Recherche de gènes avec Python

André Espaze

17 mai 2008

Présentation du projet et des acteurs

- CNRS, Centre National de Rercherche Scientifique, de Villejuif.

Présentation du projet et des acteurs

- CNRS, Centre National de Rercherche Scientifique, de Villejuif.
  - Eric Eveno

Présentation du projet et des acteurs

- CNRS, Centre National de Rercherche Scientifique, de Villejuif.
  - Eric Eveno
  - Régine Mariage-Sanson

Présentation du projet et des acteurs

- CNRS, Centre National de Rercherche Scientifique, de Villejuif.
  - Eric Eveno
  - Régine Mariage-Sanson
- LOGILAB, prestation en informatique scientifique.

Présentation du projet et des acteurs

- CNRS, Centre National de Rercherche Scientifique, de Villejuif.
  - Eric Eveno
  - Régine Mariage-Sanson
- LOGILAB, prestation en informatique scientifique.
  - Adrien Di Mascio

Présentation du projet et des acteurs

- CNRS, Centre National de Rercherche Scientifique, de Villejuif.
  - Eric Eveno
  - Régine Mariage-Sanson
- LOGILAB, prestation en informatique scientifique.
  - Adrien Di Mascio
  - André Espaze

Présentation du projet et des acteurs

- CNRS, Centre National de Rercherche Scientifique, de Villejuif.
  - Eric Eveno
  - Régine Mariage-Sanson
- LOGILAB, prestation en informatique scientifique.
  - Adrien Di Mascio
  - André Espaze

Utilisation de Python sur un cas concret et marchand

Présentation du projet et des acteurs

- CNRS, Centre National de Rercherche Scientifique, de Villejuif.
  - Eric Eveno
  - Régine Mariage-Sanson
- LOGILAB, prestation en informatique scientifique.
  - Adrien Di Mascio
  - André Espaze

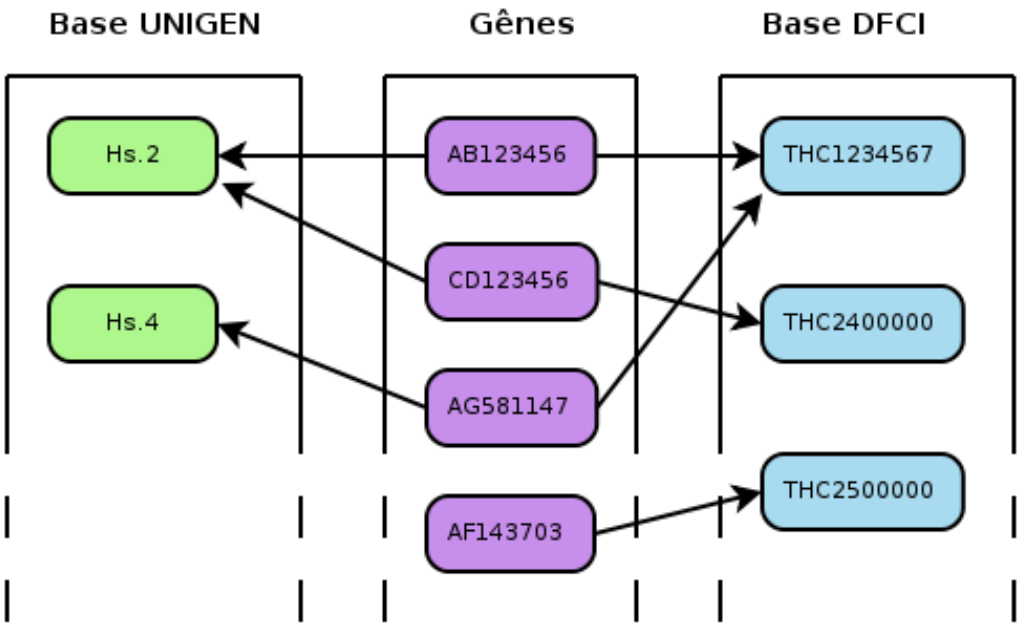
Utilisation de Python sur un cas concret et marchand

Projet d'une durée de 20 jours

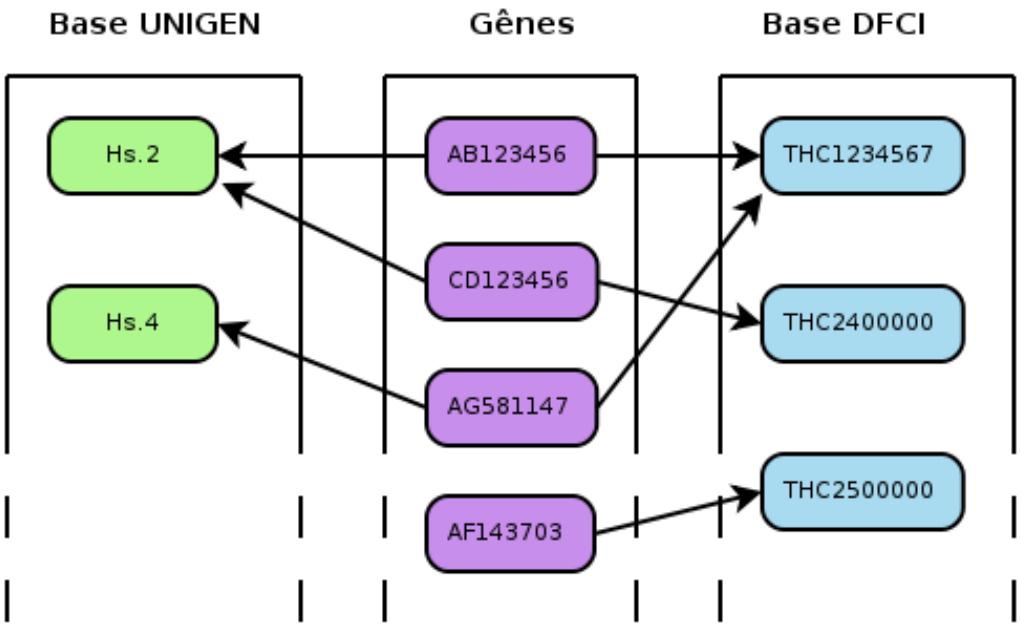
Plan de la présentation

Plan de la présentation

Bases UNIGEN et DFCI

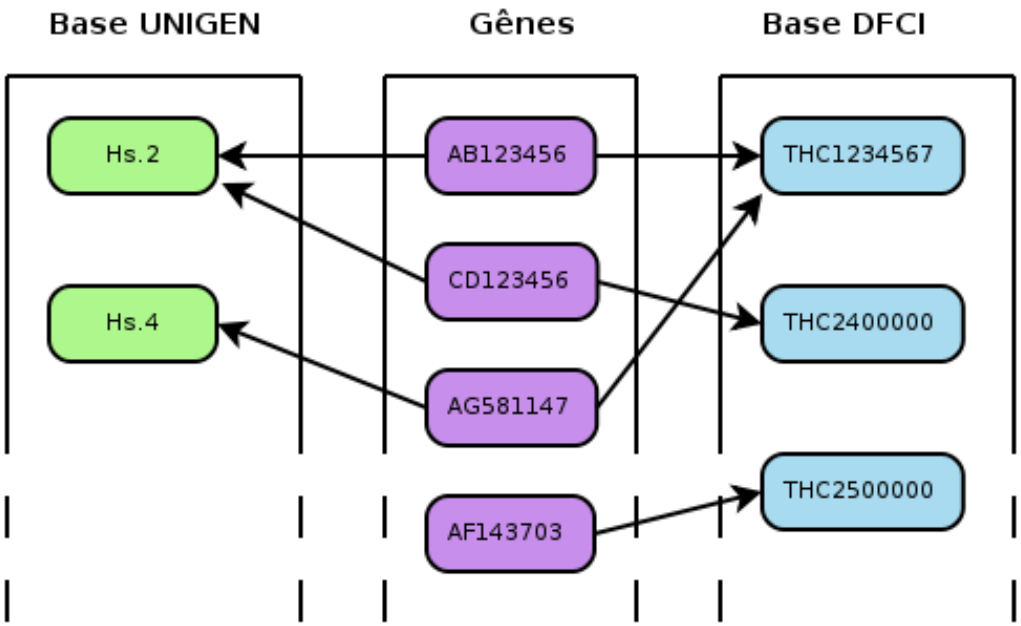


Bases UNIGEN et DFCI



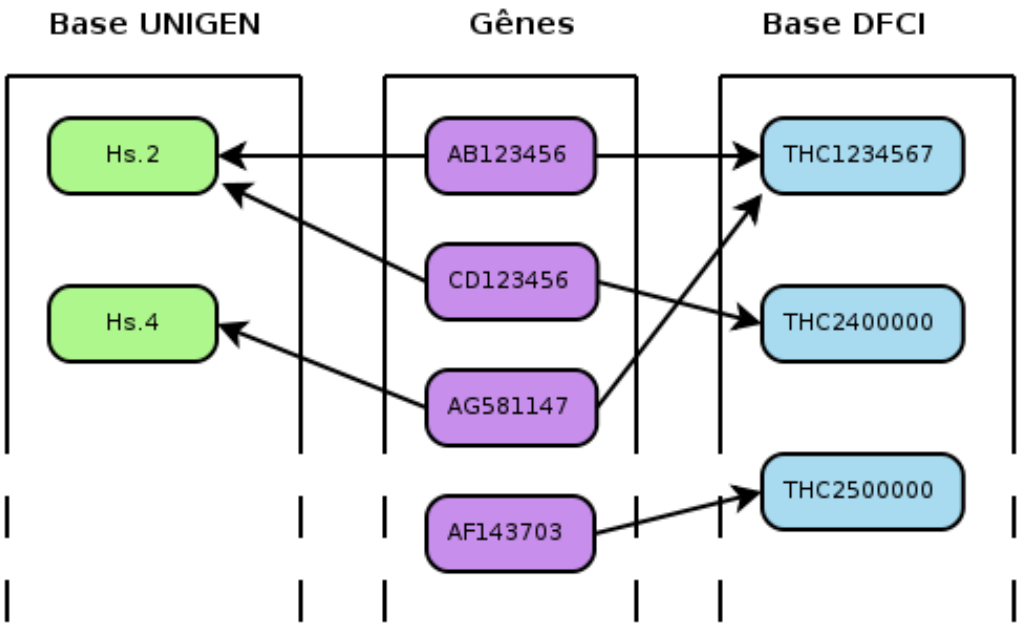
- Gènes identifiés par un code

Bases UNIGEN et DFCI



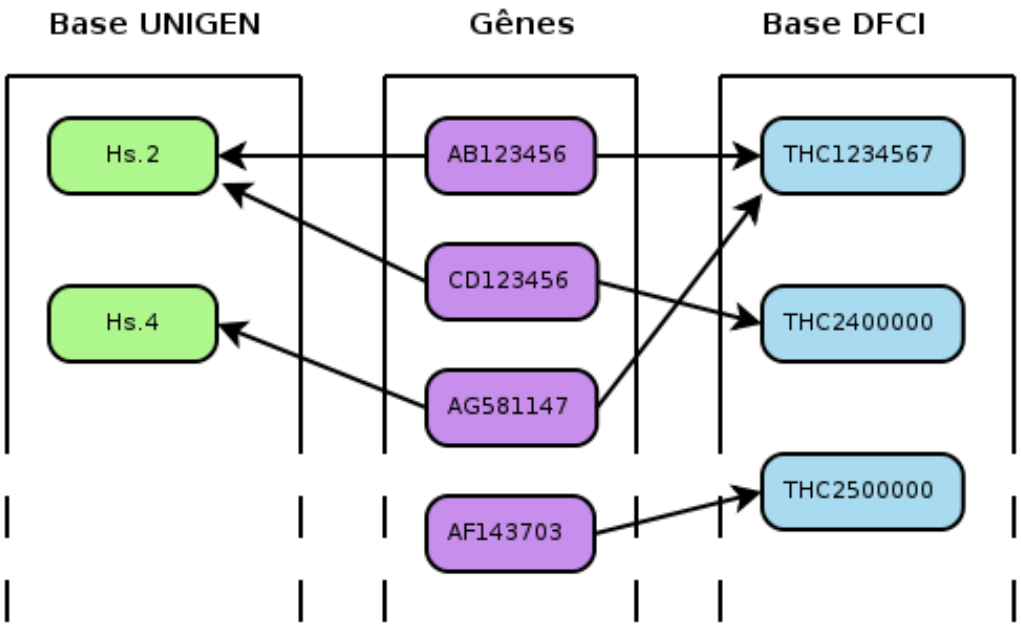
- Gènes identifiés par un code
- Gènes classés en groupes :
  - UNIGEN

Bases UNIGEN et DFCI



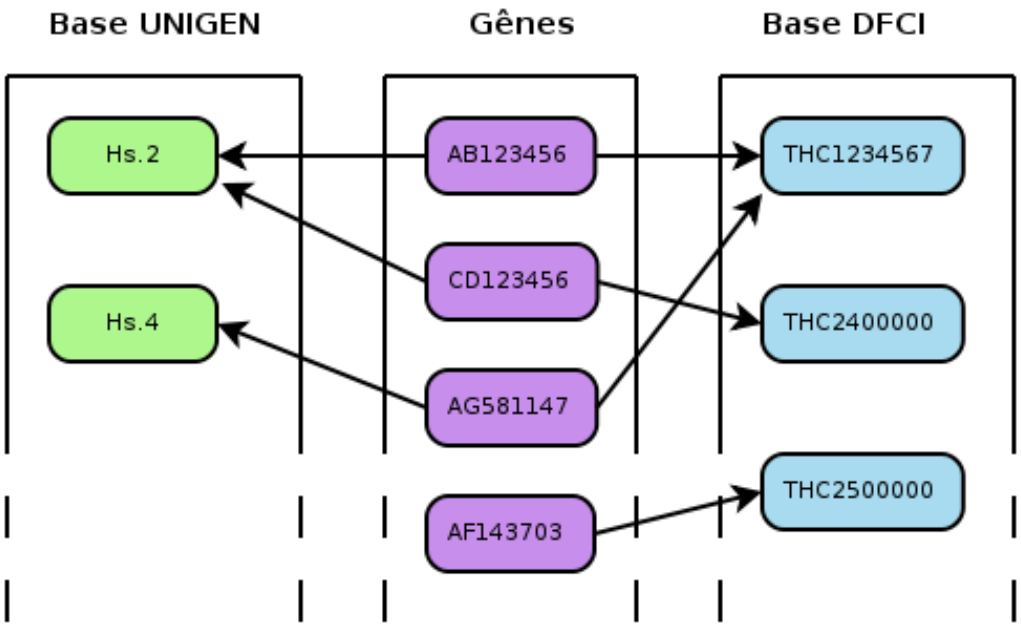
- Gènes identifiés par un code
- Gènes classés en groupes :
  - UNIGEN
  - DFCI

Bases UNIGEN et DFCI



- Gènes identifiés par un code
- Gènes classés en groupes :
  - UNIGEN
  - DFCI
- Classement différent entre les 2 bases

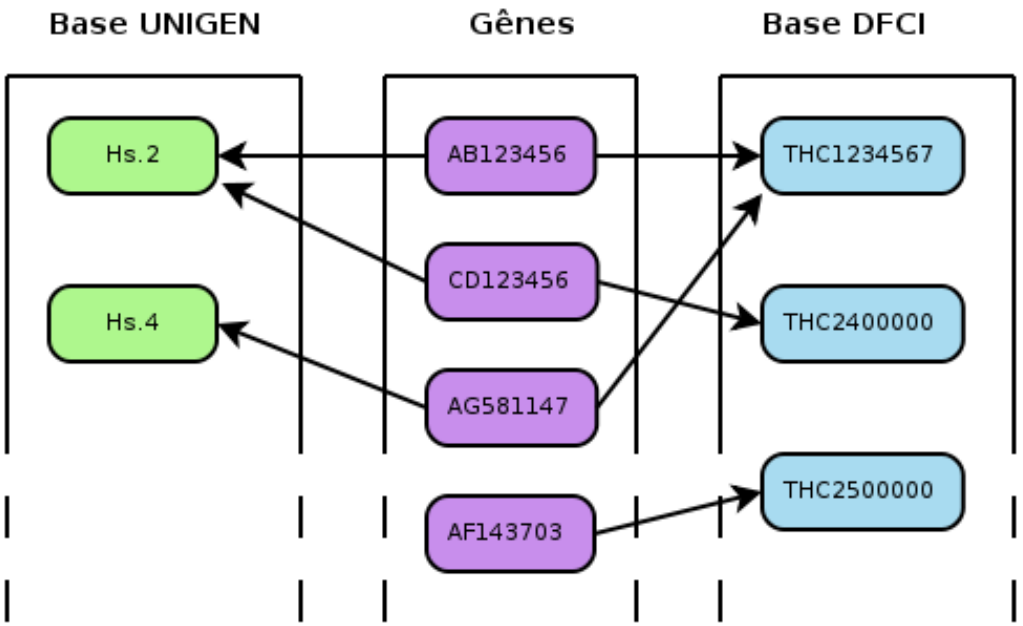
Bases UNIGEN et DFCI



- Gènes identifiés par un code
- Gènes classés en groupes :
  - UNIGEN
  - DFCI
- Classement différent entre les 2 bases

Besoin de comparaison

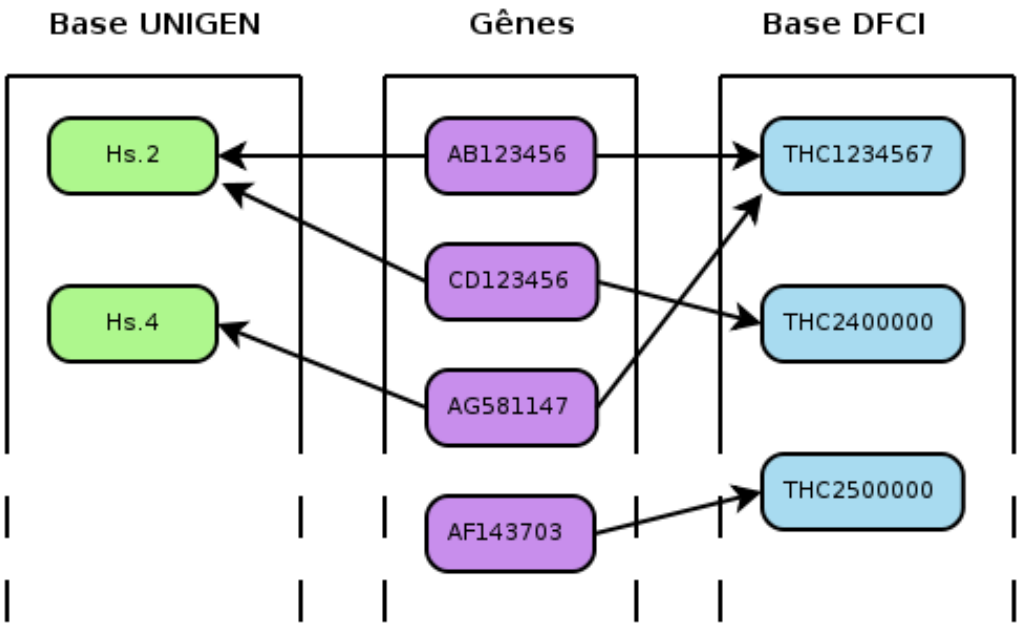
Bases UNIGEN et DFCI



- Gènes identifiés par un code
- Gènes classés en groupes :
  - UNIGEN
  - DFCI
- Classement différent entre les 2 bases

Besoin de comparaison
Information sous forme de fichiers textes

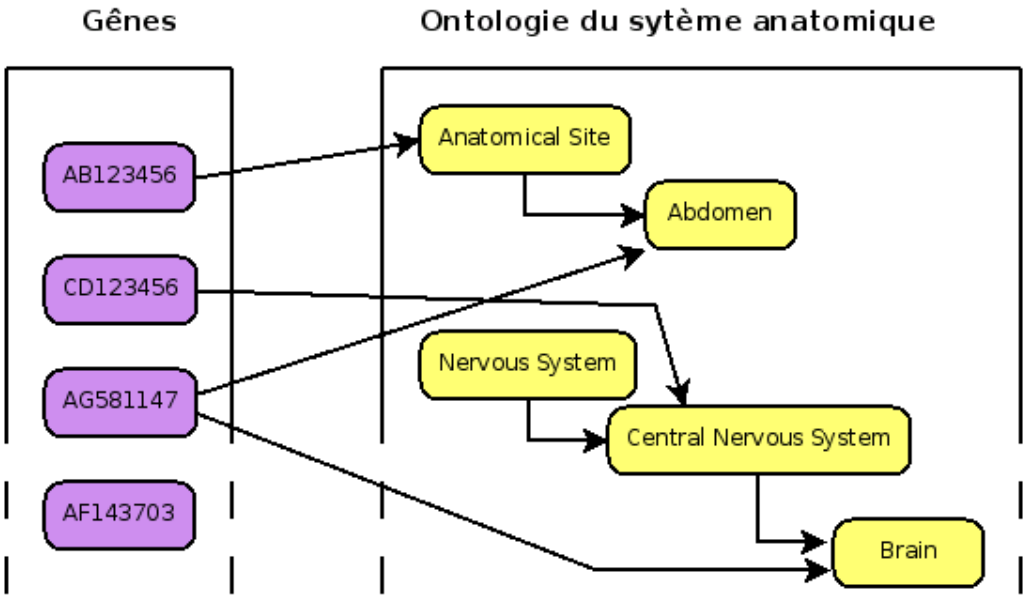
Bases UNIGEN et DFCI



- Gènes identifiés par un code
- Gènes classés en groupes :
  - UNIGEN
  - DFCI
- Classement différent entre les 2 bases

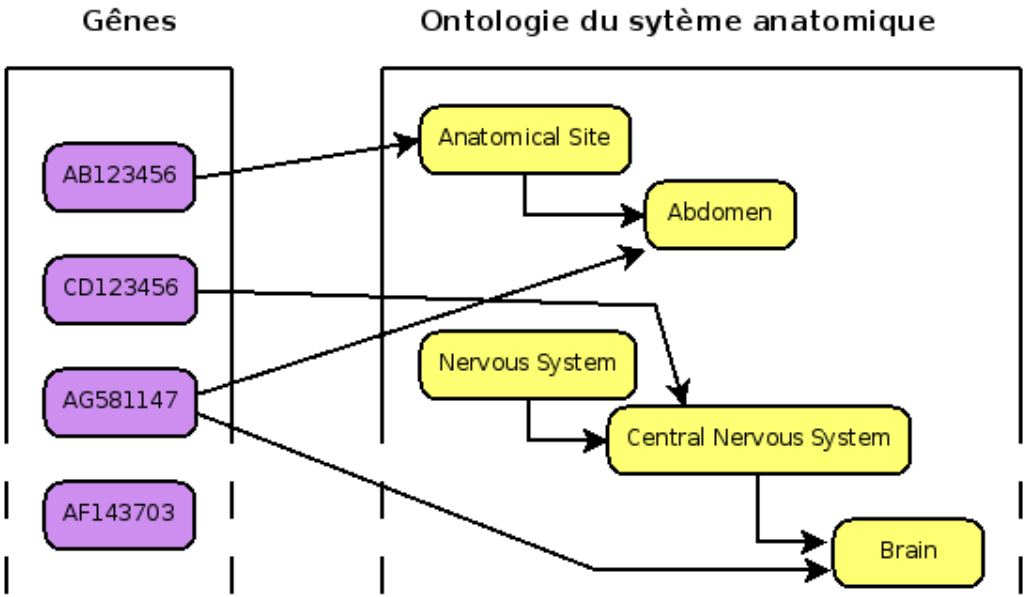
Besoin de comparaison
Information sous forme de fichiers textes
Stockage et mises à jour régulières

Les classes d'ontologie



Ontologie : le sens du gène

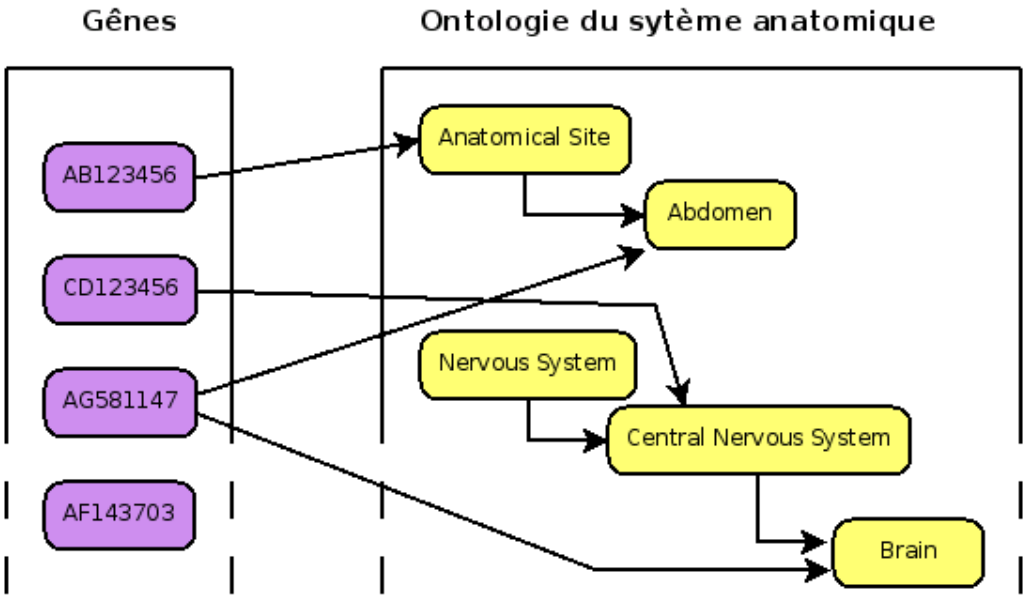
Les classes d'ontologie



Ontologie : le sens du gène

- Organisation par classes et niveaux

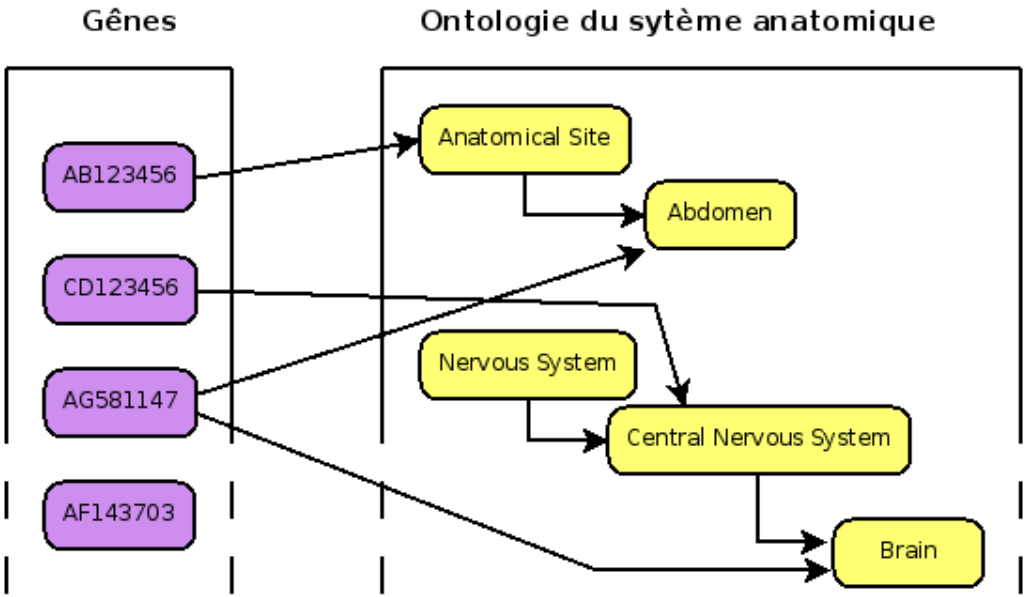
Les classes d'ontologie



Ontologie : le sens du gène

- Organisation par classes et niveaux
- Gène rattaché à une ou plusieurs classes

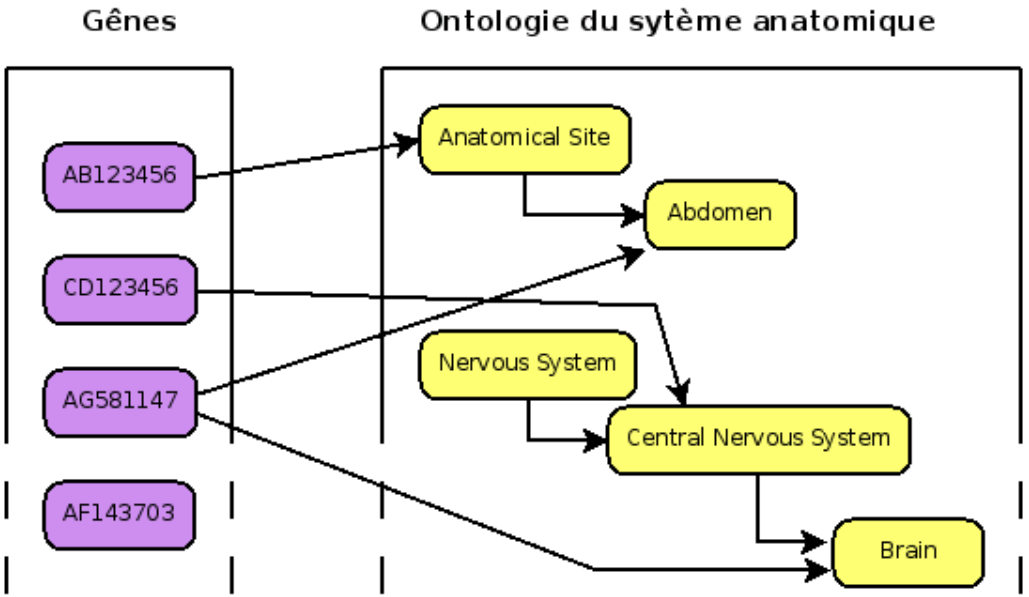
Les classes d'ontologie



Ontologie : le sens du gène

- Organisation par classes et niveaux
- Gène rattaché à une ou plusieurs classes
- Plusieurs arbres possibles (Pathologies,...)

Les classes d'ontologie

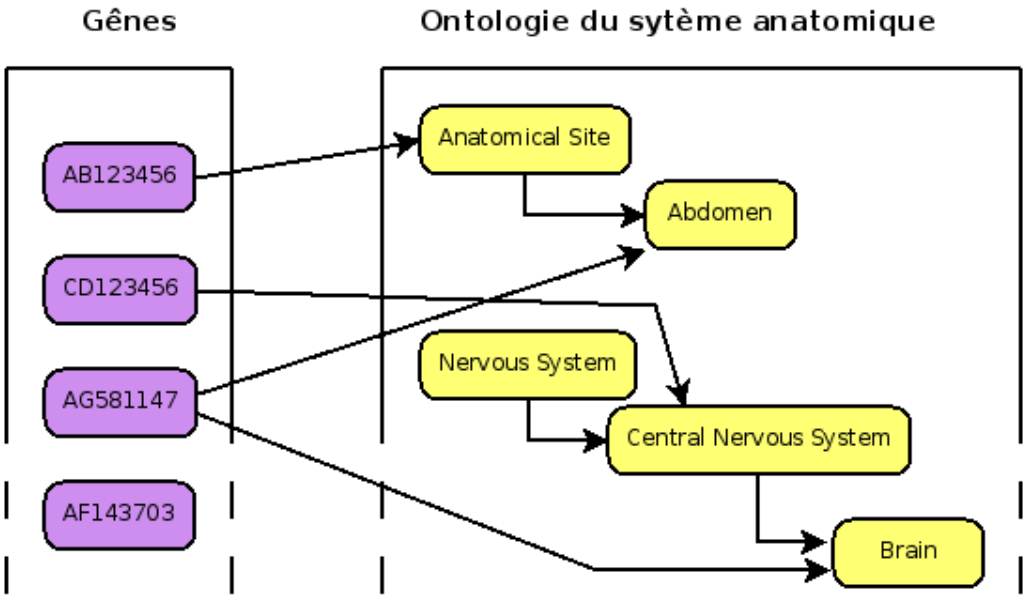


Ontologie : le sens du gène

- Organisation par classes et niveaux
- Gène rattaché à une ou plusieurs classes
- Plusieurs arbres possibles (Pathologies,...)

Information sous forme de fichiers OWL (base Evoc)

Les classes d'ontologie

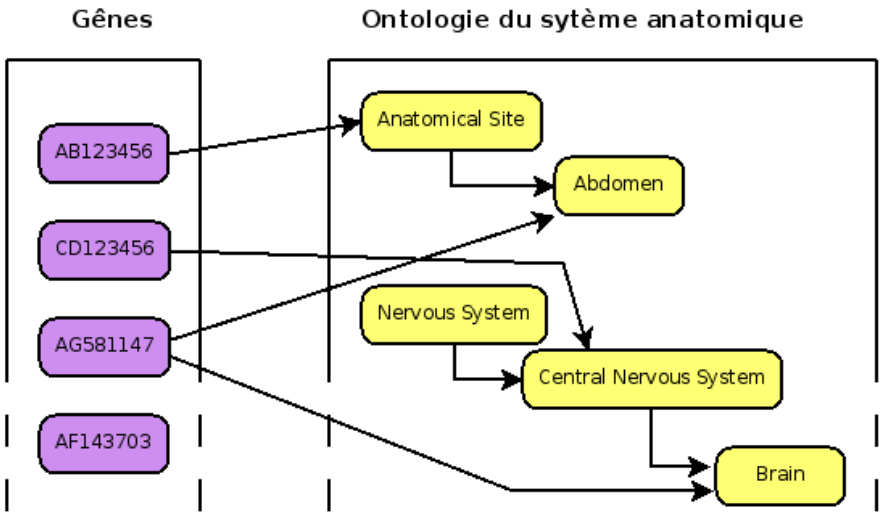


Ontologie : le sens du gène

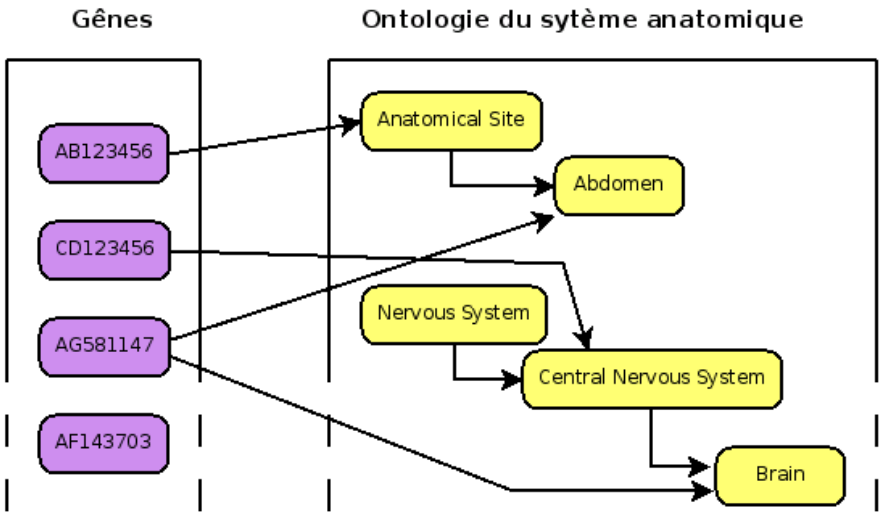
- Organisation par classes et niveaux
- Gène rattaché à une ou plusieurs classes
- Plusieurs arbres possibles (Pathologies,...)

Information sous forme de fichiers OWL (base Evoc)
Besoin de stockage

Les fréquences de liaison



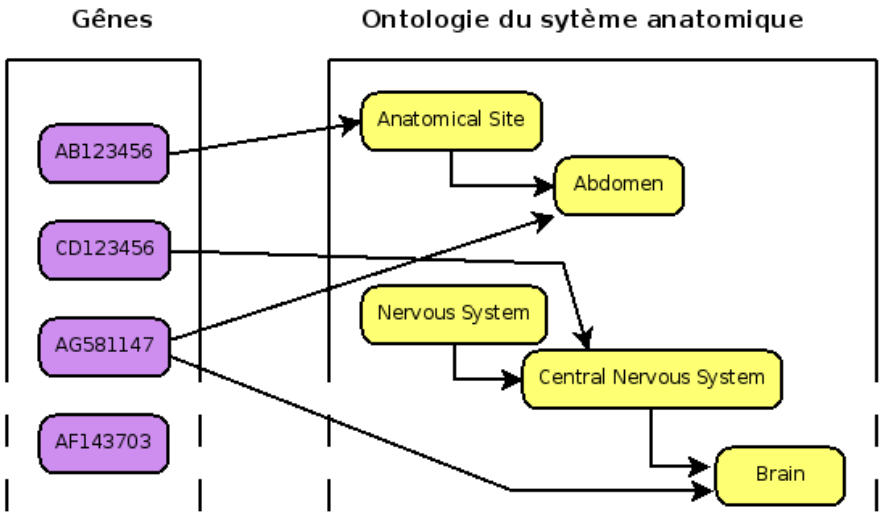
Les fréquences de liaison



Definition

Fréquence de liaison brute : nombre de relations vers une classe d'ontologie divisé par le nombre total de liaisons du groupe

Les fréquences de liaison

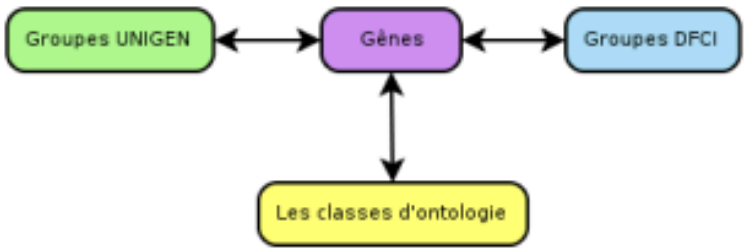


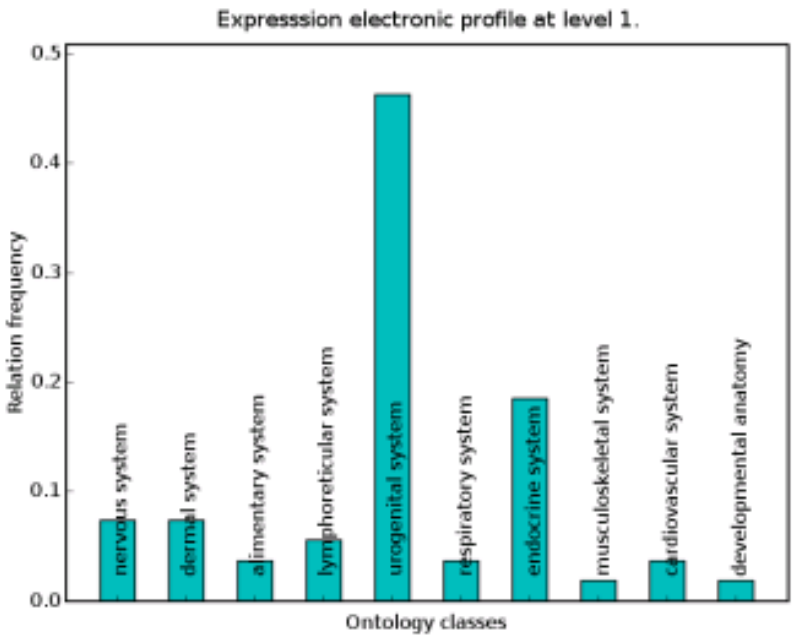
Definition

Fréquence de liaison brute : nombre de relations vers une classe d'ontologie divisé par le nombre total de liaisons du groupe

Autre calcul : fréquence de liaison normalisée (non présentée)

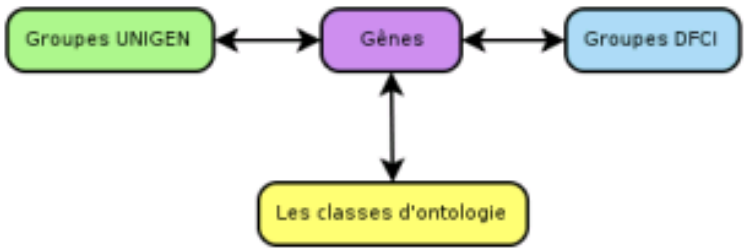
Les profils d'expression éléctronique

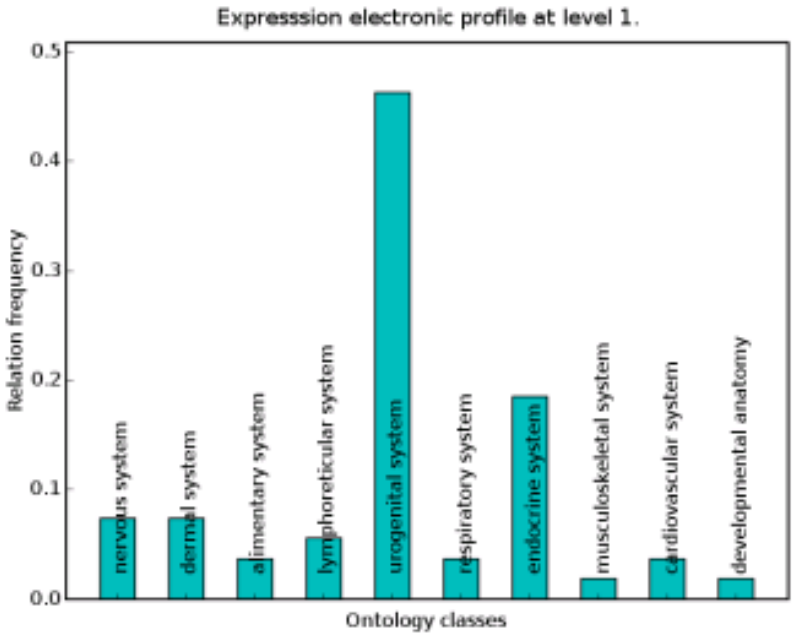




Parcourir tous les niveaux d'un arbre d'ontologie :

Les profils d'expression éléctronique

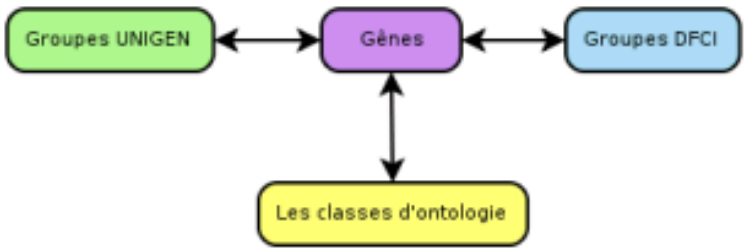


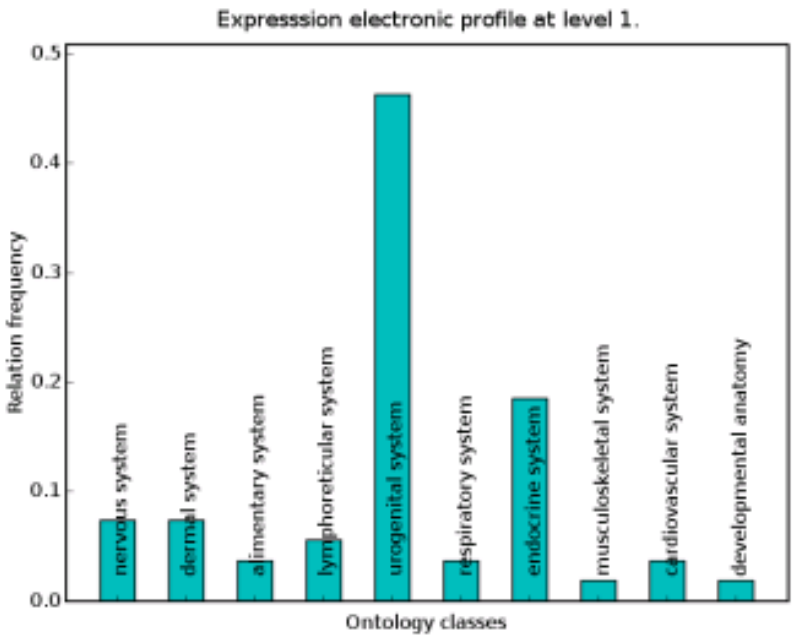


Parcourir tous les niveaux d'un arbre d'ontologie :

- Retrouver les groupes d'une base

Les profils d'expression éléctronique

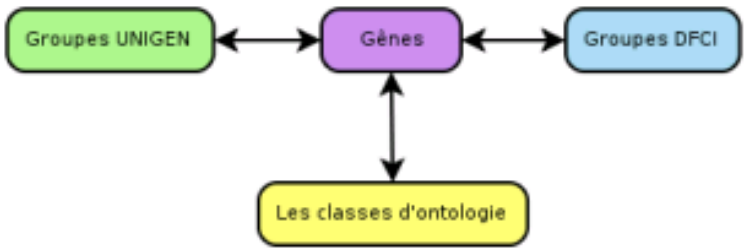


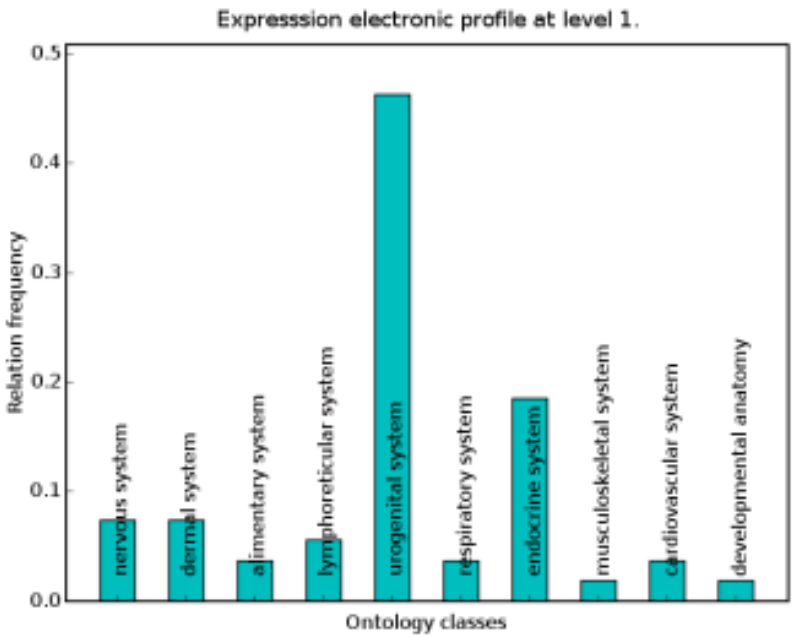


Parcourir tous les niveaux d'un arbre d'ontologie :

- Retrouver les groupes d'une base
  - Retrouver les gènes

Les profils d'expression éléctronique

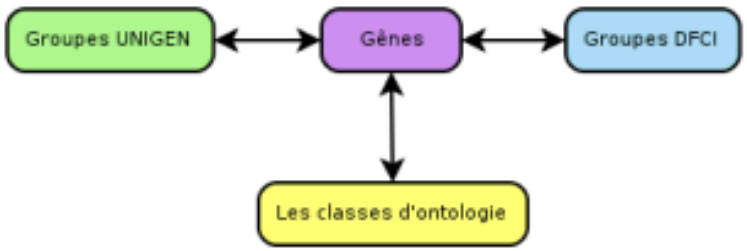


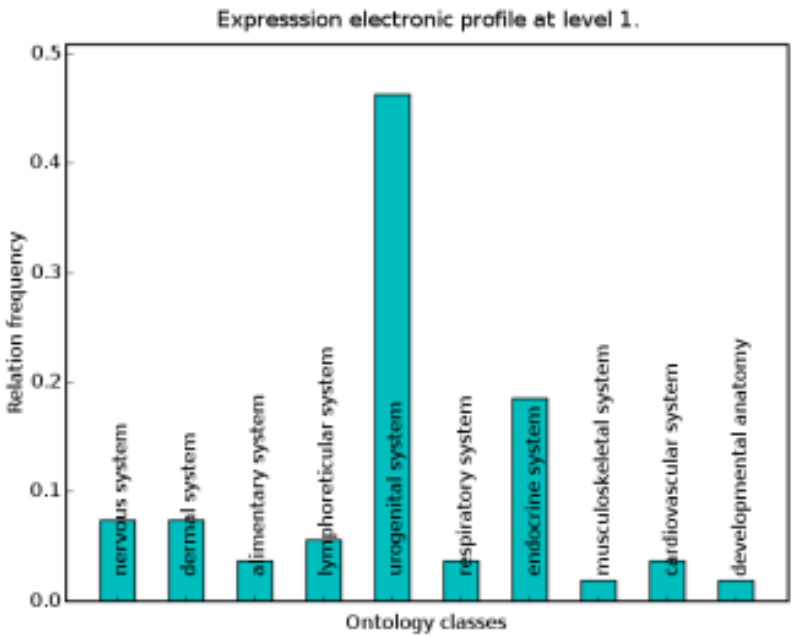


Parcourir tous les niveaux d'un arbre d'ontologie :

- Retrouver les groupes d'une base
  - Retrouver les gènes
  - Trouver ses relations vers les classes d'ontologie

Les profils d'expression éléctronique

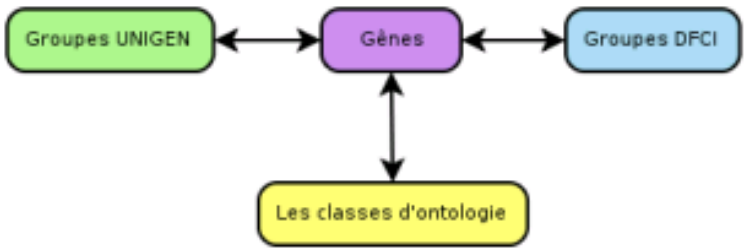


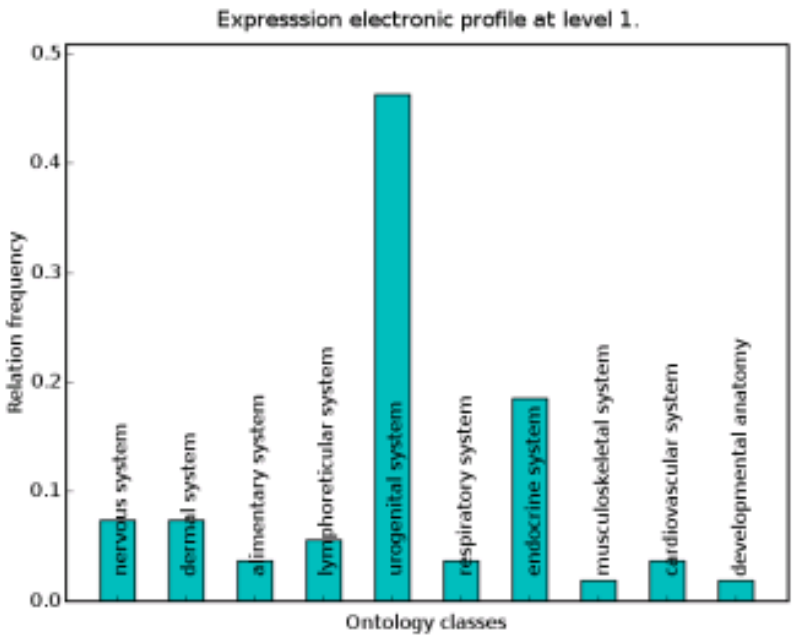


Parcourir tous les niveaux d'un arbre d'ontologie :

- Retrouver les groupes d'une base
  - Retrouver les gènes
  - Trouver ses relations vers les classes d'ontologie
- Sommer les relations

Les profils d'expression éléctronique

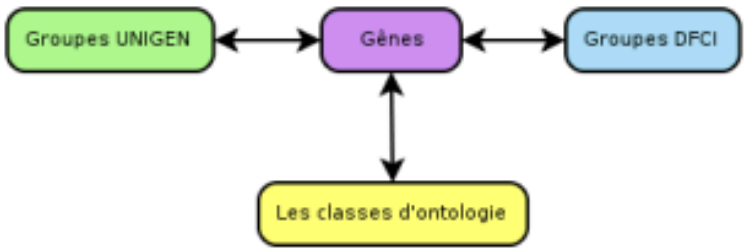


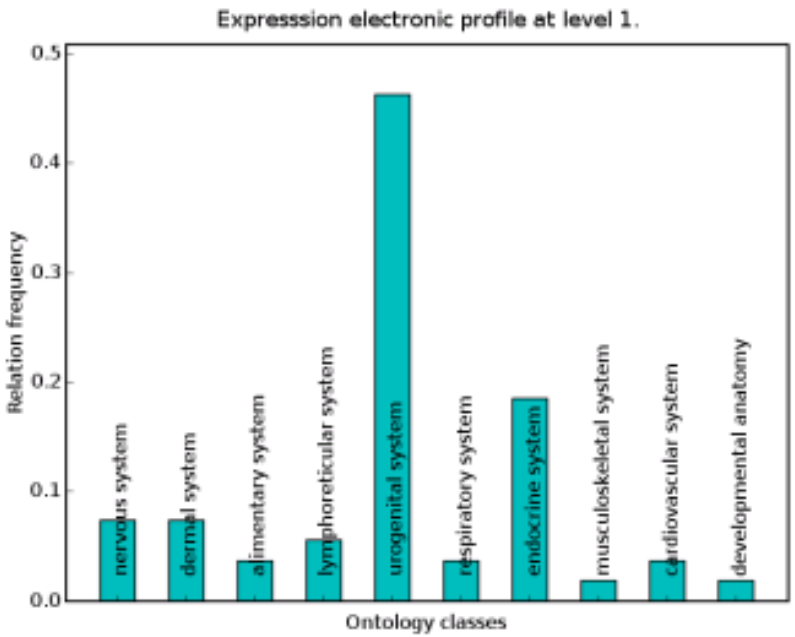


Parcourir tous les niveaux d'un arbre d'ontologie :

- Retrouver les groupes d'une base
  - Retrouver les gènes
  - Trouver ses relations vers les classes d'ontologie
- Sommer les relations
- Calculer la fréquence de liaison brute puis normalisée

Les profils d'expression éléctronique



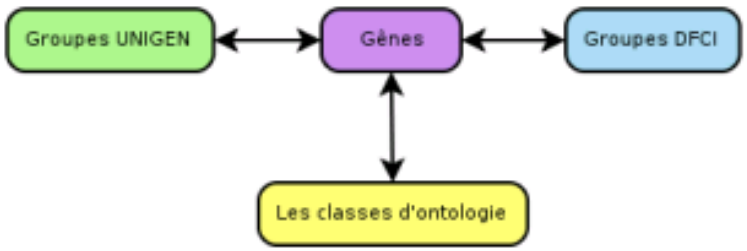


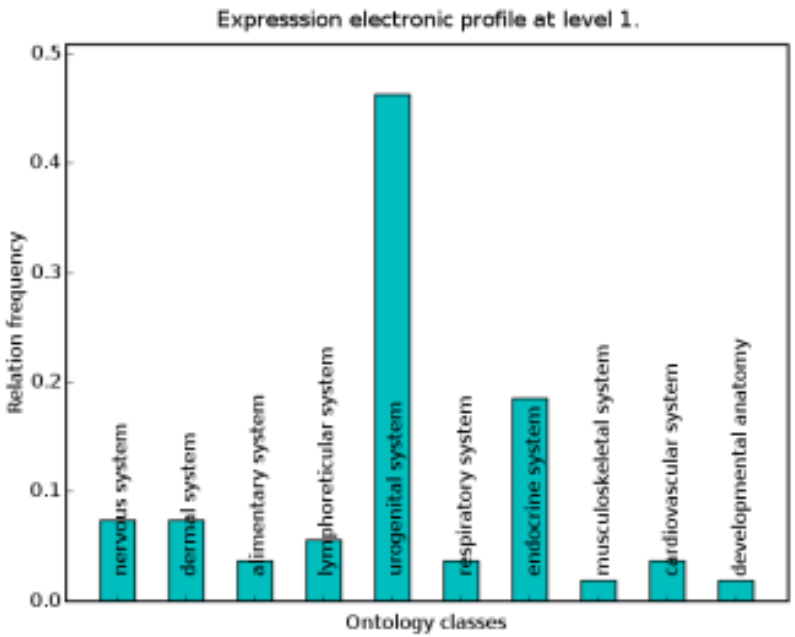
Parcourir tous les niveaux d'un arbre d'ontologie :

- Retrouver les groupes d'une base
  - Retrouver les gènes
  - Trouver ses relations vers les classes d'ontologie
- Sommer les relations
- Calculer la fréquence de liaison brute puis normalisée

A répéter sur tous les arbres d'ontologies

Les profils d'expression éléctronique





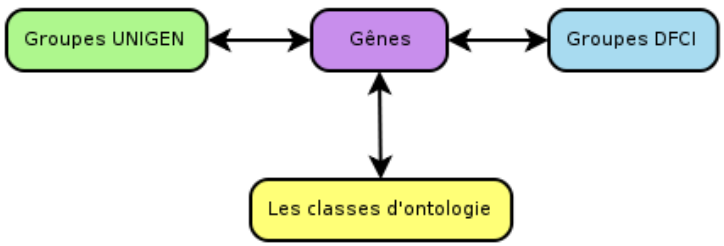
Parcourir tous les niveaux d'un arbre d'ontologie :

- Retrouver les groupes d'une base
  - Retrouver les gènes
  - Trouver ses relations vers les classes d'ontologie
- Sommer les relations
- Calculer la fréquence de liaison brute puis normalisée

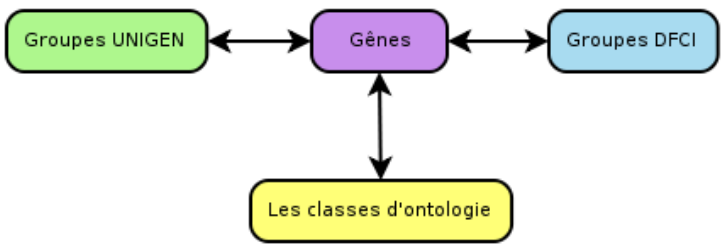
A répéter sur tous les arbres d'ontologies
Besoin de calculer et stocker

Les besoins

Le programme doit pouvoir trouver :



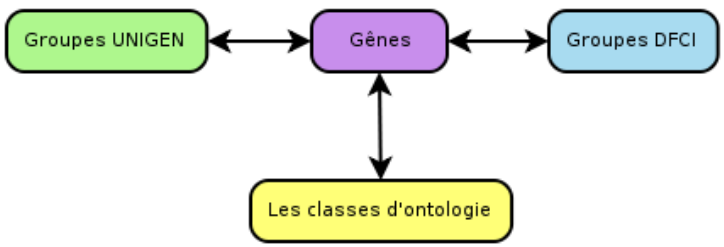
Les besoins



Le programme doit pouvoir trouver :

- les gènes d'un groupe avec une fréquence de liaison donnée

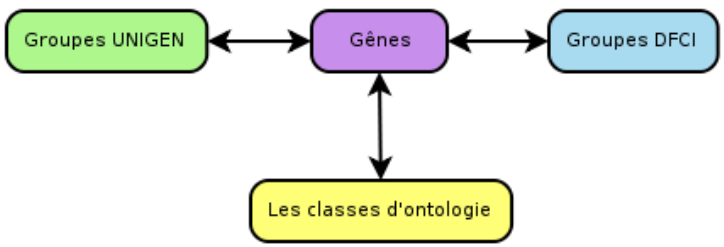
Les besoins



Le programme doit pouvoir trouver :

- les gènes d'un groupe avec une fréquence de liaison donnée
- le profil d'expression d'un groupe

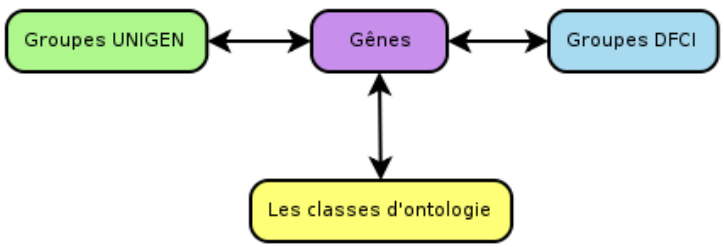
Les besoins



Le programme doit pouvoir trouver :

- les gènes d'un groupe avec une fréquence de liaison donnée
- le profil d'expression d'un groupe
- les gènes correspondant à une classe d'ontologie

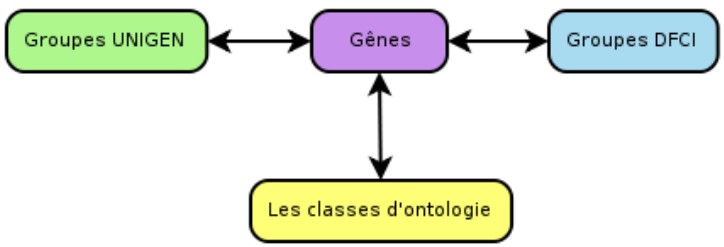
Les besoins



Le programme doit pouvoir trouver :

- les gènes d'un groupe avec une fréquence de liaison donnée
- le profil d'expression d'un groupe
- les gènes correspondant à une classe d'ontologie
- des comparaisons entre les bases UNIGEN et DFCI (indices de cohérence)

Les besoins



Le programme doit pouvoir trouver :

- les gènes d'un groupe avec une fréquence de liaison donnée
- le profil d'expression d'un groupe
- les gènes correspondant à une classe d'ontologie
- des comparaisons entre les bases UNIGEN et DFCI (indices de cohérence)

Contraintes de performance

Parseur des groupes UNIGEN

Fichier texte UNIGEN

```
ID              Hs.2
TITLE           N-acetyltransferase 2
GENE            NAT2
CHROMOSOME      8
SEQUENCE        ACC=BC067218.1; SEQTYPE=mRNA
SEQUENCE        ACC=BG569293.1; TRACE=44157214
```

Schéma du parseur

```
def parse_unigen(filename):
    hs_grp = None
    for line in open(filename):
        if line.startswith("ID"):
            if hs_grp is not None:
                yield hs_grp
            hs_grp = HSGroup()
        field, value = line.split(" ", 1)
        hs_grp.new_field(field, value.strip())
```

Parseur des groupes UNIGEN

Fichier texte UNIGEN

```
ID          Hs.2
TITLE       N-acetyltransferase 2
GENE        NAT2
CHROMOSOME  8
SEQUENCE    ACC=BC067218.1; SEQTYPE=mRNA
SEQUENCE    ACC=BG569293.1; TRACE=44157214
```

Schéma du parseur

```
def parse_unigen(filename):
    hs_grp = None
    for line in open(filename):
        if line.startswith("ID"):
            if hs_grp is not None:
                yield hs_grp
            hs_grp = HSGroup()
        field, value = line.split(" ", 1)
        hs_grp.new_field(field, value.strip())
```

- Tous les parseurs en moins d'une centaine de lignes

Parseur des groupes UNIGEN

Fichier texte UNIGEN

```
ID          Hs.2
TITLE       N-acetyltransferase 2
GENE        NAT2
CHROMOSOME  8
SEQUENCE    ACC=BC067218.1; SEQTYPE=mRNA
SEQUENCE    ACC=BG569293.1; TRACE=44157214
```

Schéma du parseur

```
def parse_unigen(filename):
    hs_grp = None
    for line in open(filename):
        if line.startswith("ID"):
            if hs_grp is not None:
                yield hs_grp
            hs_grp = HSGroup()
        field, value = line.split(" ", 1)
        hs_grp.new_field(field, value.strip())
```

- Tous les parseurs en moins d'une centaine de lignes
- Tests unitaires

Parseur des groupes UNIGEN

Fichier texte UNIGEN

```
ID          Hs.2
TITLE       N-acetyltransferase 2
GENE        NAT2
CHROMOSOME  8
SEQUENCE    ACC=BC067218.1; SEQTYPE=mRNA
SEQUENCE    ACC=BG569293.1; TRACE=44157214
```

Schéma du parseur

```
def parse_unigen(filename):
    hs_grp = None
    for line in open(filename):
        if line.startswith("ID"):
            if hs_grp is not None:
                yield hs_grp
            hs_grp = HSGroup()
        field, value = line.split(" ", 1)
        hs_grp.new_field(field, value.strip())
```

- Tous les parseurs en moins d'une centaine de lignes
- Tests unitaires (utilisation de `pytest`, `logilab.common`)

Parseur des groupes DFCI

Fichiers sources DFCI

```
THC1234567 GB|CN421213 GB|CN421214
THC2527407 GB|BF330358
THC1234567 GB|AB123456 GB|CD123456
```

```
>THC2468584 GB|AAG37073.1 NOTCH2 protein {Homo sapiens}
TCATCTGGAATTATGCCCGCCCTGCGCCCCGCTCTGCTGT
```

Parseur

```
def parse_thc_est(filename):
    thc_dict = {}
    for line in open(filename):
        chunks = line.split()
        thcid = chunks[0]
        try:
            cluster = thc_dict[thcid]
        except KeyError:
            cluster = ThcCluster(thcid)
            thc_dict[thcid] = cluster
        cluster.add_est_codes([est[3:] for est in chunks[1:]
                               if est[:3] == 'GB|'])
    return thc_dict
```

Parseur des ontologies Evoc

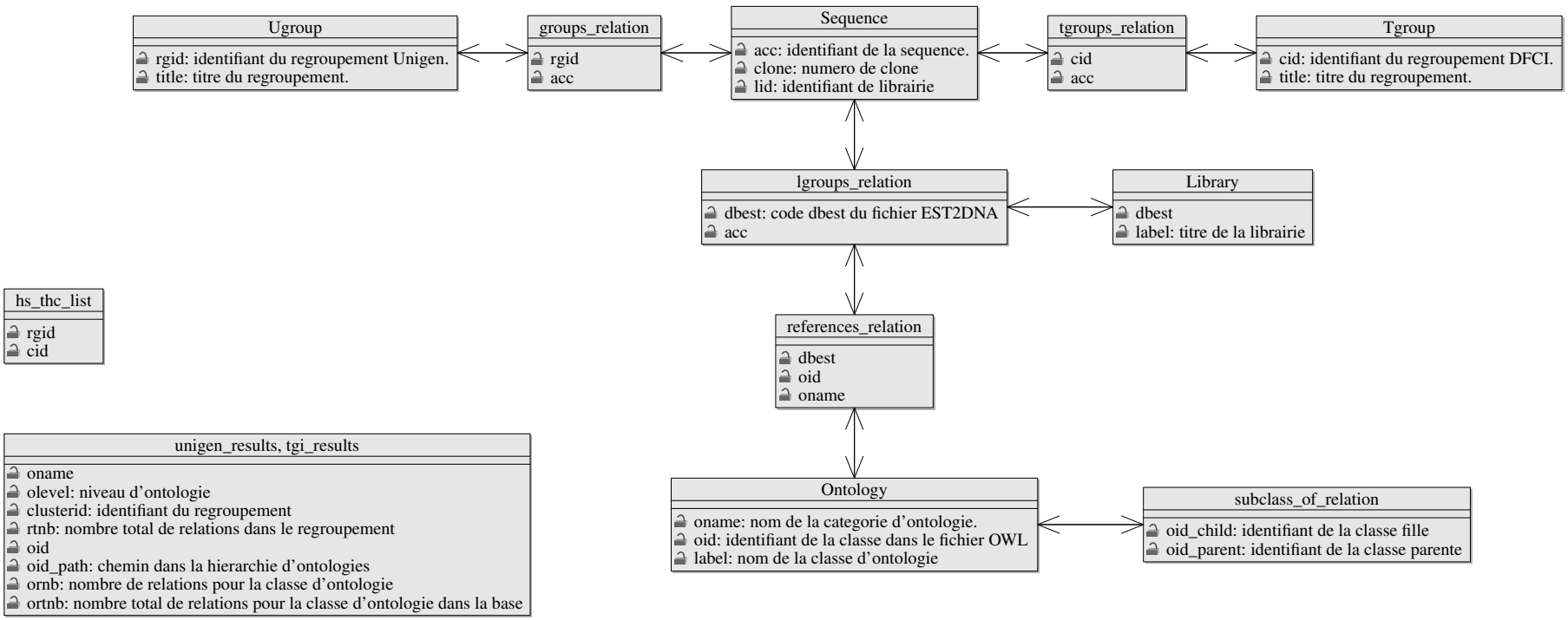
Fichier OWL

```
<owl:Class rdf:ID="EV.0100162">
<rdf:subClassOf rdf:resource="#EV.0100000" />
</owl:Class>
<owl:Class rdf:ID="EV.0100335">
<rdf:subClassOf rdf:resource="#EV.0100162" />
</owl:Class>
```

Parseur :

```
from cElementTree import parse
OWL, RDF = "{http://www.w3.org/owl#}", "{http://www.w3.org/rdf#}"
def parse_classes(filename):
    ontologies = {}
    for node in parse(filename).getiterator(OWL + 'Class'):
        oid = node.get(RDF + 'ID')
        subclass_node = node.find(RDF + 'subClassOf')
        if subclass_node is None:
            parent = None
        else:
            parent_oid = subclass_node.get(RDF + 'resource')[1:]
            parent = ontologies[parent_oid]
        oclass = OntologyClass(oid, parent)
        ontologies[oid] = oclass
    return ontologies
```

Schéma de la base de données SQL



PostgreSQL et psycopg2

Choix techniques :

PostgreSQL et psycopg2

Choix techniques :

- Stockage avec le serveur PostgreSQL (fiable et robuste)

PostgreSQL et psycopg2

Choix techniques :

- Stockage avec le serveur PostgreSQL (fiable et robuste)
- Interface Python avec psycopg2 (léger et rapide)

PostgreSQL et psycopg2

Choix techniques :

- Stockage avec le serveur PostgreSQL (fiable et robuste)
- Interface Python avec psycopg2 (léger et rapide)

Résultats atteints à cette étape :

PostgreSQL et psycopg2

Choix techniques :

- Stockage avec le serveur PostgreSQL (fiable et robuste)
- Interface Python avec psycopg2 (léger et rapide)

Résultats atteints à cette étape :

- Stockage rapide avec COPY (300 000 groupes, 7 millions de gènes)

PostgreSQL et psycopg2

Choix techniques :

- Stockage avec le serveur PostgreSQL (fiable et robuste)
- Interface Python avec psycopg2 (léger et rapide)

Résultats atteints à cette étape :

- Stockage rapide avec COPY (300 000 groupes, 7 millions de gènes)
- Recherche des relations faite par PostgreSQL

PostgreSQL et psycopg2

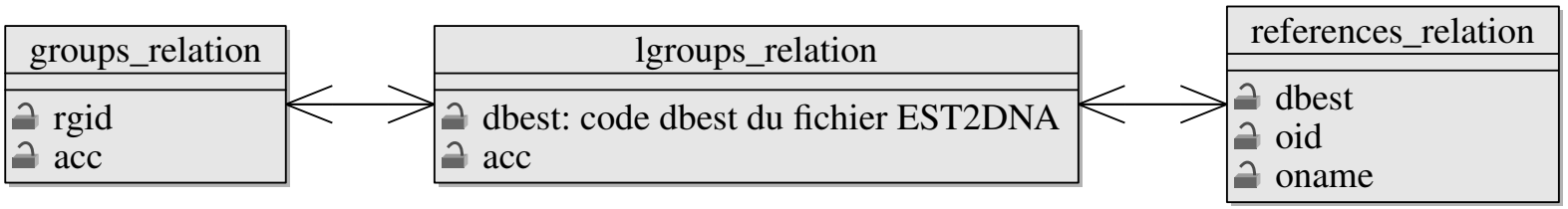
Choix techniques :

- Stockage avec le serveur PostgreSQL (fiable et robuste)
- Interface Python avec psycopg2 (léger et rapide)

Résultats atteints à cette étape :

- Stockage rapide avec COPY (300 000 groupes, 7 millions de gènes)
- Recherche des relations faite par PostgreSQL
- Réalisation de l'interface unifiée, différences des sources d'information gommées par la mise en base des résultats

Example d'utilisation




```
def get_raw_profile(cluster_id, base_key, oview_name, cursor):
    sql_queries = {}
    sql = "SELECT LR.dbest, RR.oid "
    sql += "FROM ((groups_relation as GR "
    sql +=       "NATURAL LEFT OUTER JOIN lgroups_relation as LR) "
    sql +=       "NATURAL LEFT OUTER JOIN %s as RR) " % oview_name
    sql += "WHERE GR.rgid=%(clusterid)s;"
    sql_queries["unigen"] = sql

    cursor.execute(sql_queries[base_key], {"clusterid" : cluster_id})

    return build_raw_profile(cluster_id, cursor.fetchall())

prof = get_raw_profile("Hs.2", "unigen",
                       "anatomical_system_view", cursor)
```

Interface au développement

Construction d'objets Python pour :

Interface au développement

Construction d'objets Python pour :

- abstraire les requêtes SQL

Interface au développement

Construction d'objets Python pour :

- abstraire les requêtes SQL
- contenir les résultats

Interface au développement

Construction d'objets Python pour :

- abstraire les requêtes SQL
- contenir les résultats

Utilitaire en ligne de commandes pour manipuler le module (`gendb`)

Interface au développement

Construction d'objets Python pour :

- abstraire les requêtes SQL
- contenir les résultats

Utilitaire en ligne de commandes pour manipuler le module (`gendb`)

- satisfaisant pour la mise en base

Interface au développement

Construction d'objets Python pour :

- abstraire les requêtes SQL
- contenir les résultats

Utilitaire en ligne de commandes pour manipuler le module (`gendb`)

- satisfaisant pour la mise en base
- limité sur les requêtes possibles

Interface au développement

Construction d'objets Python pour :

- abstraire les requêtes SQL
- contenir les résultats

Utilitaire en ligne de commandes pour manipuler le module (`gendb`)

- satisfaisant pour la mise en base
- limité sur les requêtes possibles
- difficile pour l'exploitation des résultats (fichiers textes)

Interface au développement

Construction d'objets Python pour :

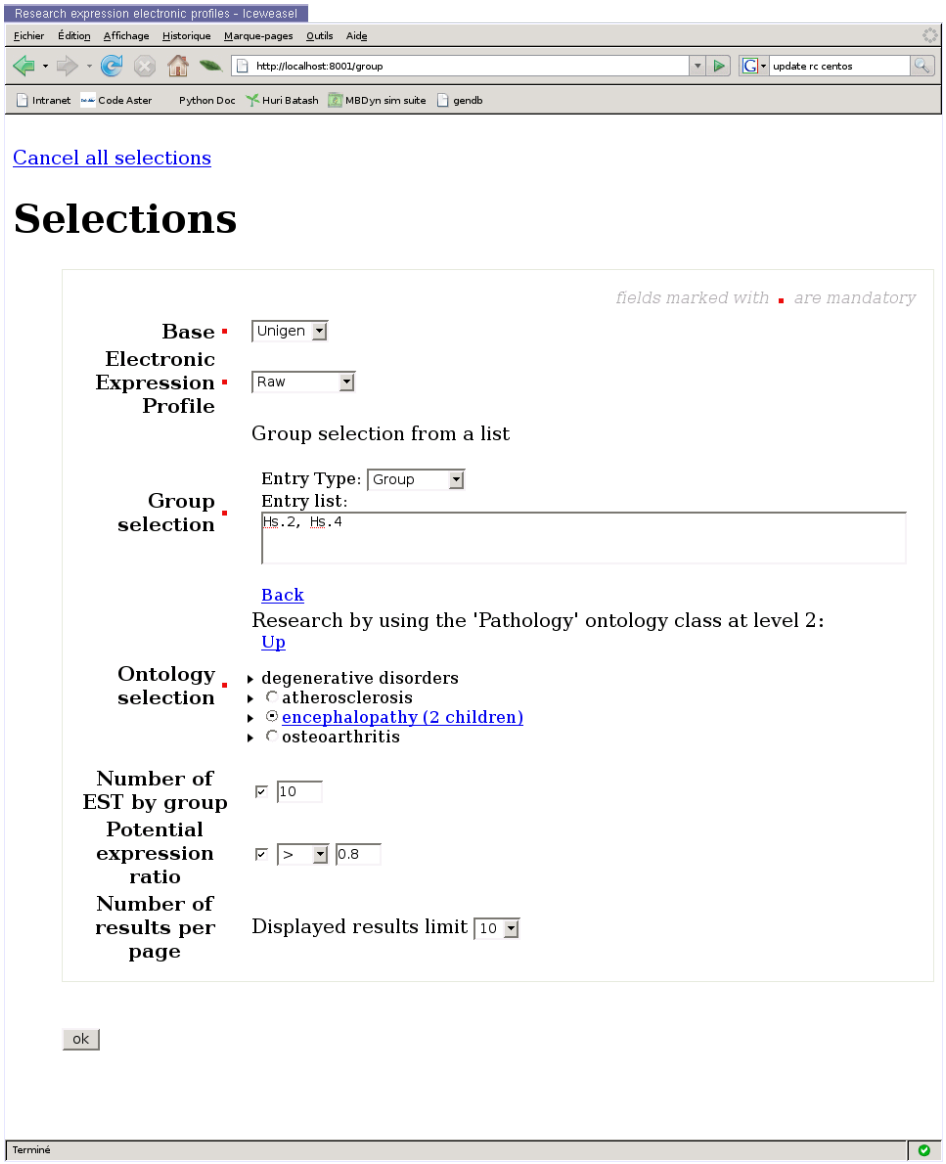
- abstraire les requêtes SQL
- contenir les résultats

Utilitaire en ligne de commandes pour manipuler le module (`gendb`)

- satisfaisant pour la mise en base
- limité sur les requêtes possibles
- difficile pour l'exploitation des résultats (fichiers textes)

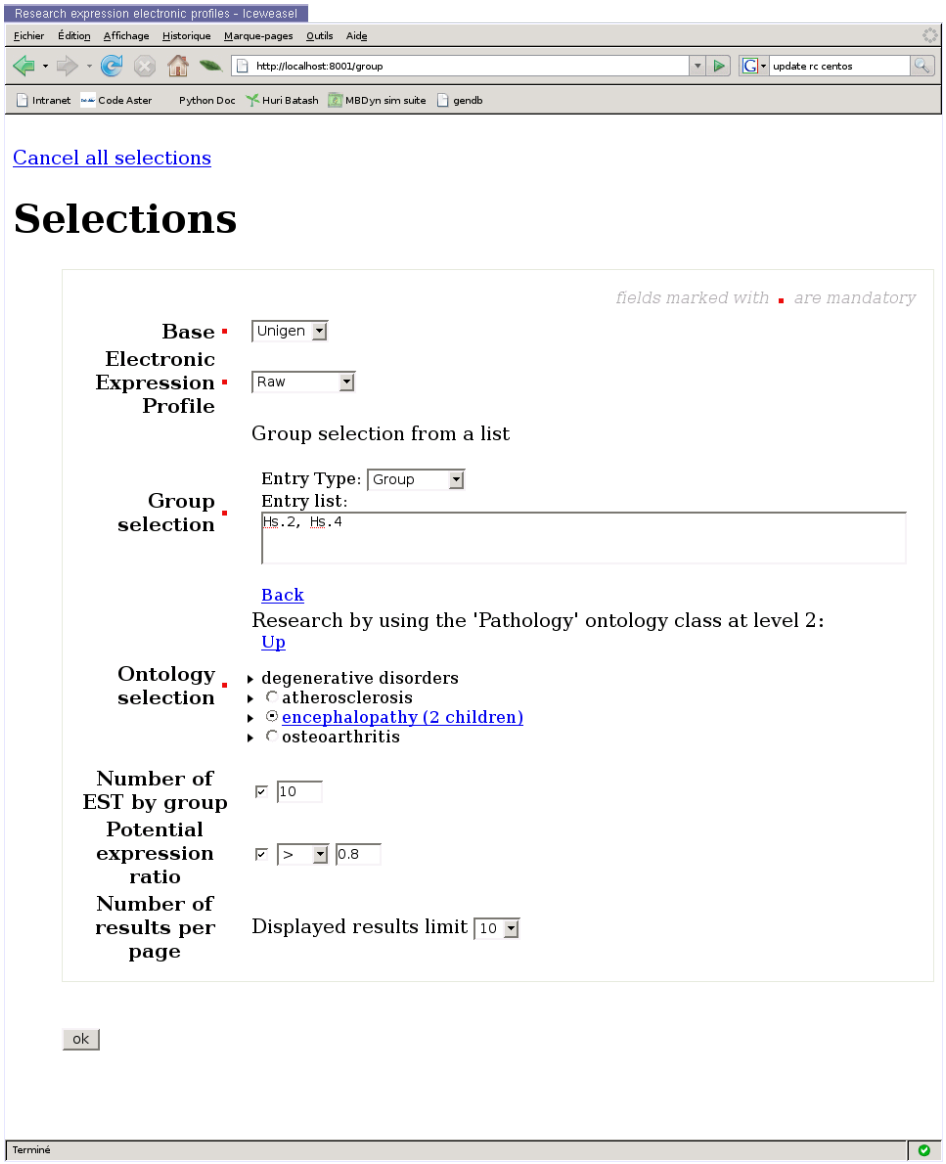
Utilisation doit être plus facile pour les chercheurs

Réalisation d'une interface web



Application web :

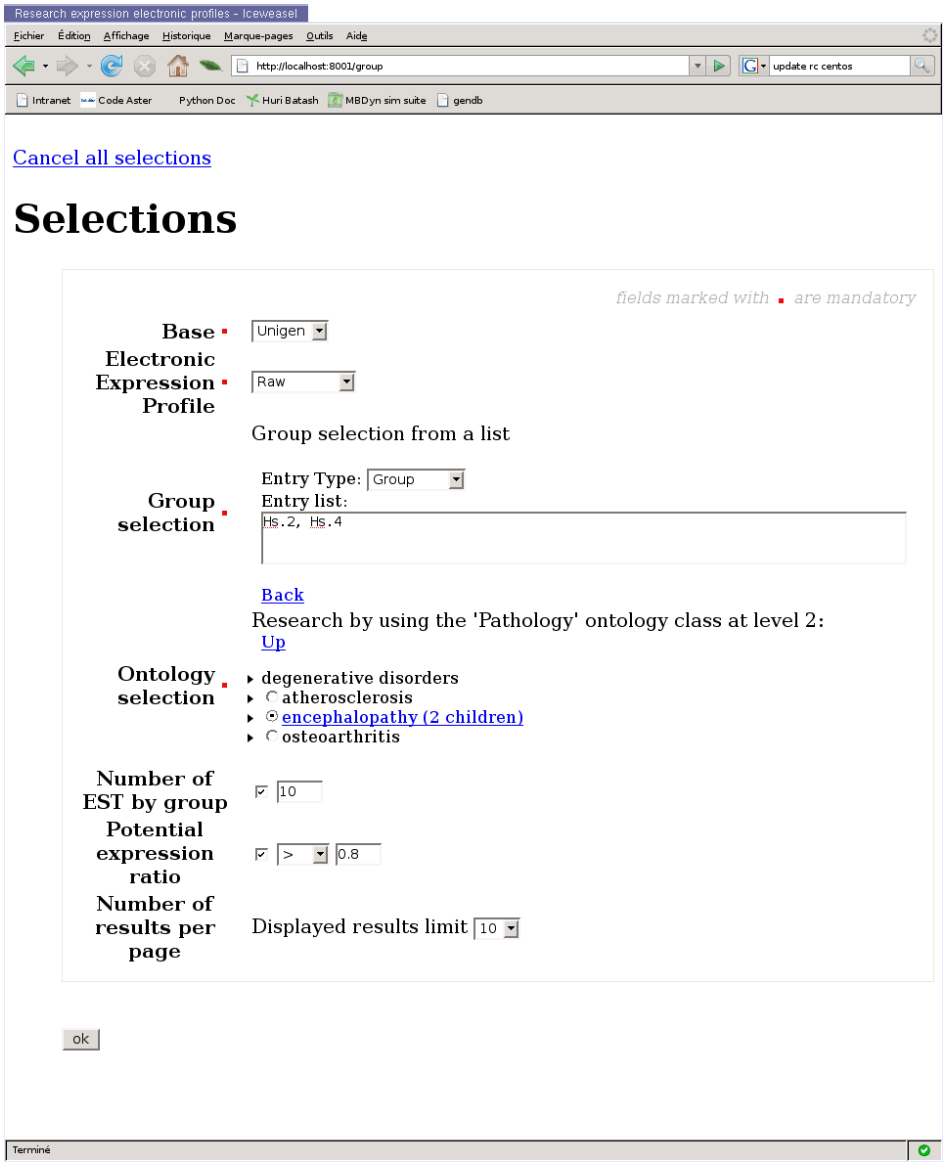
Réalisation d'une interface web



Application web :

- serveur web et gestion des sessions par `Twisted`

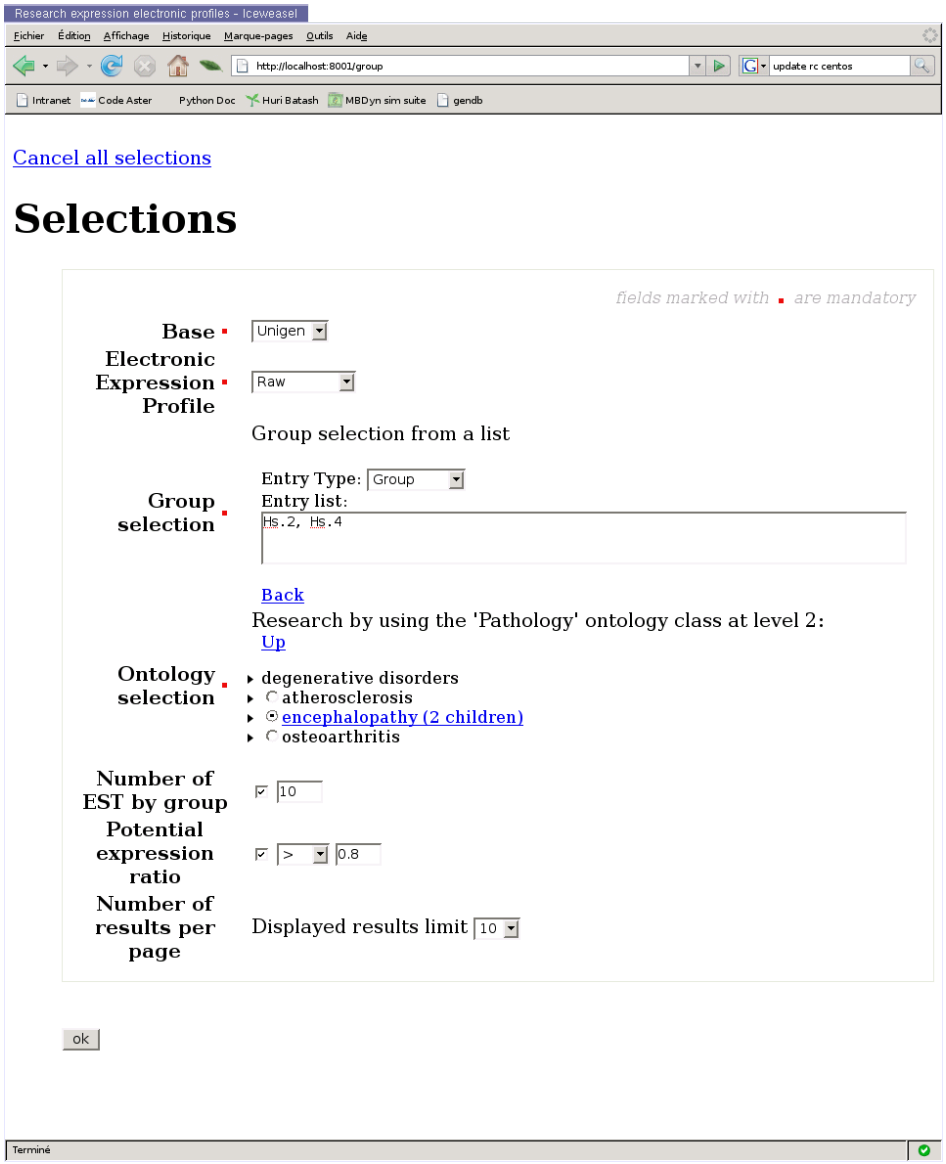
Réalisation d'une interface web



Application web :

- serveur web et gestion des sessions par `Twisted`
- formulaire pour remplir les objets requêtes

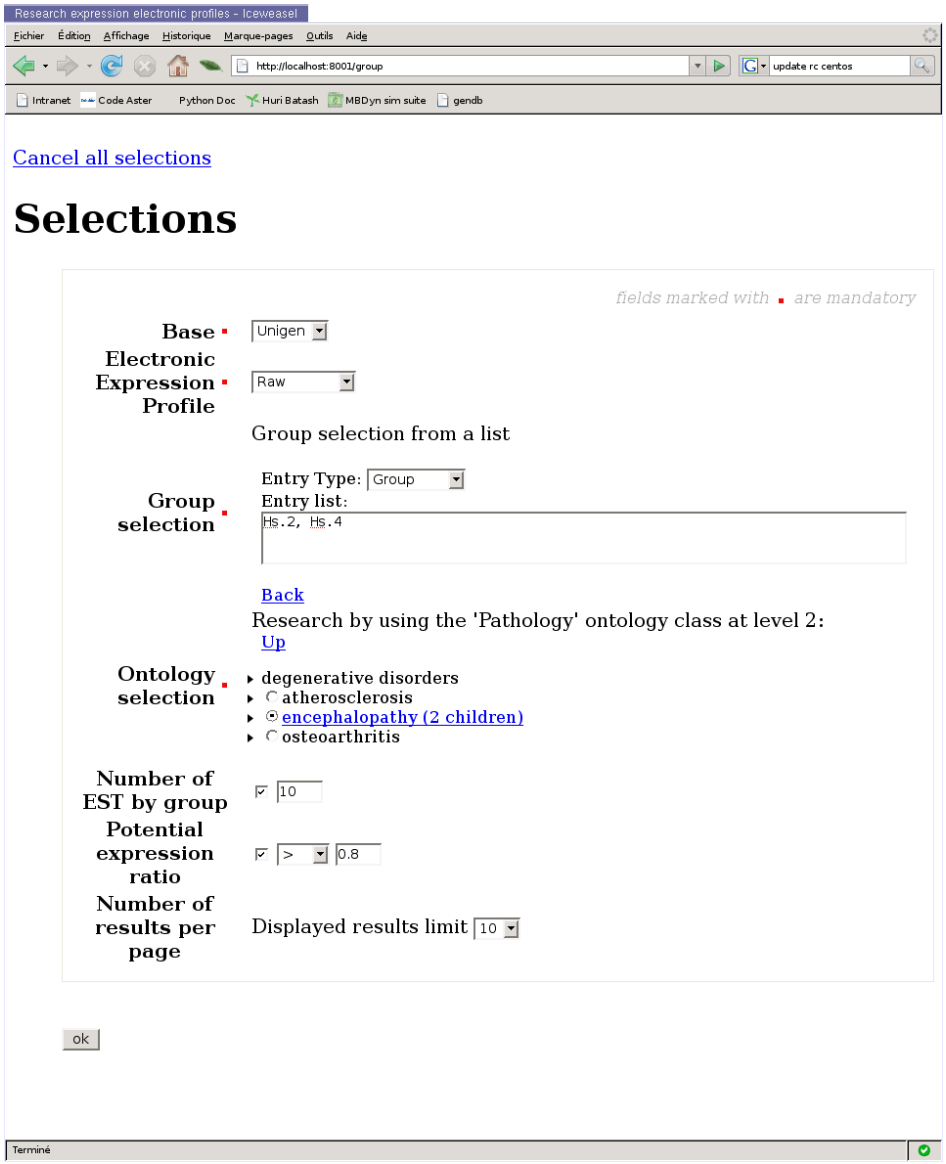
Réalisation d'une interface web



Application web :

- serveur web et gestion des sessions par `Twisted`
- formulaire pour remplir les objets requêtes
- Javascript pour exploration des ontologies (`MochiKit` et `simplejson`)

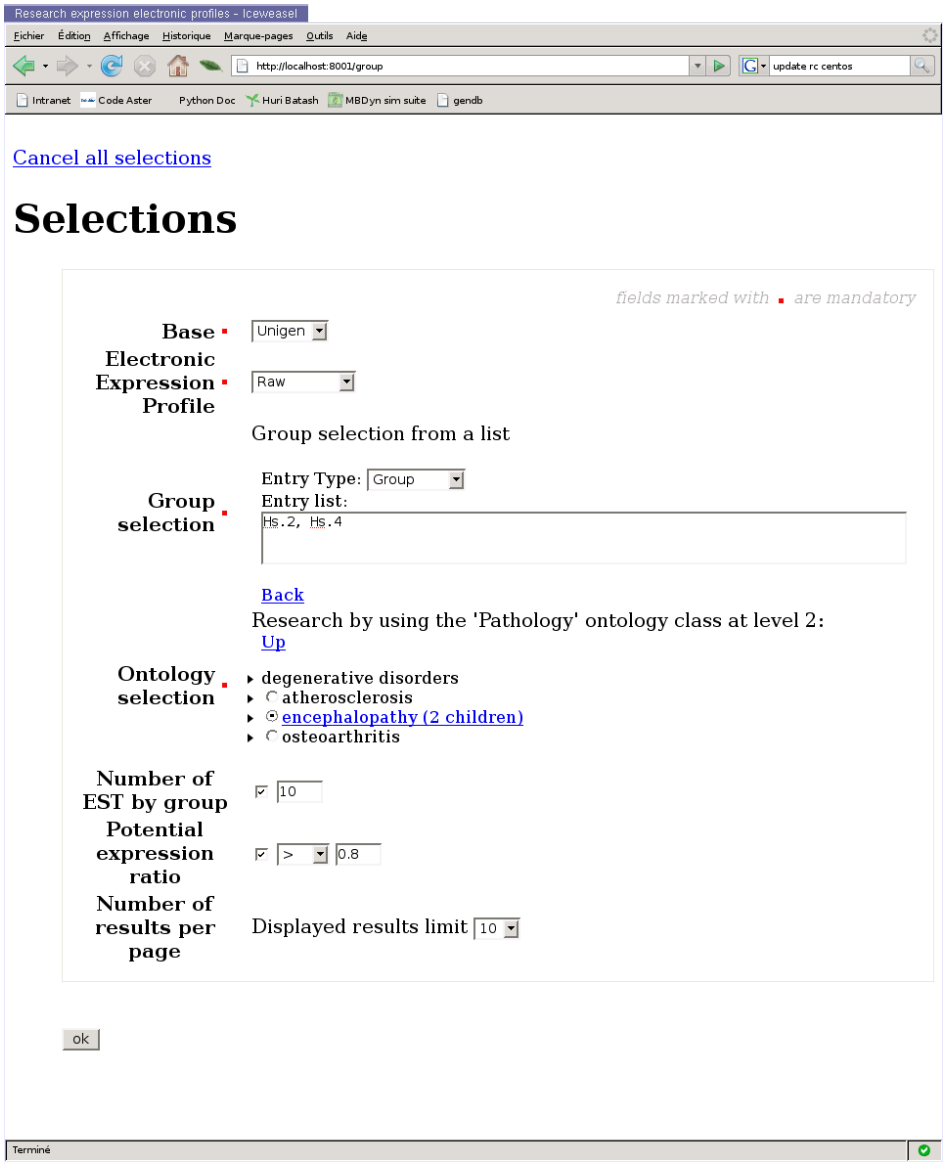
Réalisation d'une interface web



Application web :

- serveur web et gestion des sessions par `Twisted`
- formulaire pour remplir les objets requêtes
- Javascript pour exploration des ontologies (`MochiKit` et `simplejson`)
- résultats en tableaux HTML (histogrammes avec `matplotlib`, fichiers à télécharger)

Réalisation d'une interface web



Application web :

- serveur web et gestion des sessions par `Twisted`
- formulaire pour remplir les objets requêtes
- Javascript pour exploration des ontologies (`MochiKit` et `simplejson`)
- résultats en tableaux HTML (histogrammes avec `matplotlib`, fichiers à télécharger)
- recherches incrémentales

Développement rapide avec Python

Modules utilisés :

Développement rapide avec Python

Modules utilisés :

- `cElementTree` pour parser les ontologies

Développement rapide avec Python

Modules utilisés :

- `cElementTree` pour parser les ontologies
- `psycopg2` pour manipuler le serveur postgreSQL

Développement rapide avec Python

Modules utilisés :

- `cElementTree` pour parser les ontologies
- `psycopg2` pour manipuler le serveur postgreSQL
- `twisted` et `simplejson` pour l'interface web

Développement rapide avec Python

Modules utilisés :

- `cElementTree` pour parser les ontologies
- `psycopg2` pour manipuler le serveur postgreSQL
- `twisted` et `simplejson` pour l'interface web
- `matplotlib` pour les profiles d'expression éléctroniques

Développement rapide avec Python

Modules utilisés :

- `cElementTree` pour parser les ontologies
- `psycopg2` pour manipuler le serveur postgreSQL
- `twisted` et `simplejson` pour l'interface web
- `matplotlib` pour les profiles d'expression éléctroniques
- `logilab.common` et `pytest` pour tester

Développement rapide avec Python

Modules utilisés :

- `cElementTree` pour parser les ontologies
- `psycopg2` pour manipuler le serveur postgreSQL
- `twisted` et `simplejson` pour l'interface web
- `matplotlib` pour les profiles d'expression éléctroniques
- `logilab.common` et `pytest` pour tester

Projet réalisé sur la méthode du "Test-Driven-Development" (TDD) : écriture du test avant le code

Développement rapide avec Python

Modules utilisés :

- `cElementTree` pour parser les ontologies
- `psycopg2` pour manipuler le serveur postgreSQL
- `twisted` et `simplejson` pour l'interface web
- `matplotlib` pour les profiles d'expression éléctroniques
- `logilab.common` et `pytest` pour tester

Projet réalisé sur la méthode du "Test-Driven-Development" (TDD) : écriture du test avant le code
Bonne couverture de tests pour assurer la fiabilité

Conclusion sur l'application

Fonctionnement de `gendb`

Conclusion sur l'application

Fonctionnement de `gendb`

1. Chargement des données biologiques en base SQL

Conclusion sur l'application

Fonctionnement de `gendb`

1. Chargement des données biologiques en base SQL
2. Lancement des calculs et stockages (interface unifiée)

Conclusion sur l'application

Fonctionnement de `gendb`

1. Chargement des données biologiques en base SQL
2. Lancement des calculs et stockages (interface unifiée)
3. Exploration des résultats par un module Python et une interface web

Conclusion sur l'application

Fonctionnement de `gendb`

1. Chargement des données biologiques en base SQL
2. Lancement des calculs et stockages (interface unifiée)
3. Exploration des résultats par un module Python et une interface web

Futur

Conclusion sur l'application

Fonctionnement de `gendb`

1. Chargement des données biologiques en base SQL
2. Lancement des calculs et stockages (interface unifiée)
3. Exploration des résultats par un module Python et une interface web

Futur

1. Performance à améliorer sur la base et le serveur (temps de réaction, mémoire)

Conclusion sur l'application

Fonctionnement de `gendb`

1. Chargement des données biologiques en base SQL
2. Lancement des calculs et stockages (interface unifiée)
3. Exploration des résultats par un module Python et une interface web

Futur

1. Performance à améliorer sur la base et le serveur (temps de réaction, mémoire)
2. Développer de nouvelles possibilités de recherches (profils d'expression, indices de cohérence)

Conclusion sur l'application

Fonctionnement de `gendb`

1. Chargement des données biologiques en base SQL
2. Lancement des calculs et stockages (interface unifiée)
3. Exploration des résultats par un module Python et une interface web

Futur

1. Performance à améliorer sur la base et le serveur (temps de réaction, mémoire)
2. Développer de nouvelles possibilités de recherches (profils d'expression, indices de cohérence)
3. Ajouter la possibilité de statistiques