le cnam

# DAO
# Data Access Object
# design pattern

P. Graffion

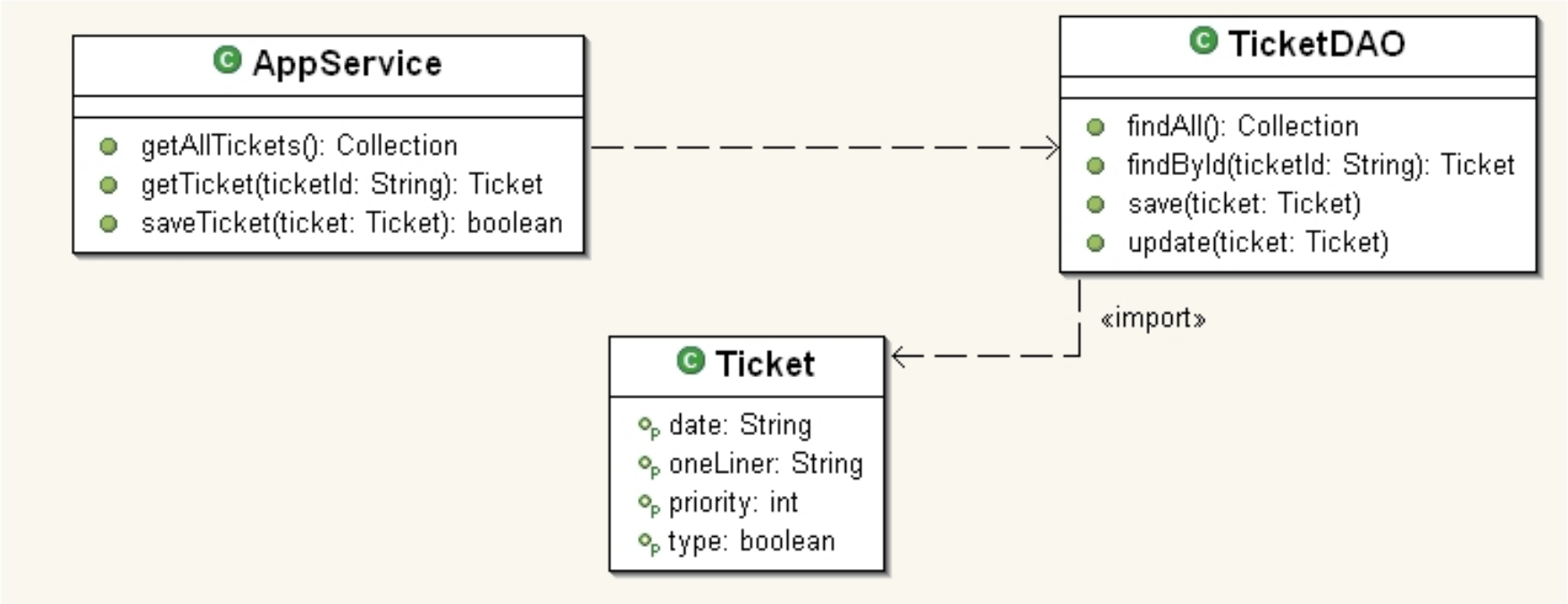
Novembre 2011

# Motivation

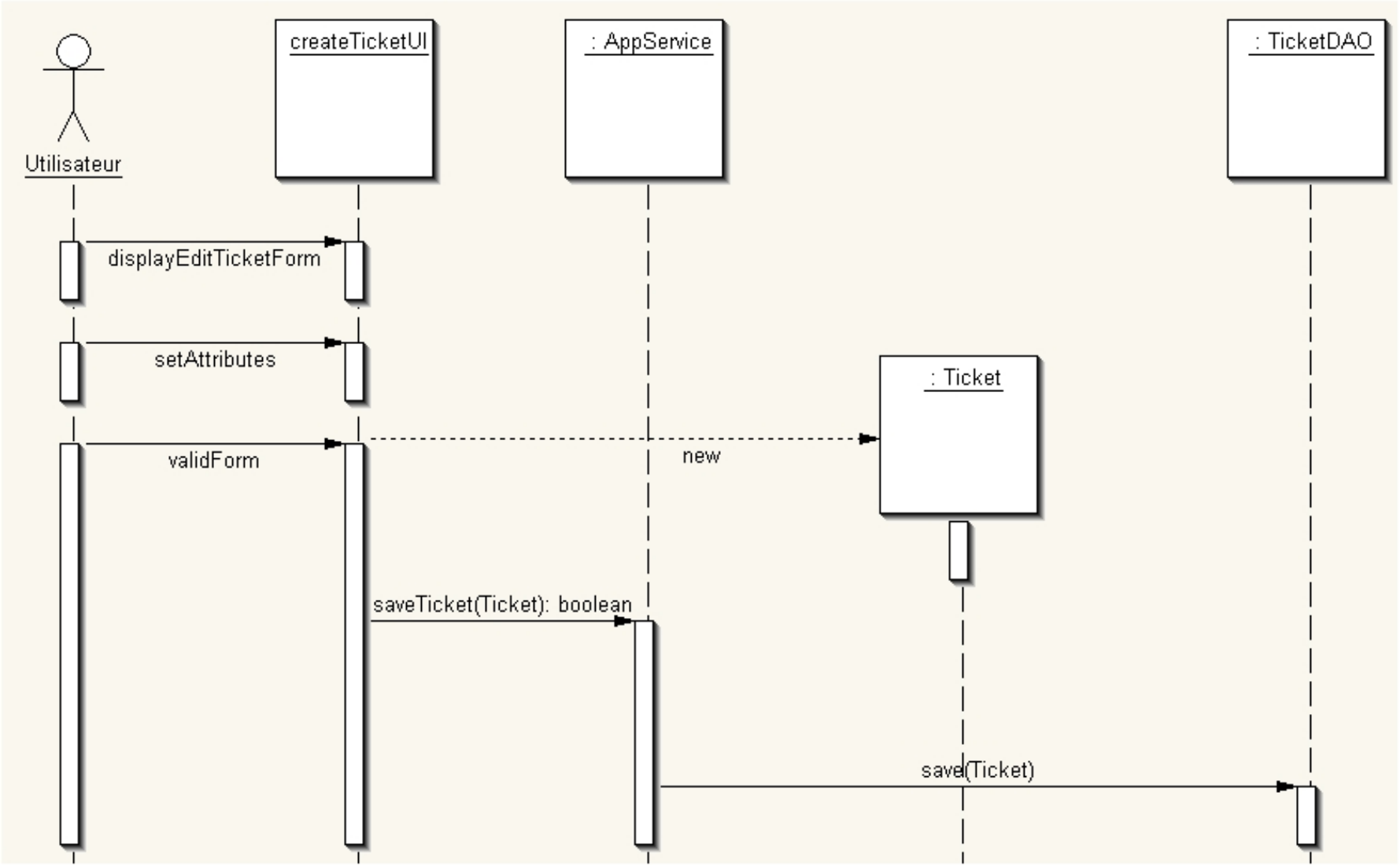
- Beaucoup d'applications ont besoin de rendre des données persistantes
- Le mécanisme de persistance utilisé peut être très divers :
  - Source de données : fichiers plats, BD, …
  - API d'accès propriétaires : MySQL, PostGres,
- On voudrait pouvoir changer de mécanisme « facilement »

# Solution

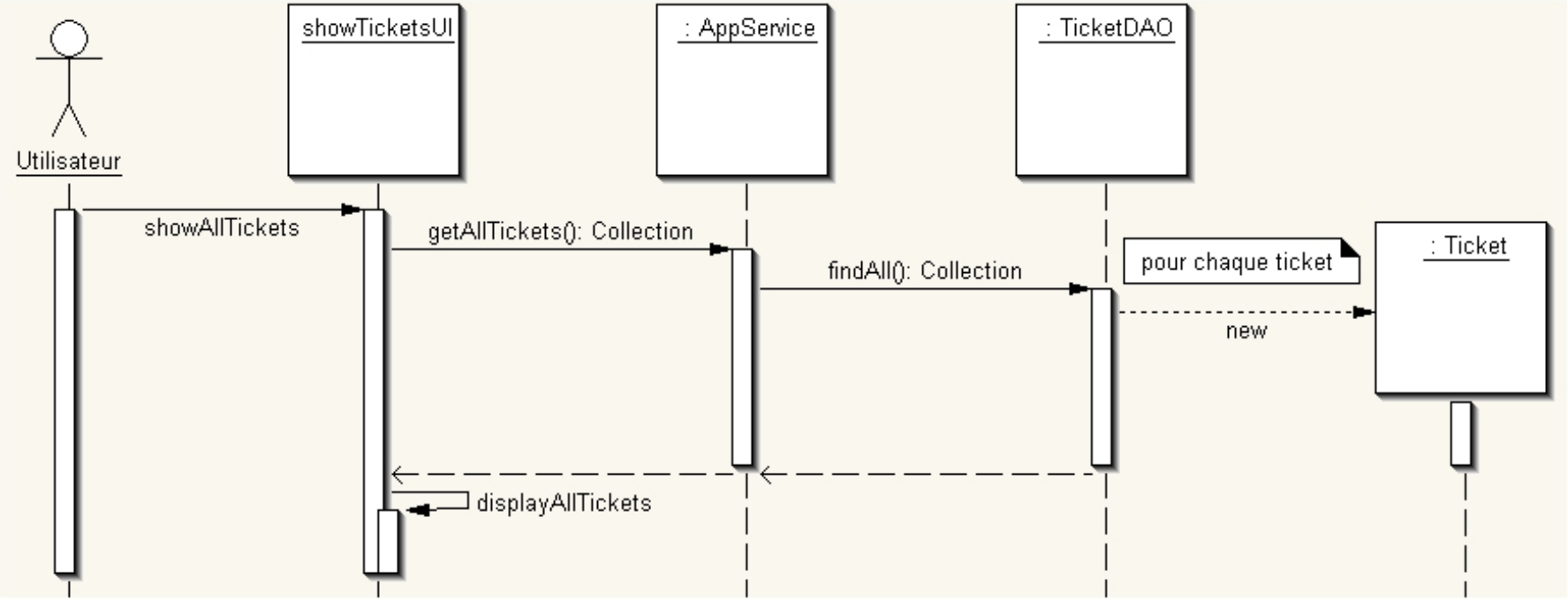
- Utiliser un Data Access Object pour encapsuler l'accès à la source de données
- Le DAO est responsable
  - d'établir la connexion à la source de données
  - de retrouver les données



# Cas d'utilisation – Créer un ticket



# Cas d'utilisation – Voir tous les tickets

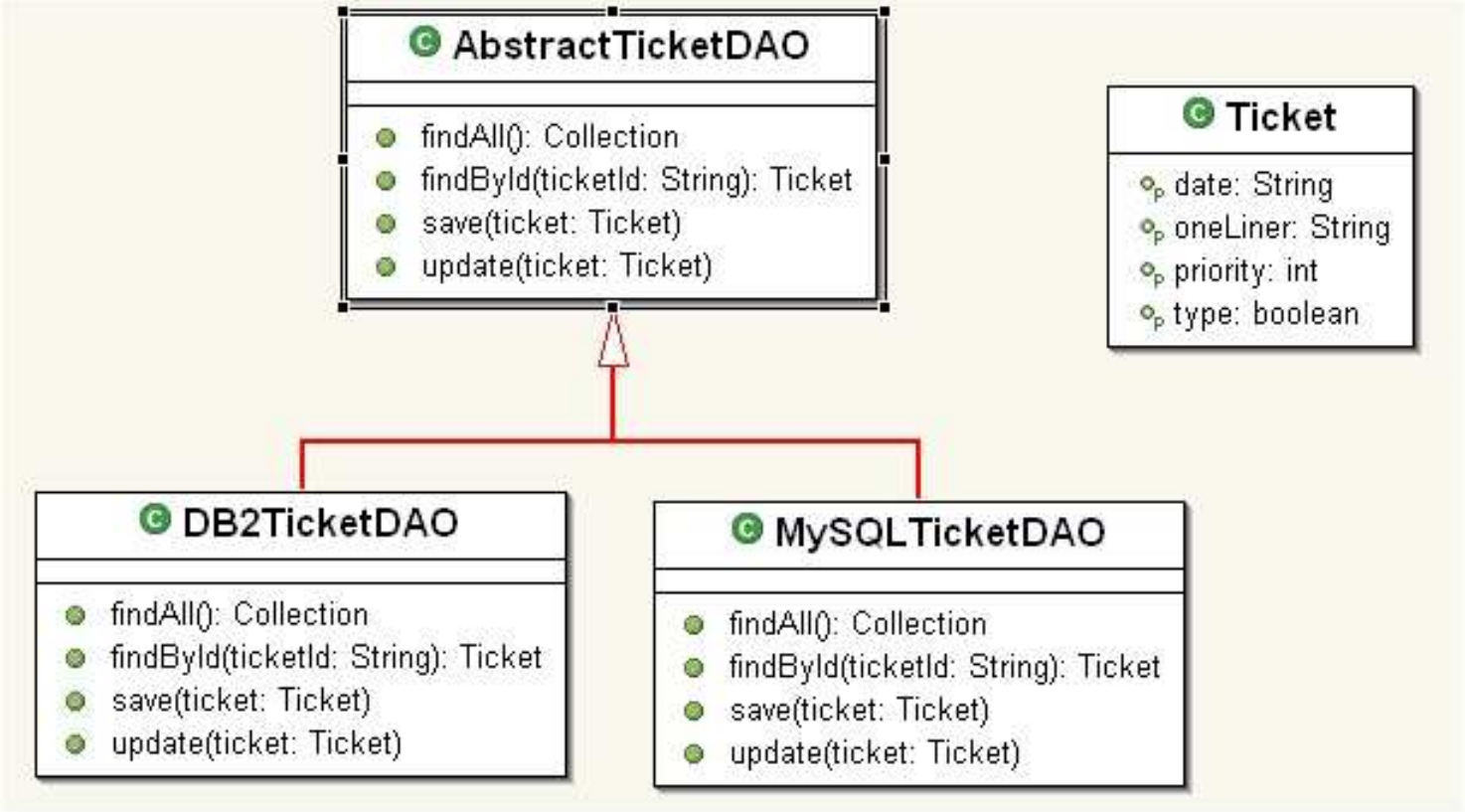


# Conséquences

- Réduit la complexité des «objets métiers» (les <<entity>>)
- Définit une couche spécifique pour l'accès aux données
- Facilite la migration vers d'autres systèmes de persistance
- Code plus évolutif et réutilisable

# DAO abstrait

- Chaque DAO peut être manipulé au travers d'une interface abstraite
  Des classes concrètes implémentent ces interfaces : MySQLTicketDAO, DB2TicketDAO



# Stratégies

- La génération des DAO peut s'automatiser : chaque «objet métier» peut avoir son propre DAO et espace de stockage (ex.: table dans SGBD)

- Possibilité pour l'application de retrouver un DAO au travers d'une fabrique de DAOs (AbstractFactory design pattern)