas :

- `RatingSimple` : une note
- `TagSimple` : un tag
- `LinkSimple` : un lien (inclut aussi `movieId` contrairement à `LinkBase`)

- **En résumé**

Schéma	Utilisé pour...
`MovieSimple`	Liste de films, sans détails
`MovieDetailed`	Détails complets d'un film
`RatingBase`	Note liée à un film (imbriquée ou non)
`TagBase`	Tag lié à un film
`LinkBase`	Lien IMDb ou TMDb (imbriqué)
`LinkSimple`	Lien utilisé seul, avec `movieId`