Conception d'applications réparties

Olivier Caron

Polytech Lille
Avenue Paul Langevin Cité Scientifique
Université de Lille
59655 Villeneuve d'Ascq cedex

http://ocaron.polytech-lille.net
Olivier.Caron@polytech-lille.fr

Unix/Linux, c’est souvent une question de vi ou de more.

Conception d'application répartie

- Problème : définir une application complexe mettant en jeu plusieurs sites, plusieurs acteurs, plusieurs applications

Conception d'application répartie

- Problème : définir une application complexe mettant en jeu plusieurs sites, plusieurs acteurs, plusieurs applications
- Solution :

Conception d'application répartie

- Problème : définir une application complexe mettant en jeu plusieurs sites, plusieurs acteurs, plusieurs applications
- Solution :
  - Spécifier l'architecture logicielle globale avant toute implémentation

Conception d'application répartie

- Problème : définir une application complexe mettant en jeu plusieurs sites, plusieurs acteurs, plusieurs applications
- Solution :
  - Spécifier l'architecture logicielle globale avant toute implémentation
  - Se préoccuper des dimensions technologiques (étudier les alternatives)

Spécification de l'architecture globale (1/2)

- Fonctions de l'application :

Spécification de l'architecture globale (1/2)

- Fonctions de l'application :
  - Identification des acteurs

Spécification de l'architecture globale (1/2)

- Fonctions de l'application :
  - Identification des acteurs
  - Identification des fonctions (cas d'utilisation UML)

Spécification de l'architecture globale (1/2)

- Fonctions de l'application :
  - Identification des acteurs
  - Identification des fonctions (cas d'utilisation UML)
- Répartition (réseau) :

Spécification de l'architecture globale (1/2)

- Fonctions de l'application :
  - Identification des acteurs
  - Identification des fonctions (cas d'utilisation UML)
- Répartition (réseau) :
  - Identification des sites et/ou types de site

Spécification de l'architecture globale (1/2)

- Fonctions de l'application :
  - Identification des acteurs
  - Identification des fonctions (cas d'utilisation UML)
- Répartition (réseau) :
  - Identification des sites et/ou types de site
  - Rôle de chaque site

Spécification de l'architecture globale (1/2)

- Fonctions de l'application :
  - Identification des acteurs
  - Identification des fonctions (cas d'utilisation UML)
- Répartition (réseau) :
  - Identification des sites et/ou types de site
  - Rôle de chaque site
  - Communication inter-sites

Spécification de l'architecture globale (1/2)

- Aspects technologiques et humains :

Spécification de l'architecture globale (1/2)

- Aspects technologiques et humains :
  - Quelle technologie ? (EJB, .Net, etc)

Spécification de l'architecture globale (1/2)

- Aspects technologiques et humains :
  - Quelle technologie ? (EJB, .Net, etc)
  - Critères (interopérabilité, performances,. . .)

Spécification de l'architecture globale (1/2)

- Aspects technologiques et humains :
  - Quelle technologie ? (EJB, .Net, etc)
  - Critères (interopérabilité, performances,. . .)
  - Combien de temps je dispose, quel est mon budget ?

Spécification de l'architecture globale (1/2)

- Aspects technologiques et humains :
  - Quelle technologie ? (EJB, .Net, etc)
  - Critères (interopérabilité, performances,. . .)
  - Combien de temps je dispose, quel est mon budget ?
  - Combien de développeurs je dispose ? quelles sont leurs compétences ?

Spécification de l'architecture globale (1/2)

- Aspects technologiques et humains :
  - Quelle technologie ? (EJB, .Net, etc)
  - Critères (interopérabilité, performances,. . .)
  - Combien de temps je dispose, quel est mon budget ?
  - Combien de développeurs je dispose ? quelles sont leurs compétences ?
- La première étape : **l'architecture logicielle**

Spécification de l'architecture globale (1/2)

- Aspects technologiques et humains :
  - Quelle technologie ? (EJB, .Net, etc)
  - Critères (interopérabilité, performances,. . .)
  - Combien de temps je dispose, quel est mon budget ?
  - Combien de développeurs je dispose ? quelles sont leurs compétences ?
- La première étape : **l'architecture logicielle**
  - Spécifications au niveau de chaque site : quels sont les composants nécessaires ?

Spécification de l'architecture globale (1/2)

- Aspects technologiques et humains :
  - Quelle technologie ? (EJB, .Net, etc)
  - Critères (interopérabilité, performances,. . .)
  - Combien de temps je dispose, quel est mon budget ?
  - Combien de développeurs je dispose ? quelles sont leurs compétences ?
- La première étape : **l'architecture logicielle**
  - Spécifications au niveau de chaque site :
    quels sont les composants nécessaires ?
    - Types des composants ?
      session (`Stateless` ou `Stateful`), web , entité, . . .

Spécification de l'architecture globale (1/2)

- Aspects technologiques et humains :
  - Quelle technologie ? (EJB, .Net, etc)
  - Critères (interopérabilité, performances,. . .)
  - Combien de temps je dispose, quel est mon budget ?
  - Combien de développeurs je dispose ? quelles sont leurs compétences ?
- La première étape : **l'architecture logicielle**
  - Spécifications au niveau de chaque site :
    quels sont les composants nécessaires ?
    - Types des composants ?
      session (`Stateless` ou `Stateful`), web , entité, . . .
    - Accessibilité des composants ?
      localement et/ou à distance ? accès sécurisé ou pas ?

Spécification de l'architecture globale (1/2)

- Aspects technologiques et humains :
  - Quelle technologie ? (EJB, .Net, etc)
  - Critères (interopérabilité, performances,. . .)
  - Combien de temps je dispose, quel est mon budget ?
  - Combien de développeurs je dispose ? quelles sont leurs compétences ?
- La première étape : **l'architecture logicielle**
  - Spécifications au niveau de chaque site :
    quels sont les composants nécessaires ?
    - Types des composants ?
      session (`Stateless` ou `Stateful`), web , entité, . . .
    - Accessibilité des composants ?
      localement et/ou à distance ? accès sécurisé ou pas ?
    - Comment accéder aux composants ? (politique de nommage)

Un exemple

Un réseau d'écoles d'ingénieurs organise le recrutement de ses étudiants de la manière suivante. Un portail web unique centralise les demandes d'inscription. Sur ce site, les candidats ont le droit de s'inscrire jusqu'à une date limite en fournissant leur adresse email, nom, prénom, date de naissance, adresse et en choisissant un mot de passe. Durant cette inscription, ils sélectionnent également une des écoles du réseau ou ils passeront une épreuve (une école est identifiée par son nom).

C'est à la charge de chaque école d'organiser cette épreuve. Pour cela, chaque école récupère la liste des candidats (email, password, nom, prénom) à faire passer, organise l'épreuve, corrige et transmet par le réseau la note finale au site central.

Chaque candidat pourra connaître sa note en se connectant soit sur le site web central soit sur le site de l'école concernée grâce à leur email et mot de passe saisis durant l'inscription. **Chaque site a pour obligation de mémoriser les informations sur les candidats pour l'année en cours.**

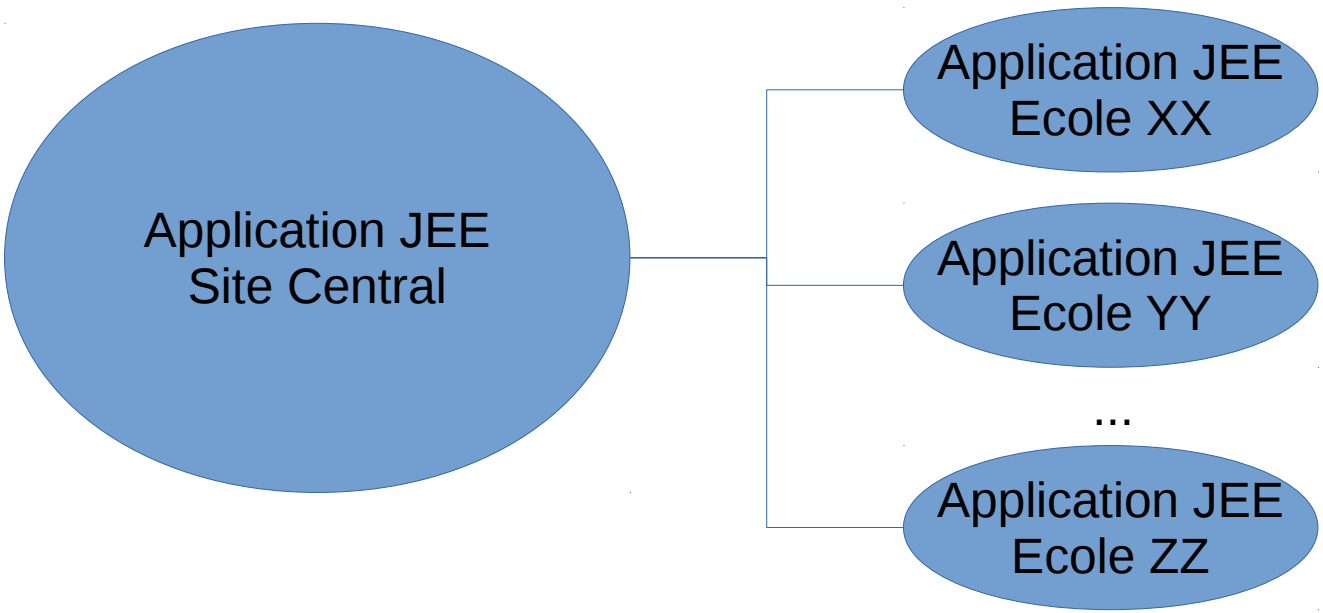
Un exemple

Question

Définissez une architecture logicielle basée sur la technologie JEE qui répond à ce problème. Décrivez très globalement (pas de détail de code) la ou les applications JEE et programmes Java nécessaires. Quels sont les composants et leur type dans chaque application JEE. Comment accède-t-on à ces applications ? Quelles sont les interactions entre ces applications ? Pour quels intervenants sont destinées ces applications.

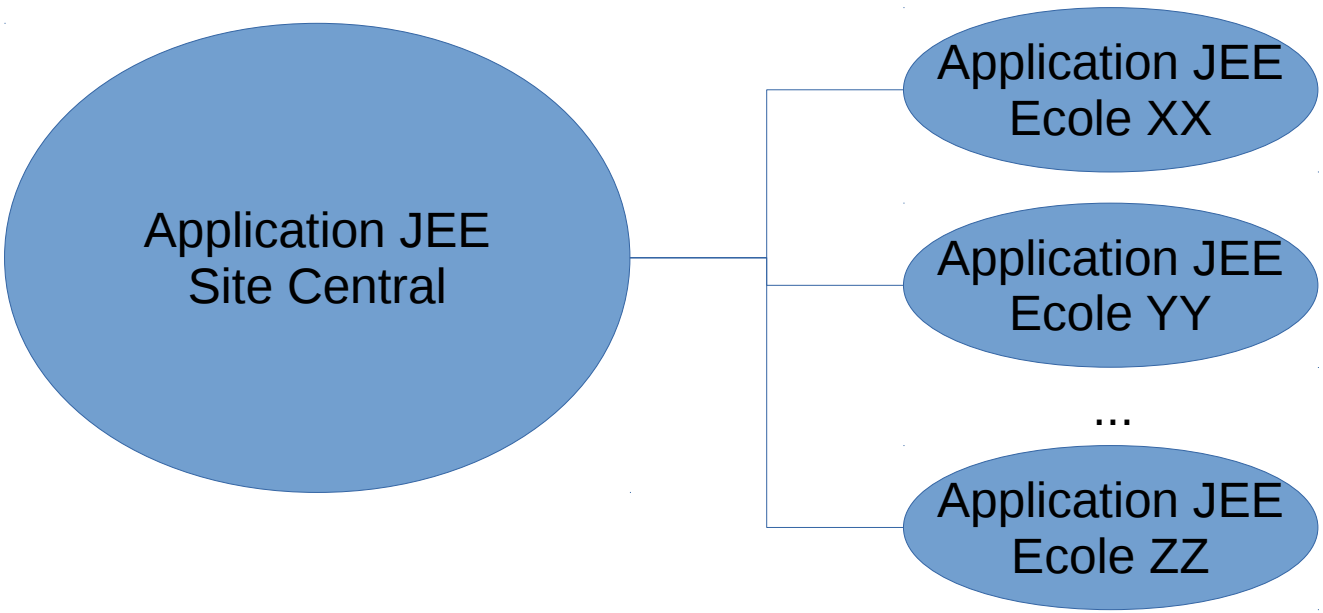
Une réponse possible

- Architecture logicielle générale : 2 types d'applications à construire :



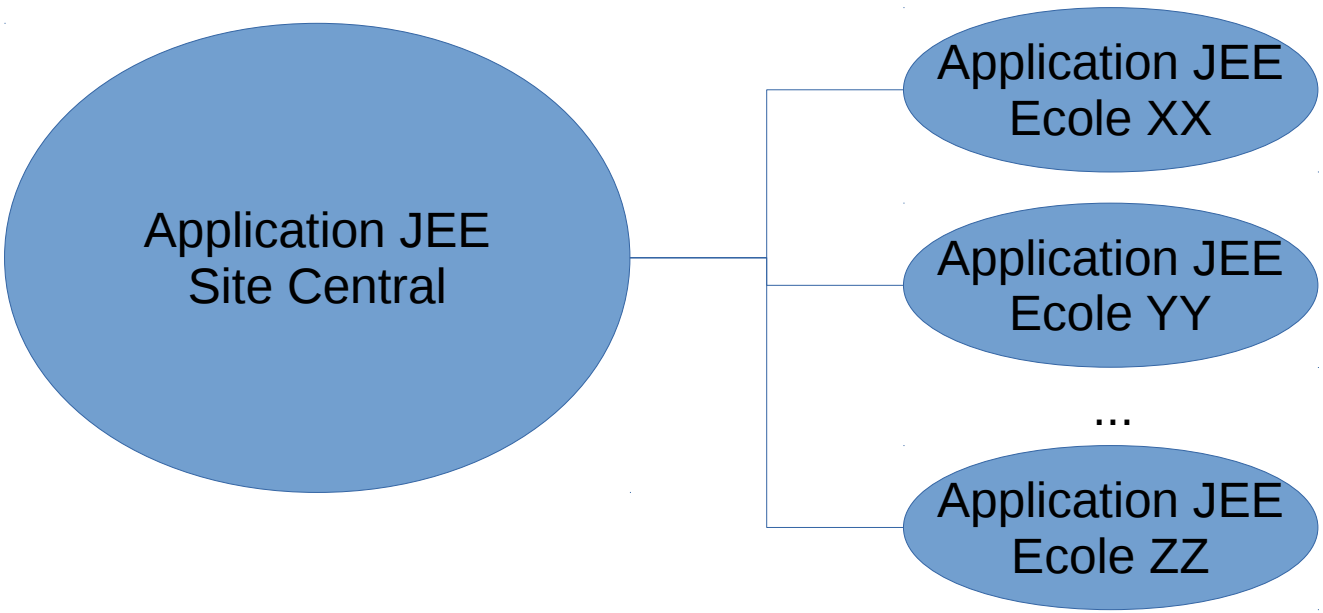
Une réponse possible

- Architecture logicielle générale : 2 types d'applications à construire :
  - Une application JEE pour le site central, gère l'inscription (via web) des étudiants, fournit des infos aux écoles et reçoit des notes



Une réponse possible

- Architecture logicielle générale : 2 types d'applications à construire :
  - Une application JEE pour le site central, gère l'inscription (via web) des étudiants, fournit des infos aux écoles et reçoit des notes
  - Une même application JEE déployée dans chaque école



Identification des données (Entités)

Exercice 1

Proposez un diagramme de classes UML détaillant les données nécessaires à gérer pour le site central.

Identification des données (Entités)

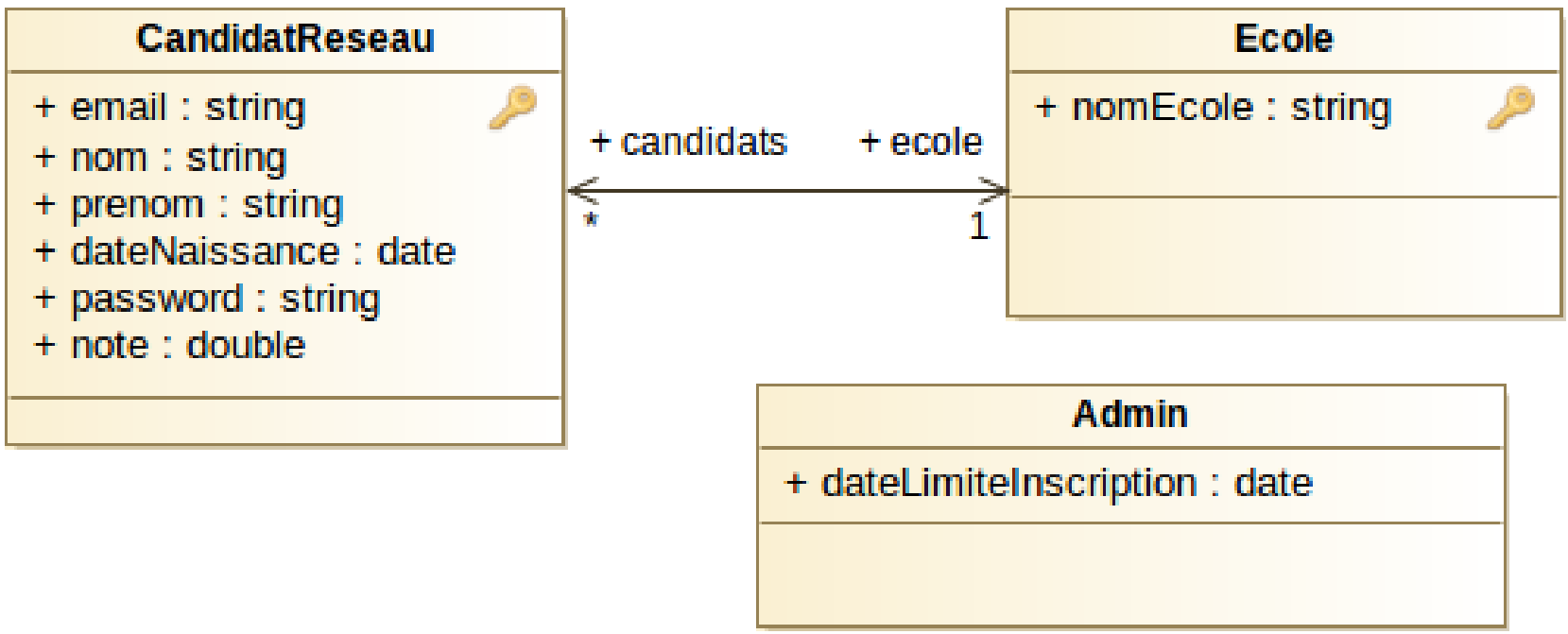
Exercice 1

Proposez un diagramme de classes UML détaillant les données nécessaires à gérer pour le site central.

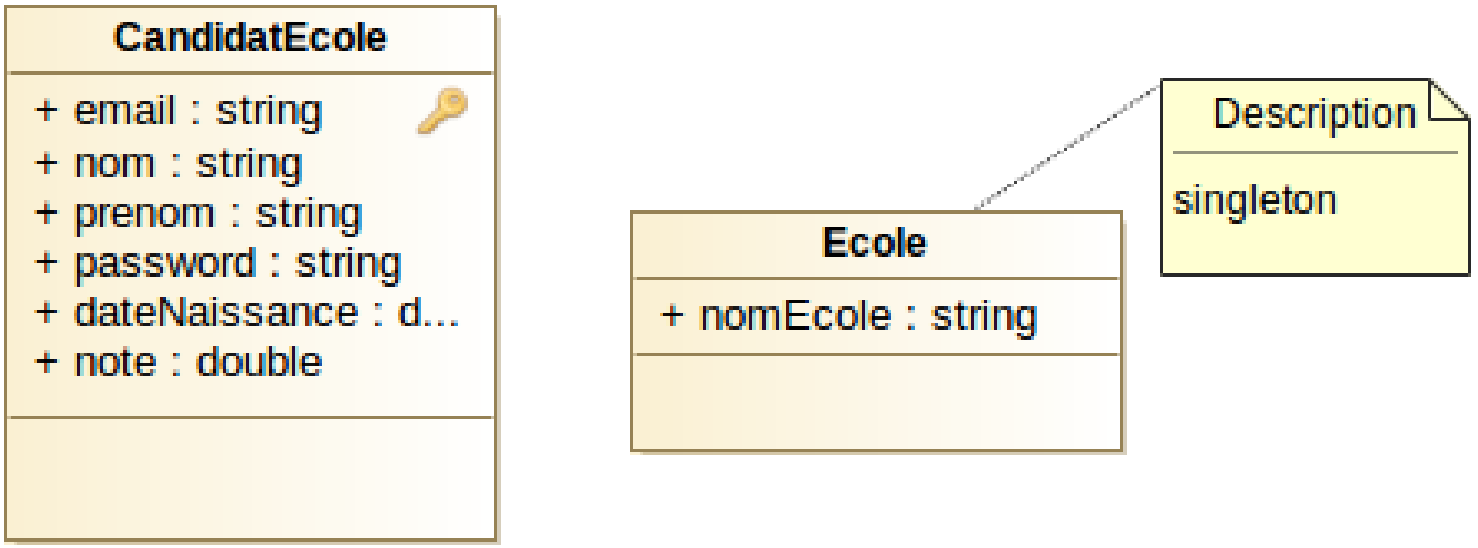
Exercice 2

Proposez un diagramme de classes UML détaillant les données nécessaires à gérer pour un site école

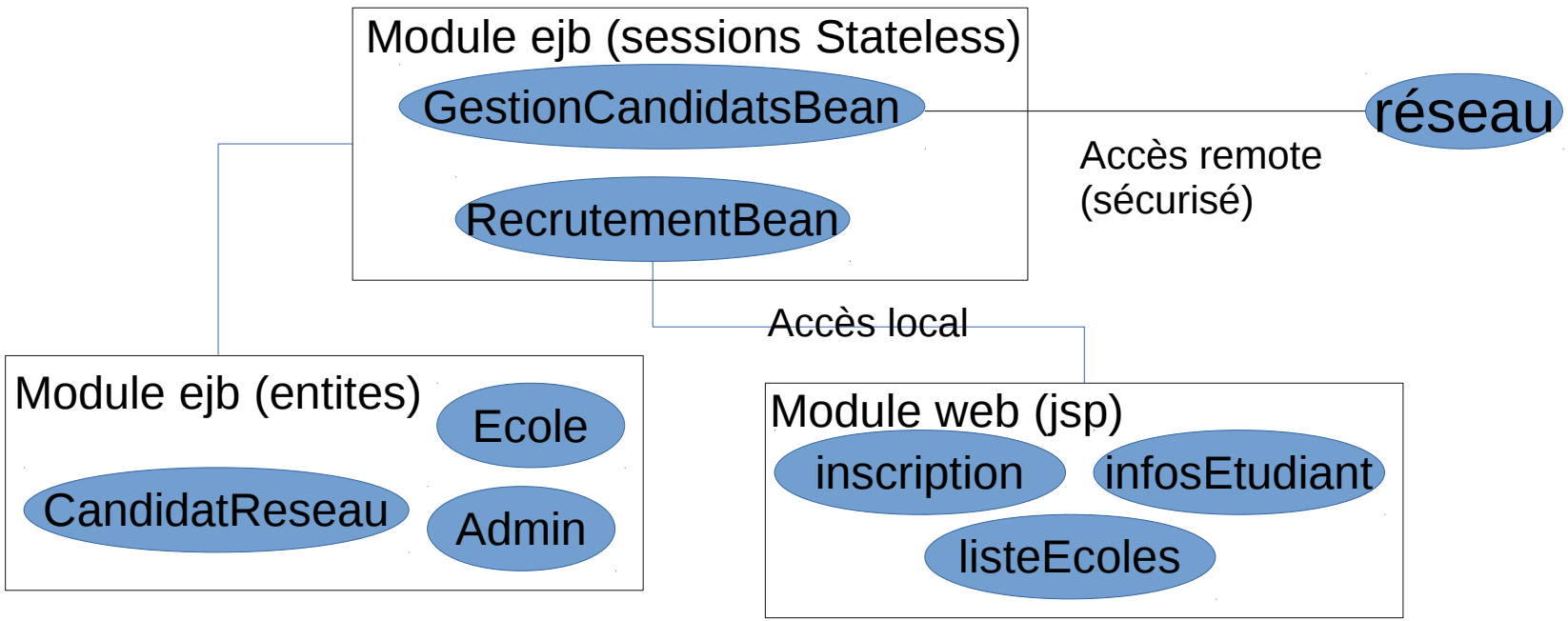
Les composants entités du site central



Les composants entités des écoles



L'application JEE sur le site central (1/2)



L'application JEE sur le site central (2/2)

- Un composant web pour gérer l'inscription

L'application JEE sur le site central (2/2)

- Un composant web pour gérer l'inscription
- Un composant web pour lister les écoles (ou utilisé dans inscription)

L'application JEE sur le site central (2/2)

- Un composant web pour gérer l'inscription
- Un composant web pour lister les écoles (ou utilisé dans inscription)
- Un composant web qui donne les infos (note,etc) sur le candidat déjà inscrit (via mot de passe)

L'application JEE sur le site central (2/2)

- Un composant web pour gérer l'inscription
- Un composant web pour lister les écoles (ou utilisé dans inscription)
- Un composant web qui donne les infos (note,etc) sur le candidat déjà inscrit (via mot de passe)
- Un composant session Stateless accessible localement pour les trois composants web : `RecrutementBean`

L'application JEE sur le site central (2/2)

- Un composant web pour gérer l'inscription
- Un composant web pour lister les écoles (ou utilisé dans inscription)
- Un composant web qui donne les infos (note,etc) sur le candidat déjà inscrit (via mot de passe)
- Un composant session Stateless accessible localement pour les trois composants web : `RecrutementBean`
- Un composant session Stateless accessible à distance uniquement pour les écoles (politique de sécurité) : `GestionCandidatsBean`

L'application JEE sur le site central (2/2)

- Un composant web pour gérer l'inscription
- Un composant web pour lister les écoles (ou utilisé dans inscription)
- Un composant web qui donne les infos (note,etc) sur le candidat déjà inscrit (via mot de passe)
- Un composant session Stateless accessible localement pour les trois composants web : `RecrutementBean`
- Un composant session Stateless accessible à distance uniquement pour les écoles (politique de sécurité) : `GestionCandidatsBean`
- Trois composants entités : `CandidatReseau`, `Ecole` et `Admin`

Le composant session `RecrutementBean`

- Accessible localement (uniquement pour composants webs)

Le composant session `RecrutementBean`

- Accessible localement (uniquement pour composants webs)
- Nommage du composant :

  `java:app/recrutementSessions/RecrutementBean!`
  `ejb.sessions.Recrutement`

  ou bien

  `@EJB Recrutement service ;`

Le composant session `RecrutementBean`

- Accessible localement (uniquement pour composants webs)
- Nommage du composant :
  `java:app/recrutementSessions/RecrutementBean!ejb.sessions.Recrutement`
  ou bien
  `@EJB Recrutement service ;`

Exercice 3

Spécifiez l'interface Java JEE `Recrutement`

Le composant session `RecrutementBean`

- Accessible localement (uniquement pour composants webs)
- Nommage du composant :
  `java:app/recrutementSessions/RecrutementBean!ejb.sessions.Recrutement`
  ou bien
  `@EJB Recrutement service ;`

Exercice 3

Spécifiez l'interface Java JEE `Recrutement`

- Développeurs de sessions et web peuvent développer leur code en parallèle une fois l'interface `Recrutement` définie.

Résultat exercice 3

Exercice 3

Spécifiez l'interface JEE `Recrutement`

```
@Local
interface Recrutement {
  public Collection<Ecole> getAllEcoles() ;
  public void inscrire(String email, String nom, prenom,
                       String password, Date naissance, String nomEcole)
    throws EtudiantDejaInscritException, DateLimiteException,
           EcoleInconnueException ;
  public CandidatReseau getInfoCandidat(String email, String password)
    throws EtudiantInconnuException, PasswordException;
}
```

Le composant session `GestionCandidatsBean`

- Accessible à distance

Le composant session `GestionCandidatsBean`

- Accessible à distance
- Nommage du composant :

  `ejb:recrutementApp/recrutementSessions/`
  `/GestionCandidatsBean!ejb.sessions.GestionCandidats`

Le composant session `GestionCandidatsBean`

- Accessible à distance
- Nommage du composant :
  `ejb:recrutementApp/recrutementSessions/`
  `/GestionCandidatsBean!ejb.sessions.GestionCandidats`
- Politique de sécurité (annotations standards JEE)

Le composant session `GestionCandidatsBean`

- Accessible à distance
- Nommage du composant :
  `ejb:recrutementApp/recrutementSessions/`
  `/GestionCandidatsBean!ejb.sessions.GestionCandidats`
- Politique de sécurité (annotations standards JEE)

Exercice 4

Spécifiez l'interface Java JEE `GestionCandidats`

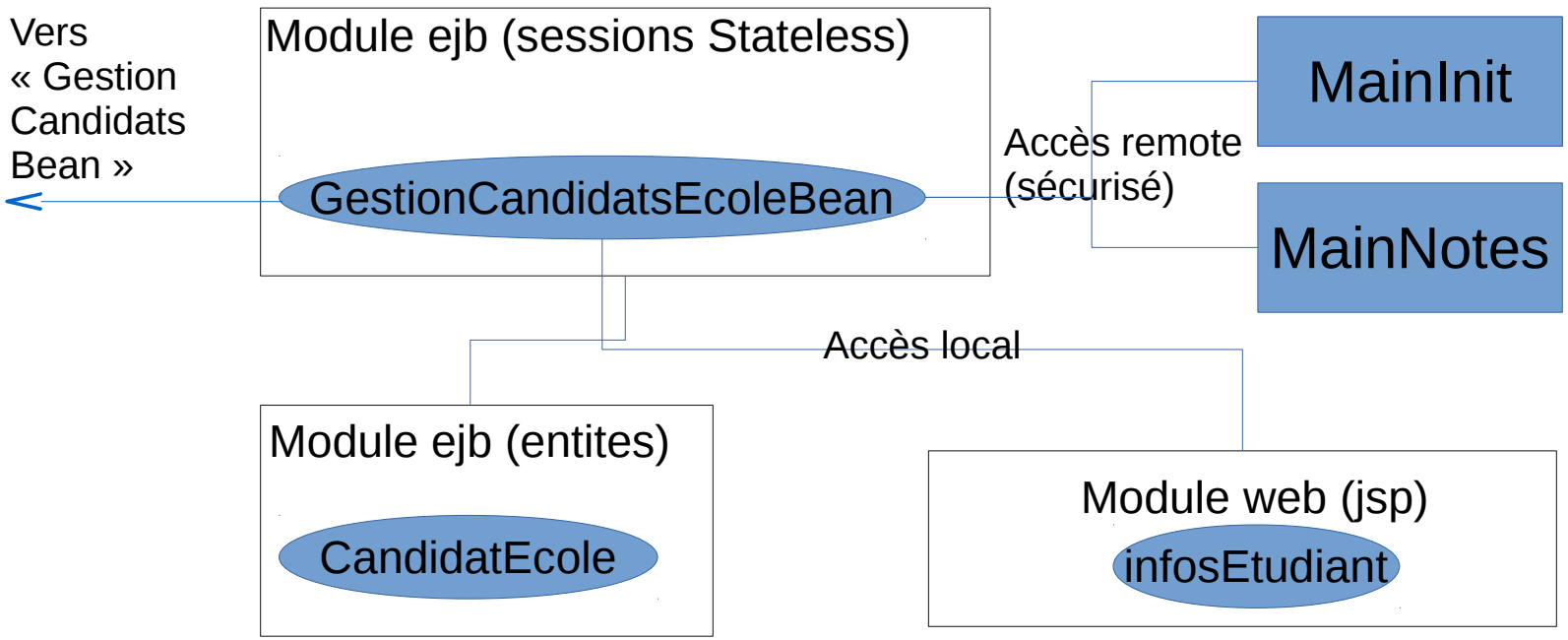
Résultat exercice 4

Exercice 4

Spécifiez l'interface JEE `GestionCandidats`

```
@Remote
interface GestionCandidats {
  public Collection<CandidatReseau> getCandidatsParEcole(String nomEcole)
  throws DateNonAtteinteException, EcoleInconnueException ;
  public void saisieNote(String emailCandidat, double note)
    throws EtudiantInconnuException
}
```

Au niveau de chaque école (1/2)



Au niveau de chaque école (2/2)

- Une application JEE contenant

Au niveau de chaque école (2/2)

- Une application JEE contenant
  - Un composant web pour fournir les infos sur les étudiants

Au niveau de chaque école (2/2)

- Une application JEE contenant
  - Un composant web pour fournir les infos sur les étudiants
  - Un composant session `GestionCandidatsEcoleBean` de type Stateless accessible :

Au niveau de chaque école (2/2)

- Une application JEE contenant
  - Un composant web pour fournir les infos sur les étudiants
  - Un composant session `GestionCandidatsEcoleBean` de type Stateless accessible :
    - localement pour le module web

Au niveau de chaque école (2/2)

- Une application JEE contenant
  - Un composant web pour fournir les infos sur les étudiants
  - Un composant session `GestionCandidatsEcoleBean` de type Stateless accessible :
    - localement pour le module web
    - à distance (mais sécurisé) pour les deux programmes java ci-dessous

Au niveau de chaque école (2/2)

- Une application JEE contenant
  - Un composant web pour fournir les infos sur les étudiants
  - Un composant session `GestionCandidatsEcoleBean` de type Stateless accessible :
    - localement pour le module web
    - à distance (mais sécurisé) pour les deux programmes java ci-dessous
  - Deux composants entité :`CandidatEcole` et `Ecole`

Au niveau de chaque école (2/2)

- Une application JEE contenant
  - Un composant web pour fournir les infos sur les étudiants
  - Un composant session `GestionCandidatsEcoleBean` de type Stateless accessible :
    - localement pour le module web
    - à distance (mais sécurisé) pour les deux programmes java ci-dessous
  - Deux composants entité :`CandidatEcole` et `Ecole`
- Deux programmes Java qui invoque chacun une méthode du composant Session `GestionCandidatsEcoleRemote`. Ce dernier accède au composant session distant `GestionCandidatsBean` :

Au niveau de chaque école (2/2)

- Une application JEE contenant
  - Un composant web pour fournir les infos sur les étudiants
  - Un composant session `GestionCandidatsEcoleBean` de type Stateless accessible :
    - localement pour le module web
    - à distance (mais sécurisé) pour les deux programmes java ci-dessous
  - Deux composants entité :`CandidatEcole` et `Ecole`
- Deux programmes Java qui invoque chacun une méthode du composant Session `GestionCandidatsEcoleRemote`. Ce dernier accède au composant session distant `GestionCandidatsBean` :

  MainInit.java récupération de la liste des candidats de l'école.

Au niveau de chaque école (2/2)

- Une application JEE contenant
  - Un composant web pour fournir les infos sur les étudiants
  - Un composant session `GestionCandidatsEcoleBean` de type Stateless accessible :
    - localement pour le module web
    - à distance (mais sécurisé) pour les deux programmes java ci-dessous
  - Deux composants entité :`CandidatEcole` et `Ecole`
- Deux programmes Java qui invoque chacun une méthode du composant Session `GestionCandidatsEcoleRemote`. Ce dernier accède au composant session distant `GestionCandidatsBean` :

  MainInit.java récupération de la liste des candidats de l'école.

  MainNotes.java Programme de saisie des notes puis transfert des notes vers le site central.

Le composant session `GestionCandidatsEcoleBean` (1/2)

- Une interface accessible localement (web)
- Nommage du composant : `java:app/recrutementSessions/GestionCandidatsEcoleBean!ejb.sessions.GestionCandidatsEcoleLocal`
  ou bien
  `@EJB GestionCandidatsEcoleLocal service ;`

```
@Local
public interface GestionCandidatsEcoleLocal {
  public CandidatEcole getInfoCandidat(String email, String password)
    throws EtudiantInconnuException, PasswordException;
}
```

Le composant session `GestionCandidatsEcoleBean` (2/2)

- Une interface accessible à distance (sécurité)
  `ejb:recrutementEcoleApp/recrutementSessions/`
  `/GestionCandidatsEcoleBean!ejb.sessions.`
  `GestionCandidatsEcoleRemote`
- Politique de sécurité (annotations standards JEE)

```
@Remote
interface GestionCandidatsEcoleRemote {
  public void initEtudiants() throws DateNonAtteinteException;
  public void saisieNote(String emailCandidat, double note)
    throws CandidatInconnuException, NoteNonValideException ;
  public void transfererNotes() ;
}
```

Un geek est une personne qui pense que 1 km correspond à 1024 mètres.