

M2 DataScale

TP - Architectures Orientées Service

Spécification du Service Web Composite : Evaluation de
Demande de Prêt Immobilier

Thivagini SUGUMAR, Marine CONOR

3 décembre 2023

Table Des Matières

1 Notes sur le projet	4
2 Contexte général du TP	4
3 Objectif	4
4 Fonctionnement du système	4
4.1 Architecture	4
4.2 Étapes	5
5 Identification des services	6
6 Modélisation des services sous SOAP	7
6.1 Modélisation	7
6.2 Fonctionnement interne des services	7
7 Implémentation avec SOAP	10
7.1 Structure du projet	10
7.2 Spécifications des services	11
7.2.1 Service Composite	11
7.2.2 Service Extraction	11
7.2.3 Service Solvabilité	12
7.2.4 Service Évaluation	12
7.2.5 Service Décision	13
7.3 Spécificités de SOAP	13
7.3.1 Spyne Application	13
7.3.2 WSGI Application et WSGI Mounter	14
7.3.3 Interaction entre les services sous le protocole SOAP	14
8 Interface	17
8.1 Pages web	17
8.1.1 Accueil : page d'accueil	17
8.1.2 Formulaire : page contenant le formulaire de demande de prêt à évaluer	17
8.1.3 Confirmation : page de confirmation de la soumission du formulaire	18
8.1.4 Connexion : page de connexion pour consulter les résultats	18
8.1.5 Disponible : page affichant que le résultat est disponible	19
8.1.6 Introuvable : page affichant que le résultat est introuvable	19
8.1.7 Traitement : page affichant que le résultat n'est pas encore disponible	20
8.2 Lancement de l'application	20
8.2.1 Commandes	20
8.2.2 Côté Client : Web	20
8.2.3 Côté Serveur : Terminal	23
9 Démonstration avec SOAP	25
9.1 Scénario 1 : Demande de prêt acceptée	25
9.1.1 Saisie dans le formulaire	25
9.1.2 Résultat attendu	25
9.1.3 Page résultat attendue	25
9.2 Scénario 2 : Demande de prêt refusée	26
9.2.1 Saisie dans le formulaire	26
9.2.2 Résultat attendu	26
9.2.3 Page résultat attendue	26
10 Modélisation des services sous REST	27
10.1 Modélisation	27
10.2 Différences par rapport au SOAP	27

11 Implémentation avec REST	28
11.1 Structure du projet	28
11.2 Spécifications des services	29
11.2.1 Service Composite	29
11.2.2 Service Extraction	29
11.2.3 Service VerificationSolvabilite	29
11.2.4 Service EvaluationPropriete	30
11.2.5 Service DecisionApprobation	30
11.2.6 App	31
11.3 Interaction entre les services sous l'API REST	31
12 Démonstration avec REST	34
12.1 Scénario 1 : Profil parfait	34
12.1.1 Saisie dans le formulaire	34
12.1.2 Résultat attendu	34
12.1.3 Page résultat attendu	34
12.2 Scénario 2 : Profil emploi instable	35
12.2.1 Saisie dans le formulaire	35
12.2.2 Résultat attendu	35
12.2.3 Page résultat attendu	35
12.3 Scénario 3 : Profil score trop faible	36
12.3.1 Saisie dans le formulaire	36
12.3.2 Résultat attendu	36
12.3.3 Page résultat attendu	36
12.4 Affichage côté Serveur	37
12.4.1 Scénario 1 : Profil parfait	37
12.4.2 Scénario 2 : Profil emploi instable	37
12.4.3 Scénario 3 : Profil score trop faible	38
12.5 Bases de données evaluationPret.db	39
12.5.1 Table DEMANDE : formulaires saisis par les 3 clients des scénarios précédents	39
12.5.2 Table EVALUATION : fin de l'évaluation des prêts pour les 3 clients	39
12.5.3 Table RESULTAT : résultats des 3 clients	39

1 Notes sur le projet

Veuillez trouver les dernières versions du projet juste ici :

- **SOAP** : <https://github.com/uvsq21704755/ProjetSOAP.git>
- **REST** : <https://github.com/uvsq21704755/ProjetREST.git>

2 Contexte général du TP

Le service Web Composite d'Évaluation de Demande de Prêt Immobilier est conçu pour automatiser le processus d'évaluation des demandes de prêt immobilier en utilisant des services Web spécialisés. Il permet aux clients de soumettre des demandes de prêt immobilier exprimées en langage naturel. Le service intègre des composants d'extraction des informations métiers de texte de la demande, de vérification de solvabilité, d'évaluation de la propriété et de décision d'approbation pour fournir une évaluation complète et précise des demandes de prêt.

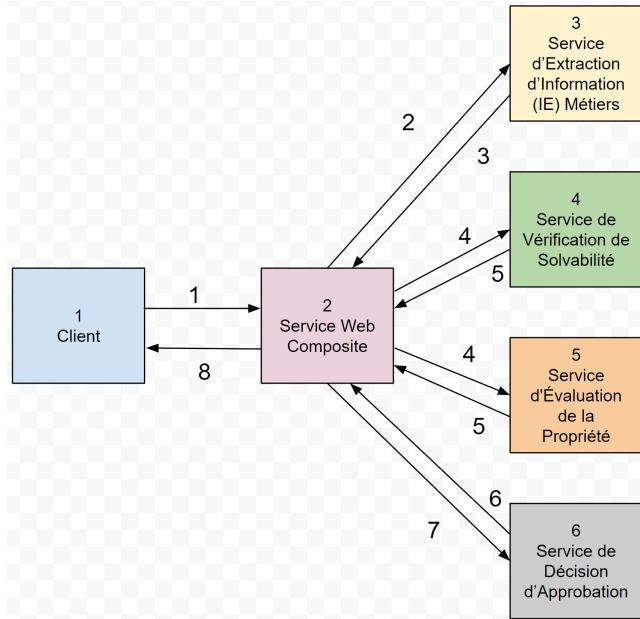
3 Objectif

L'objectif de ce service est de fournir une interface unique pour les clients qui souhaitent demander un prêt immobilier. Le service coordonne les différents services Web nécessaires pour évaluer la demande du client, garantissant ainsi un processus fluide et automatisé.

4 Fonctionnement du système

4.1 Architecture

Nous représentons l'architecture du fonctionnement du système par le diagramme suivant :



4.2 Étapes

- Sur l'interface, le client va effectuer une demande de prêt sur l'interface en soumettant un formulaire. Cette action va enclencher la création d'un fichier `txt` dans le répertoire `demandeTxt`
- Le service **Composite** va détecter l'arrivée du fichier `txt` dans le répertoire `demandeTxt` grâce à une routine qui se lance automatiquement. Ainsi le service composite va récupérer ce fichier `txt`
- Le service composite va effectuer une requête au service **Extraction** afin de récupérer le fichier WSDL
- Le service composite va alors transmettre le fichier `txt` au service d'**Extraction** via sa méthode web service encapsulé dans une enveloppe SOAP
- Le service d'**Extraction** va effectuer le prétraitement, identifier des entités et extraire les entités avec les expressions régulières. Cette extraction sera sauvegardée dans un fichier XML dans `demandeXML` sous le format : `numeroDossier.xml`
- Le service d'**Extraction** renvoie ce fichier XML obtenu au service **Composite**
- Le service **Composite** donne le fichier XML au service de **Vérification de Solvabilité**
- Le service de **Vérification de Solvabilité** va utiliser une fonction de scoring pour donner un score. Il ajoute les balises `<scoring></scoring>` et `<decisionScoring></decisionScoring>` dans le fichier XML
- Le service de **Vérification de Solvabilité** renvoie le fichier XML modifié au service **Composite**
- Le service **Composite** donne le fichier XML modifié au service d'**Évaluation de la Propriété**
- Le service d'**Évaluation de la Propriété** va utiliser une fonction pour évaluer le marché immobilier afin de donner une moyenne et évaluer les normes légales et réglementaires du bâtiment.
Il peut aussi décider de faire une visite virtuelle et une visite sur place.
Il ajoute les balises :
`<estimation_valeur></estimation_valeur>` et
`<decisionConformite></decisionConformite>`, si la décision est "Non admissible".
Il ajoute également aussi une balise `<raisons></raisons>` dans le fichier XML donné en entrée
- Le service d'**Évaluation de la Propriété** renvoie le fichier XML modifié au service **Composite**
- Le service **Composite** donne le fichier XML modifié au service de **Décision d'Approbation**
- Le service de **Décision d'Approbation** va utiliser une fonction comparant les résultats et analyses précédents. Il va ensuite créer un nouveau fichier `txt` dans le dossier `reponseTxt` sous le format `numeroDossier.txt`
- Le service de **Décision d'Approbation** renvoie le fichier `txt` au service **Composite**
- Le client peut retrouver sa réponse sur l'interface en consultant les résultats

5 Identification des services

Les services identifiés sont les suivants :

- Le service **Composite** :

Permet l'interaction entre le Client et les différents Services. Il s'occupe notamment de la réception de la demande, l'extraction d'information métier, la vérification de solvabilité, l'évaluation de la propriété et la décision d'approbation.

- Le service d'**Extraction** :

Reçoit le texte de la demande de prêt soumise par le client, via le service **Composite**. Il s'occupe principalement du prétraitement, analyse, identification des entités et l'extraction de ces informations. De plus, il permet le stockage des informations extraites dans la base de données du service **Composite**

- Le service de **Vérification de Solvabilité** :

Permet d'évaluer la capacité financière du client à rembourser le prêt. Il s'occupe donc de l'intégration des bureaux de crédit pour se connecter à leurs bureaux qui stockent l'historique financier des clients. Il récupère des informations telles que les dettes en cours, les paiements en retard et les antécédents de faillite pour évaluer la crédibilité du client. Ensuite, le service utilise des algorithmes de scoring de crédit pour attribuer un score au client en fonction de son historique financier. Ce score représente le niveau de risque associé à l'accord d'un prêt au client. Les clients avec des scores élevés ont une solvabilité plus élevée. Il analyse aussi les revenus et les dépenses mensuelles.

- Le service d'**Évaluation de la Propriété** :

Permet d'estimer la valeur marchande de la propriété pour laquelle le prêt est demandé. Il peut utiliser des données immobilières, des expertises locales et des critères légaux pour effectuer cette évaluation. Il s'occupe donc d'analyser les données du marché immobilier, peut effectuer une inspection virtuelle de la propriété ou même des visites sur place et vérifier si la propriété est conforme aux normes légales et réglementaires en vigueur.

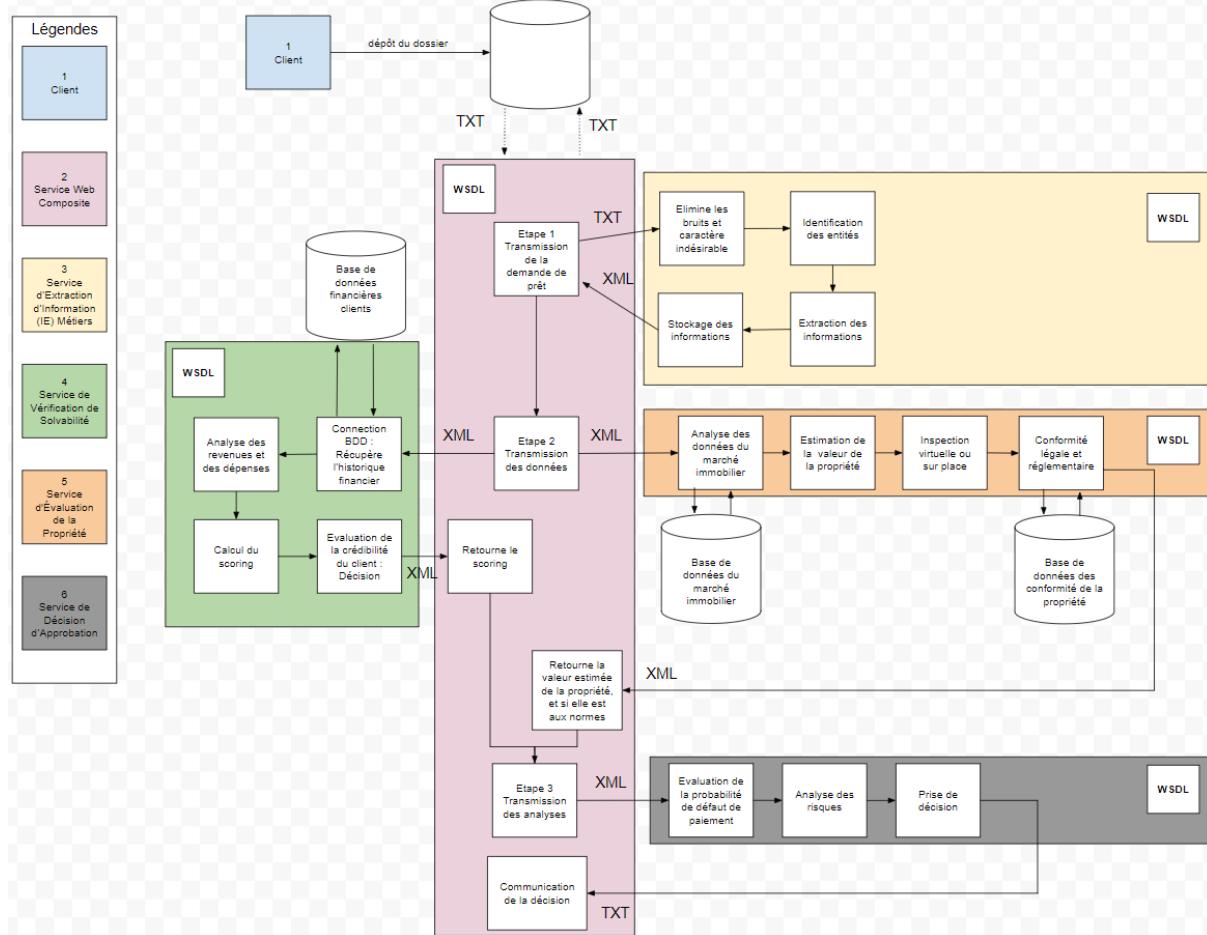
- Le service de **Décision d'Approbation** :

Analyse les données recueillies lors des étapes précédentes (Vérification de Solvabilité et Évaluation de la Propriété) pour déterminer si le prêt immobilier peut être approuvé. Il s'occupe d'analyser les risques associés à l'accord du prêt, de comparer les données de la demande de prêt avec les politiques internes de l'institution financière pour évaluer la probabilité de défaut de paiement du client. En fonction des résultats, le service prend une décision d'approbation ou de refus du prêt puis communique celle-ci au service **Composite** (en indiquant les raisons de l'approbation ou du refus du prêt) qui va retourner cette réponse au client.

6 Modélisation des services sous SOAP

6.1 Modélisation

Voici une architecture à base de service pour mettre en œuvre le processus d'évaluation de demande de prêt immobilier :



6.2 Fonctionnement interne des services

Le service **Composite** permet les échanges entre les services et les échanges avec le client et organise le SOAP.

Le service d'**Extraction** prend en paramètre le fichier **txt** de la demande du client.

Ce service va récupérer les informations indiquées dans le fichier **txt**, supprime les caractères indésirables et extrait les données après le caractère " : ".

Une fois extraites, les données sont écrites dans des balises correspondantes dans un fichier **XML**.

Le fichier **XML** sera nommé selon le numéro de dossier attribué au client.

Le service de **Vérification de Solvabilité** prend en paramètre le fichier **XML** concernant la demande du client.

Ce service va récupérer les informations concernant le montant et la durée du prêt ainsi que les revenus et les dépenses mensuelles à l'aide du numéro de dossier du client.

Il va également récupérer le fichier **json** banque :

(**idBanque**, **age**, **enfants**, **emploi**, **nbCreditsEnCours**, **antecedents**, **tauxEndettement**) où :

- **idBanque** est l'identifiant de la banque du client
- **age** est l'âge du client
- **enfants** est le nombre d'enfants à sa charge
- **emploi** est un booléen avec 0 si l'emploi n'est pas stable, 1 sinon

- **nbCreditsEnCours** est le nombre de crédits totaux toujours en cours
- **antecedents** est le nombre d'antécédents à son actif (dettes non payées, retards...)
- **tauxEndettement** est le taux d'endettement actuel

Le service va ensuite extraire les données en prenant le tuple correspondant à l'**idBanque** du fichier XML et faire un scoring en fonction des informations.

La décision sera immédiatement "Non Admissible" si le client a moins de 18 ans ou si sa capacité d'emprunt ajouté à ses dépenses est supérieure à son revenu ou encore si son taux d'endettement est supérieur à 33%.

Une fois l'algorithme de scoring effectué, le service va déterminer en fonction du score, une décision :

- entre 0 et 10 : peu probable
- entre 10 et 20 : à défendre
- entre 20 et 30 : sous conditions
- entre 30 et 40 : très favorable

Finalement, le service ajoute au fichier XML reçu, les balises **<scoring>Valeur du score</scoring>** et **<decisionScoring>Decision en fonction du score</decisionScoring>**.

Le service d'**Évaluation de la Propriété** prend en paramètre le fichier XML selon la demande du client.

Le service vérifie si la propriété est conforme aux normes légales et réglementaires en vigueur.

Il va pour cela, récupérer l'adresse de la propriété puis récupérer le fichier json immobilier : (**idImmobilier, adresse, age, normeLegal, normeReglementaire, litigesEnCours, normeElectricite, normeGaz**) où :

- **idImmobilier** est l'identifiant de la propriété
- **adresse** est l'adresse de la propriété
- **age** est l'âge de la propriété
- **normeLegal** est un booléen avec 0 si la propriété est aux normes, 1 sinon
- **normeReglementaire** est un booléen avec 0 si la propriété est aux normes, 1 sinon
- **litigesEnCours** est le nombre de litiges en cours
- **normeElectricite** est la norme électrique ("NFC 15-100", "NF C 15-100" ou "C 15-100")
- **normeGaz** est la norme pour le gaz ('NF P 45-500' ou vide s'il n'y a pas de gaz dans la propriété)

Le service va ensuite extraire les données en prenant le tuple correspondant à l'adresse du fichier XML et vérifier que tout est conforme : soit 0, soit dans les normes définies.

Si l'un des attributs n'est pas aux normes, la décision est "Non Admissible à un prêt" et liste les raisons pour lesquelles la propriété pose problème.

Il ajoute au fichier XML reçu la balise :

```
<decisionConformite>
Decision en fonction du respect ou non des normes
</decisionConformite>
```

Si la décision est "Non admissible", il ajoute aussi une balise :

```
<raisons>Liste des raisons du refus</raisons>
```

Le service peut demander à faire une visite virtuelle ou sur place, on considère que si le service demande une visite, il effectue d'abord une visite virtuelle, si celle-ci n'est pas concluante, il demande alors un expert pour une visite sur place.

Le service va choisir un entier aléatoire entre 0 et 1 pour savoir s'il fait une visite.

Si c'est 0, il ne fait aucune demande de visite, sinon il enclenche une visite virtuelle.

Par la suite, on suppose qu'il va faire une visite sur place uniquement si l'âge de la propriété est inférieur à 10 ans (il n'est plus nécessaire d'avoir un certificat de conformité si la propriété a plus de 10 ans). En fonction des résultats précédents vis-à-vis de la conformité, la visite sur place est déterminée concluante ou non.

Le service va aussi estimer la valeur marchande de la propriété pour laquelle le prêt est demandé. Il va récupérer le code postal de l'adresse puis récupérer le type de bâtiment (maison/appartement) et, s'il est donné, le nombre d'étages, avec la **descriptionPropriete** du fichier XML.

Ensuite il récupère le fichier json **marcheImmobilier** :

(**adresse**, **codePostal**, **batiment**, **nbEtage**, **valeur**) où :

- **adresse** est l'adresse de la propriété
- **codePostal** est le code postal
- **batiment** est le type de bâtiment
- **nbEtage** est le nombre d'étages
- **valeur** est la valeur marchande de la propriété

Le service va récupérer tous les tuples qui correspondent au même code postal que celui du fichier XML ainsi que le même type de bâtiment, le nombre d'étages s'il est donné et faire une moyenne de leur valeur. Finalement, il ajoute au fichier XML reçu, la balise :

```
<estimation_valeur>  
Valeur moyenne des bâtiments du même type et dans le même secteur  
</estimation_valeur>
```

Le service de **Décision d'Approbation** prend en paramètre le fichier XML concernant la demande du client.

Ce service va récupérer les informations concernant le montant du prêt, le score, la décision du score, la décision de conformité (les raisons pour lesquelles c'est un refus si c'en est un) ainsi que l'estimation de la valeur.

- Si la valeur du score est -1, le score n'a pas été calculé à cause de sa situation financière, le prêt est refusé.
- Si la décision de conformité est "Non admissible à un prêt immobilier", le service émet le refus du prêt avec une liste de raisons.
- Si le montant est supérieur à l'estimation de la valeur en plus d'une valeur choisie arbitrairement, le service émet un refus avec les raisons que le montant demandé est trop important pour la propriété.

Par la suite en fonction de la décision du score, s'il s'agit de "Très favorable", "Sous conditions" ou "À défendre" et que la décision de conformité est aussi "Admissible à un prêt immobilier" alors le service accepte la demande, sinon il la refuse.

En fonction des résultats précédents, le service a pris une décision d'approbation ou de refus du prêt puis communique celle-ci au service **Composite** (en indiquant les raisons du refus du prêt) sous la forme d'un fichier **txt** qui sera stocké dans le répertoire **reponseTxt**.

7 Implémentation avec SOAP

7.1 Structure du projet

```
└── EvaluationDemandePretImmobilier
    └── services
        ├── bdd
        │   ├── banque.json
        │   ├── immobilier.json
        │   └── marcheImmobilier.json
        ├── demandeTxt
        │   ├── 188190.txt
        │   └── 369789.txt
        ├── demandeXml
        │   ├── 188190.xml
        │   └── 369789.xml
        ├── modeles
        │   ├── acceptation.txt
        │   └── refus.txt
        ├── reponseTxt
        │   ├── 188190.txt
        │   └── 369789.txt
        ├── serviceComposite.py
        ├── serviceDecision.py
        ├── serviceEvaluation.py
        ├── serviceExtraction.py
        ├── serviceSolvabilite.py
        └── soap
            ├── decision.xml
            ├── evaluation.xml
            ├── extraction.xml
            └── solvabilite.xml
        └── static
            ├── css
            │   ├── accueil.css
            │   ├── confirmation.css
            │   ├── connexion.css
            │   ├── disponible.css
            │   ├── formulaire.css
            │   ├── introuvable.css
            │   └── traitement.css
            └── js
                └── soap.js
        └── templates
            ├── accueil.html
            ├── confirmation.html
            ├── connexion.html
            ├── disponible.html
            ├── formulaire.html
            ├── introuvable.html
            └── traitement.html
    └── README.md
    └── archive
    └── clean
    └── configure
    └── run
```

- Dans le dossier ProjetSOAP, nous avons:
 - un dossier `EvaluationDemandePretImmobilier`, contenant le coeur du projet;
 - un `README.md`, donnant les commandes et les scénarios à exécuter;
 - et des scripts (`archive`, `clean`, `configure`, `run`).
- Dans le dossier `EvaluationDemandePretImmobilier`, nous avons un dossier `services` contenant le coeur du projet.
- Le projet se situant dans le dossier `services`, il est composé de:
 - un dossier `bdd` contenant l'ensemble des bases de données (`banque`, `immobilier`, `marcheImmobilier`) en format `json`;
 - un dossier `demandeTxt` contenant les formulaires à traiter, soumis par les clients sur l'interface web, en format `txt`;
 - un dossier `demandeXml` contenant les demandes en cours d'évaluation, créée dans l'extraction des données et utilisés par les autres services, en format `XML`;
 - un dossier `modeles` contenant les modèles de textes pour générer le résultat, 2 versions : acceptation et refus;
 - un dossier `reponseTxt` contenant les résultats obtenus suite à l'évaluation du prêt;
 - un dossier `soap` qui contient les messages SOAP envoyés en interaction entre 2 services sous format `XML` (ex : `extraction.xml` qui est le message de communication entre le service composite et le service extraction);
 - un dossier `static` qui contient l'ensemble des fichiers CSS et JS nécessaire pour l'interface web;
 - un dossier `templates` qui contient l'ensemble des fichiers HTML contenant les pages web de l'interface;
 - ainsi que les fichiers .py pour chaque service tels que : `serviceComposite.py`, `serviceDecision.py`, `serviceEvaluation.py`, `serviceExtraction.py`, `serviceSolvabilite.py`.

7.2 Spécifications des services

7.2.1 Service Composite

```

class serviceComposite(ServiceBase):
    def recupererDossier(numDoss: int): ...

    def nouvelleDemandeClient(numDoss: int): ...

wsdl_app = Application([serviceComposite], ...)

if __name__ == '__main__':

```

Le `service Composite` est l'orchestrateur de ce projet. Il permet de lancer l'évaluation de la demande de prêt et de faire appel aux autres services (Extraction, VerificationSolvabilité, EvaluationPropriété, Décision). Il contient les fonctions suivantes:

- `recupererDossier` : permet de récupérer le formulaire en fichier `txt` dans le dossier `demandeTxt` pour lancer l'évaluation de prêt;
- `nouvelleDemandeClient` : permet de communiquer avec les autres services après lancement de l'évaluation de demande de prêt.

7.2.2 Service Extraction

```

def extractionDonnees(cheminFichierTxt): ...

def ecritureXML(cheminFichierTxt, donnees): ...

class serviceExtraction(ServiceBase):
    @srpc(Unicode, _returns = Unicode)
    def extractionTxt(cheminFichierTxt): ...

wsdl_appExtraction = Application([serviceExtraction], ...)

```

Le **service Extraction** permet d'extraire un fichier .txt (formulaire) en entités, de récupérer les entités nécessaires pour l'évaluation du prêt et d'enregistrer dans un fichier .xml. Il contient les fonctions suivantes:

- **extractionDonnees** : permet d'extraire en entités un fichier .txt;
- **ecritureXML** : permet de créer un nouveau fichier .xml et de remplir entre chaque balise les entités extraites dans la fonction précédente;
- **extractionTxt** : fonction de la classe **ServiceExtraction** qui permet d'appliquer les 2 fonctions précédentes.

7.2.3 Service Solvabilité

```
> def creationDBBanque(): ...
> def recuperation(lienXML, lienJSON): ...
> def calculScoring(donnees, lienXML): ...
class serviceSolvabilite(ServiceBase):
    @srpc(Unicode, _returns = Unicode)
    def verificationSolvabilite(lienXML): ...

> wsdl_appSolvabilite = Application([serviceSolvabilite], ...)
```

Le **service Solvabilité** permet de vérifier la solvabilité en fonction de la situation financière du demandeur de prêt. Pour cela, il va recueillir les données nécessaires du client et calculer le scoring avec ces données. Il contient les fonctions suivantes:

- **creationDBBanque** : permet de créer un fichier JSON qui contient la base de données Banque;
- **recuperation** : permet de récupérer les données concernant le client et sa situation financière provenant de la base de données Banque;
- **calculScoring** : permet de calculer le scoring du client en fonction des données récupérées dans la fonction précédente et d'écrire le score ainsi que la décision du score dans le fichier .xml du dossier demandeXML/;
- **verificationSolvabilite** : fonction de la classe **serviceSolvabilite** qui permet d'appliquer les 3 fonctions précédentes.

7.2.4 Service Évaluation

```
> def creationDBImmobilier(): ...
> def creationDBMarcheImmobilier(): ...
> def recuperationImmobilier(lienXML, lienJSONImmobilier): ...
> def recuperationMarcheImmobilier(lienXML): ...
> def verificationConformite(donnees, lienXML, lienJSONImmobilier): ...
> def valeurMarche(donnees, lienXML, lienJSONMarcheImmobilier): ...
class serviceEvaluation(ServiceBase):
    @srpc(Unicode, _returns=Unicode)
    def evaluationPropriete(lienXML): ...

> wsdl_appEvaluation = Application([serviceEvaluation], ...)
```

Le **serviceEvaluation** permet d'évaluer la propriété pour laquelle le client demande son prêt. Pour cela, il va dans un premier temps vérifier la conformité de la propriété et ensuite vérifier la valeur de cette propriété dans le marché. Il contient les fonctions suivantes:

- **creationDBImmobilier** : permet de créer un fichier JSON qui contient la base de données Immobilier;
- **creationDBMarcheImmobilier** : permet de créer un fichier JSON qui contient la base de données MarcheImmobilier;

- **recupDonneesImmobilier** : permet de récupérer les données concernant les informations de la propriété provenant de la base de données **Immobilier**;
- **recupDonneesMarcheImmobilier** : permet de récupérer les données concernant la situation de la propriété provenant du fichier **.xml** dans le dossier **demandeXML/**;
- **verificationConformite** : permet de vérifier si la situation de la propriété est conforme et légale grâce aux données récupérées dans la fonction précédente et d'écrire la décision de conformité dans le fichier **.xml** du dossier **demandeXML/**;
- **valeurMarche** : permet de calculer la moyenne des valeurs des propriétés similaires situées dans la même ville en récupérant les données nécessaires dans la base de données **marcheImmobilier** et d'écrire l'estimation valeur dans le fichier **.xml** du dossier **demandeXML/**;
- **evaluationPropriete** : fonction de la classe **serviceEvaluation** qui permet d'appliquer les 6 fonctions précédentes.

7.2.5 Service Décision

```
> def recuperXML(lienXML): ...
>
> def decision(donnees,lienXML): ...
>
class serviceDecision(ServiceBase):
    ...
    @srpc(Unicode,_returns = Unicode)
    def decisionApprobation(lienXML): ...
    ...
>
wsdl_appDecision = Application([serviceDecision], ...)
```

Le **serviceDecision** permet de générer une décision suite à l'évaluation du prêt effectué par le **serviceSolvabilité** et le **serviceEvaluation**. Il contient les fonctions suivantes:

- **recuperXML** : permet de récupérer les données nécessaires pour pouvoir générer une décision dans le fichier **.xml** dans le dossier **demandeXML/**;
- **decision** : permet de générer un résultat suite aux évaluations des services précédentes. Ce résultat peut être une acceptation ou un refus. Pour générer le résultat, on se sert des modèles situés dans le dossier **modeles** et on remplace les données manquantes par les données récupérées dans la fonction précédente. On crée ainsi un fichier **.txt** dans le dossier **reponseTxt/** qui sera récupéré par le **service Composite** pour pouvoir l'afficher sur l'interface lorsque le client lui demande les résultats.
- **decisionApprobation** : fonction de la classe **serviceDecision** qui permet d'appliquer les 2 fonctions précédentes.

7.3 Spécificités de SOAP

7.3.1 Spyne Application

```
wsdl_app = Application([serviceComposite],
    tns = 'http://localhost:8000',
    in_protocol = Soap11(validation='lxml'),
    out_protocol = Soap11()
)
```

- Exemple d'une mise en place d'une application **Spyne** (ici pour **serviceComposite**) qui permet d'appliquer le protocole **SOAP** entre chaque service.
- Chaque service dispose d'une déclaration équivalente à celle-ci.

7.3.2 WSGI Application et WSGI Mounter

```
wsgi_app = WsgiApplication(wsdl_app)

wsgi_app = WsgiMounter({
    '' : creerSite(),
    'serviceExtraction': serviceExtraction.wsdl_appExtraction,
    'serviceSolvabilite': serviceSolvabilite.wsdl_appSolvabilite,
    'serviceEvaluation': serviceEvaluation.wsdl_appEvaluation,
    'serviceDecision': serviceDecision.wsdl_appDecision
})

server = make_server('localhost', 8000, wsgi_app)
server.serve_forever()
```

- Déclaration d'une WSGI Application s'appuyant sur une Application Spyne
- Montage des services avec leurs endpoints respectifs grâce à WSGI Mounter
- Démarrage du serveur sur le port 8000 de l'Application Spyne

7.3.3 Interaction entre les services sous le protocole SOAP

- Service Composite ↔ Service Extraction

```
wsdl_url = "http://localhost:8000/serviceExtraction?wsdl"
wsdl_headers = {'Content-Type': 'text/xml'}
requests.get(wsdl_url, headers=wsdl_headers).text

client = Client(wsdl_url)
client.service.extractionTxt(str(CHEMIN_RACINE)+"demandeTxt/"+str(numDoss)+".txt")
soapReçu = client.last_received()

lienSOAP = str(CHEMIN_RACINE)+"soap/extraction.xml"
soapFichier = open(lienSOAP,'w',encoding="utf-8")
soapFichier.write(str(soapReçu))
soapFichier.close()

tree = ET.parse(lienSOAP)
root = tree.getroot()
namespace = {'soap11env': 'http://schemas.xmlsoap.org/soap/envelope/', 'tns': 'serviceExtraction'}
lienXML = str(root.find('.//tns:extractionTxtResult', namespaces=namespace).text)
```

- Récupération du fichier WSDL du `serviceExtraction`
- Création d'un client à l'aide du module `suds`
- Récupération de la requête SOAP et extraction du lien du fichier .XML dans le dossier `demandeXML/` dans le `serviceComposite`

Enveloppe SOAP générée : `extraction.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<soap11env:Envelope xmlns:soap11env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tns="serviceExtraction">
    <soap11env:Body>
        <tns:extractionTxtResponse>
            <tns:extractionTxtResult>./EvaluationDemandePretImmobilier/services/demandeXml/369789.xml</tns:extractionTxtResult>
        </tns:extractionTxtResponse>
    </soap11env:Body>
</soap11env:Envelope>
```

- Service Composite ↔ Service Solvabilité

```

wsdl_url2 = "http://localhost:8000/serviceSolvabilite?wsdl"
client = Client(wsdl_url2)
client.service.verificationSolvabilite(lienXML)
soapReçu2 = client.last_received()

lienSOAP2 = str(CHEMIN_RACINE)+"soap/solvabilite.xml"
soapFichier2 = open(lienSOAP2,'w',encoding="utf-8")
soapFichier2.write(str(soapReçu2))
soapFichier2.close()

```

- Récupération du fichier WSDL du `serviceSolvabilite`
- Création d'un client à l'aide du module `suds`
- Récupération de la requête SOAP et écriture du lien du fichier .XML dans le dossier `demandeXML/` dans le `serviceComposite`

Enveloppe SOAP générée : `solvabilite.xml`

```

<?xml version="1.0" encoding="UTF-8"?>
<soap11env:Envelope xmlns:soap11env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tns="serviceSolvabilite">
  <soap11env:Body>
    <tns:verificationSolvabiliteResponse>
      |   <tns:verificationSolvabiliteResult>./EvaluationDemandePretImmobilier/services/demandeXml/369789.xml</tns:verificationSolvabiliteResult>
    </tns:verificationSolvabiliteResponse>
  </soap11env:Body>
</soap11env:Envelope>

```

- Service Composite ↔ Service Evaluation

```

wsdl_url3 = "http://localhost:8000/serviceEvaluation?wsdl"
client = Client(wsdl_url3)
client.service.evaluationPropriete(lienXML)
soapReçu3 = client.last_received()

lienSOAP3 = str(CHEMIN_RACINE)+"soap/evaluation.xml"
soapFichier3 = open(lienSOAP3,'w',encoding="utf-8")
soapFichier3.write(str(soapReçu3))
soapFichier3.close()

```

- Récupération du fichier WSDL du `serviceEvaluation`
- Création d'un client à l'aide du module `suds`
- Récupération de la requête SOAP et écriture du lien du fichier .XML dans le dossier `demandeXML/` dans le `serviceComposite`

Enveloppe SOAP générée : `evaluation.xml`

```

<?xml version="1.0" encoding="UTF-8"?>
<soap11env:Envelope xmlns:soap11env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tns="serviceEvaluation">
  <soap11env:Body>
    <tns:evaluationProprieteResponse>
      |   <tns:evaluationProprieteResult>./EvaluationDemandePretImmobilier/services/demandeXml/369789.xml</tns:evaluationProprieteResult>
    </tns:evaluationProprieteResponse>
  </soap11env:Body>
</soap11env:Envelope>

```

- Service Composite ↔ Service Décision

```

wsdl_url4 = "http://localhost:8000/serviceDecision?wsdl"
client = Client(wsdl_url4)
client.service.decisionApprobation(lienXML)
soapRecu4 = client.last_received()

lienSOAP4 = str(CHEMIN_RACINE)+"soap/decision.xml"
soapFichier4 = open(lienSOAP4, 'w', encoding="utf-8")
soapFichier4.write(str(soapRecu4))
soapFichier4.close()

tree = ET.parse(lienSOAP4)
root = tree.getroot()
namespace = {'soap11env': 'http://schemas.xmlsoap.org/soap/envelope/', 'tns': 'serviceDecision'}
lienTXT = str(root.find('.//tns:decisionApprobationResult', namespaces=namespace).text)

```

- Récupération du fichier WSDL du `serviceDecision`
- Création d'un client à l'aide du module `suds`
- Récupération de la requête SOAP et écriture du lien du fichier .XML dans le dossier `demandeXML/` dans le `serviceComposite`

Enveloppe SOAP générée : `decision.xml`

```

<?xml version="1.0" encoding="UTF-8"?>
<soap11env:Envelope xmlns:soap11env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tns="serviceDecision">
    <soap11env:Body>
        <tns:decisionApprobationResponse>
            <tns:decisionApprobationResult>./EvaluationDemandePretImmobilier/services/reponseTxt/369789.txt</tns:decisionApprobationResult>
        </tns:decisionApprobationResponse>
    </soap11env:Body>
</soap11env:Envelope>

```

8 Interface

Nous avons effectué une interface afin que le client puisse déposer sa demande de prêt pour l'évaluation et qu'il puisse consulter les résultats.

8.1 Pages web

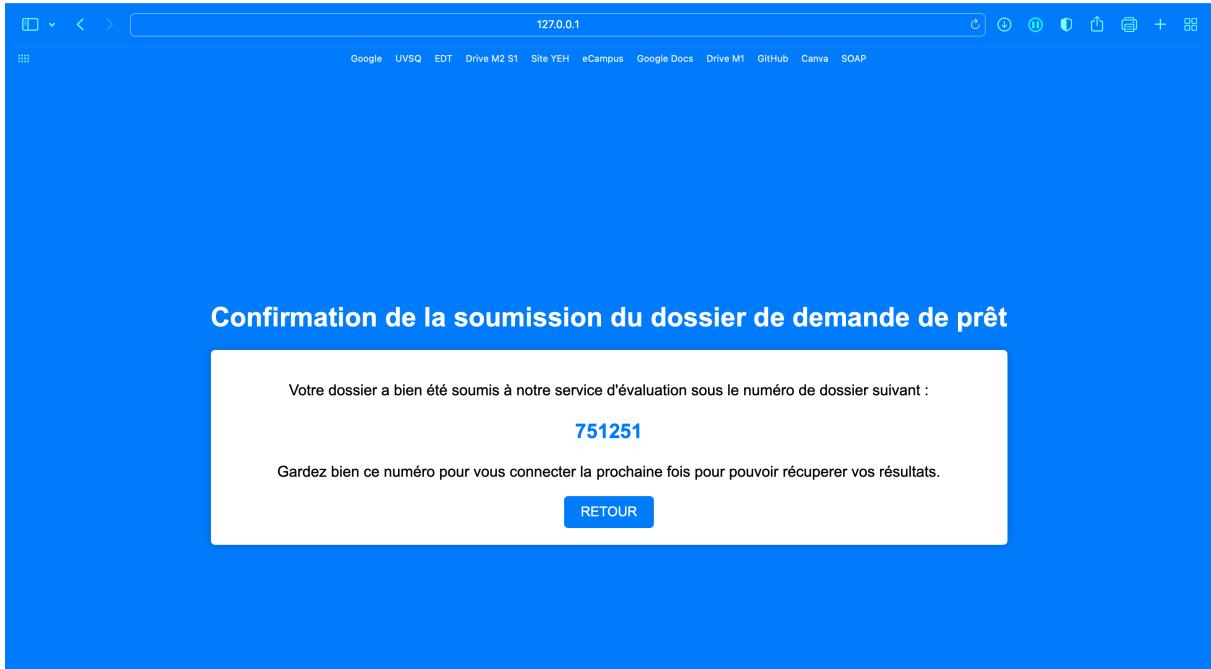
8.1.1 Accueil : page d'accueil

The screenshot shows a web browser window with the URL 127.0.0.1. The title bar says "Bienvenue sur la page d'accueil pour évaluation de votre demande de prêt immobilier". Below this, there are two sections: one for new users ("Vous souhaitez déposer un dossier de demande de prêt immobilier pour une évaluation") with a "DEPOSER UN DOSSIER" button, and one for existing users ("Vous avez déjà déposé votre dossier de demande de prêt et vous souhaitez récupérer les résultats de l'évaluation") with a "VOIR LE RESULTAT" button.

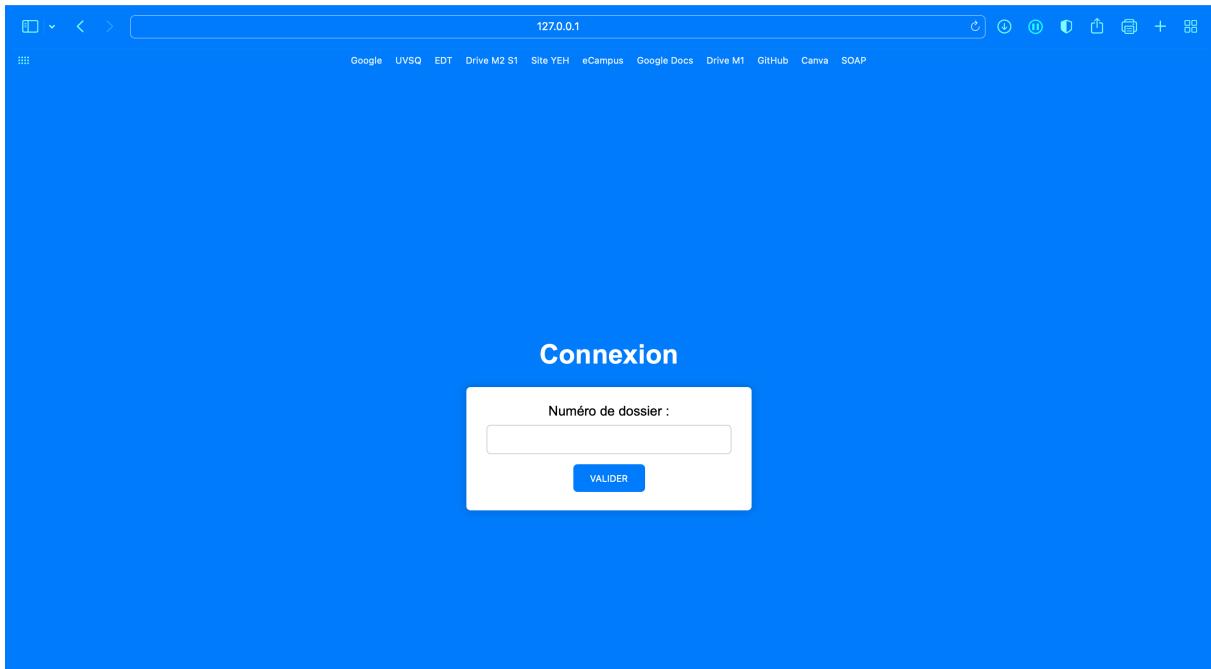
8.1.2 Formulaire : page contenant le formulaire de demande de prêt à évaluer

The screenshot shows a web browser window with the URL 127.0.0.1. The title bar says "Formulaire de demande de prêt". The form contains fields for personal information (Nom complet, Adresse, E-mail), financial details (Montant du prêt, Durée du prêt, Revenu mensuel, Dépense mensuelle), and banking information (Numéro de compte bancaire, Référence de la propriété). A "SOUMETTRE" button is at the bottom right.

8.1.3 Confirmation : page de confirmation de la soumission du formulaire

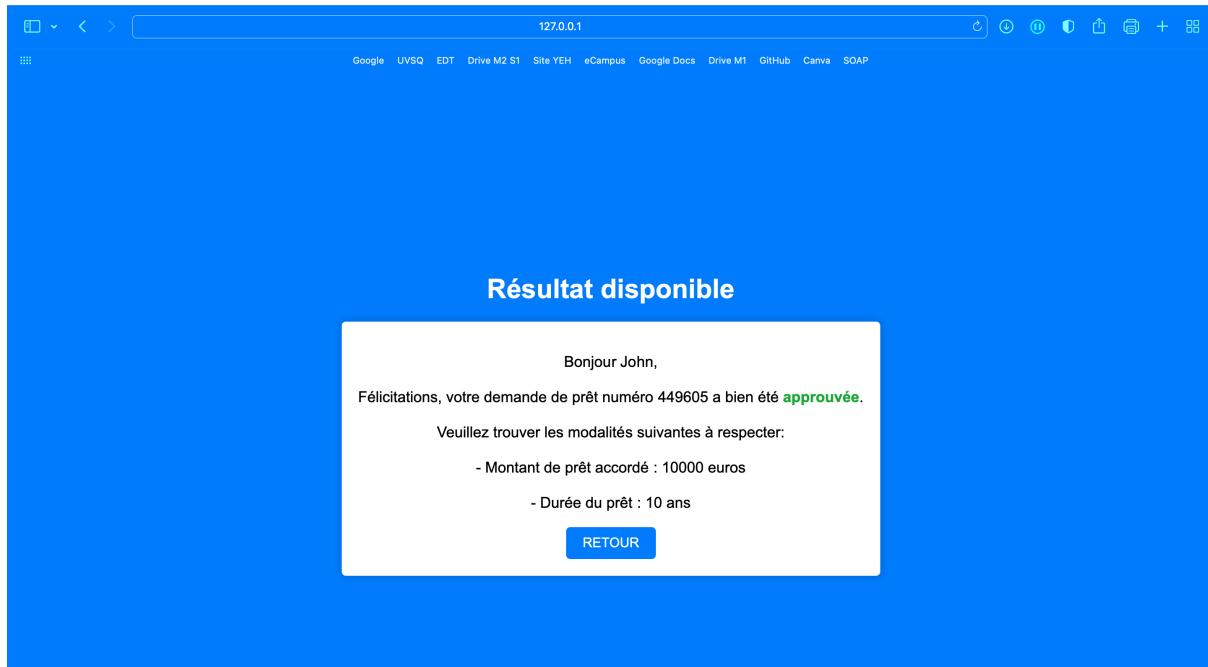


8.1.4 Connexion : page de connexion pour consulter les résultats



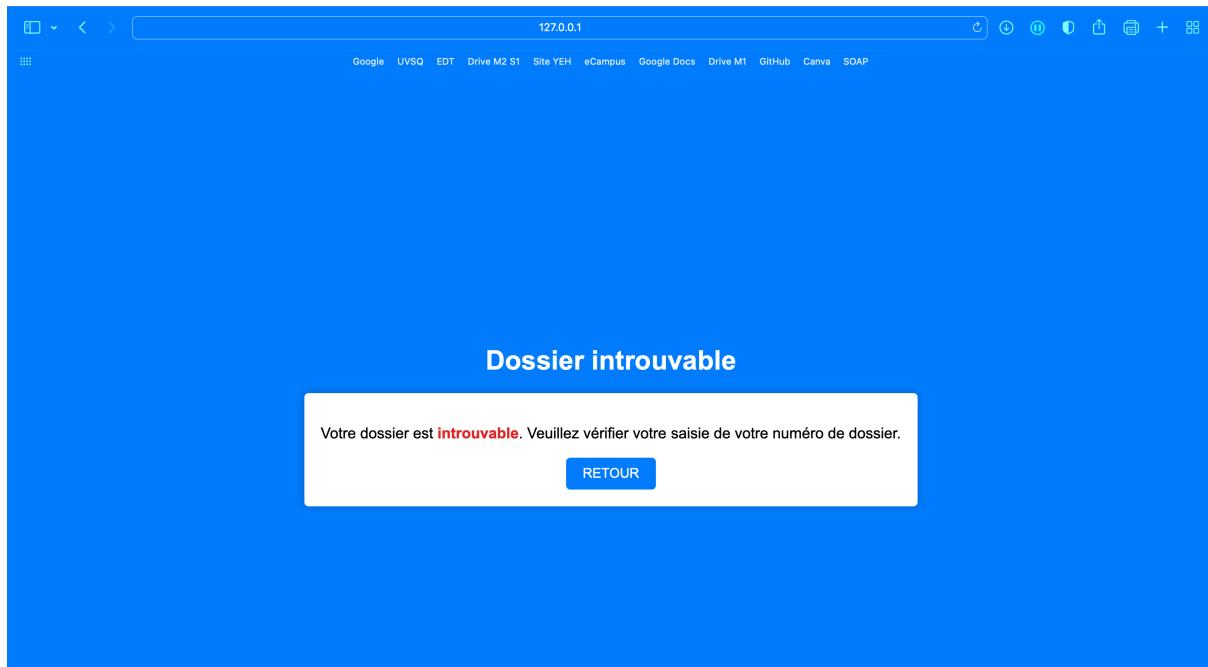
8.1.5 Disponible : page affichant que le résultat est disponible

[Cas où l'évaluation de prêt a eu lieu]



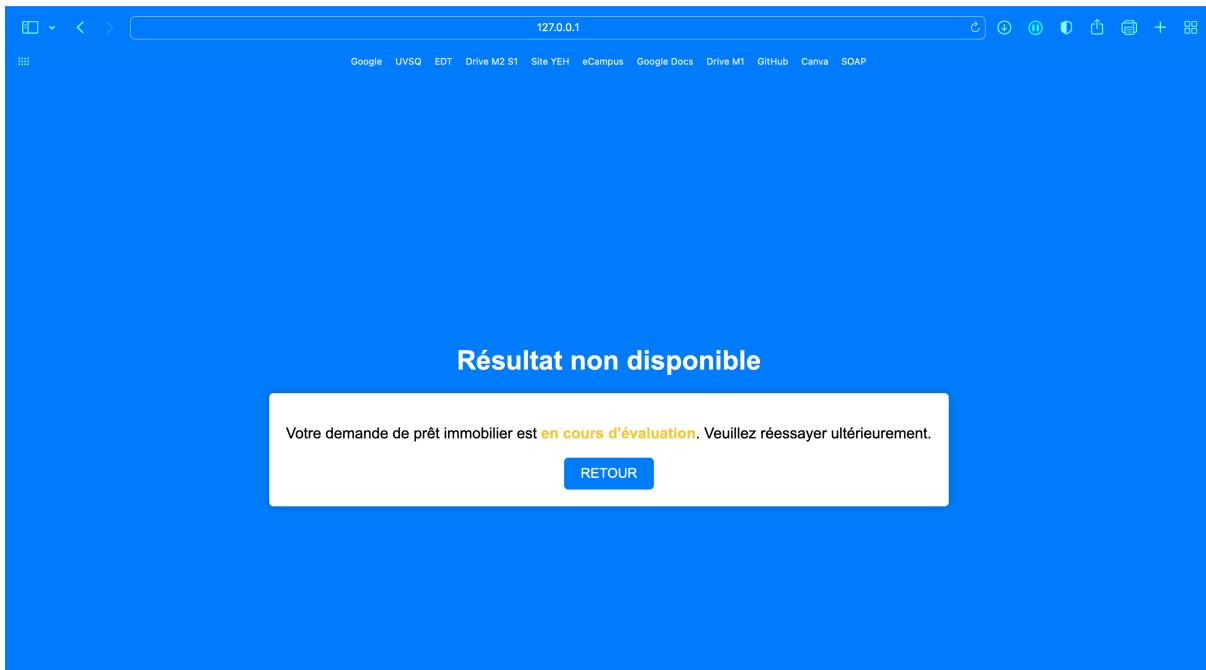
8.1.6 Introuvable : page affichant que le résultat est introuvable

[Cas où l'évaluation de prêt n'a pas eu lieu car le numéro de dossier indiqué est inexistant]



8.1.7 Traitement : page affichant que le résultat n'est pas encore disponible

[Cas où l'évaluation de prêt a eu lieu mais n'a pas aboutit car il y a eu un bug lors de l'évaluation de prêt]



8.2 Lancement de l'application

8.2.1 Commandes

Voici les étapes à suivre afin de pouvoir exécuter l'application:

- Configuration de l'environnement virtuel `venv` et ajout des dépendances nécessaires :
`chmod 774 ./configure` puis `./configure`
- Lancement du serveur web : `chmod 774 ./run` puis `./run`
- Disponibilité de l'interface : `http://localhost:8000`

8.2.2 Côté Client : Web

En lançant l'application, le client arrive sur `http://localhost:8000` qui est la page d'accueil (voir 8.1.1). Il a le choix entre "Déposer un dossier" et "Voir le Résultat".

Cas 1 : DEPOT D'UN DOSSIER

Si il clique sur "Déposer un dossier", il est redirigé vers la page de formulaire. Il remplit le formulaire comme ceci:

The screenshot shows a web browser window with a blue header bar. The main content is a form titled "Formulaire de demande de prêt". The form fields are as follows:

- Nom complet : John Doe
- Adresse : 123 Rue de la Liberté, 75001 Paris, France
- E-mail : john.doe@email.com
- Numéro de téléphone : 0123456789
- Montant du prêt (en euros) : 200000
- Durée du prêt (en années) : 20
- Description de la propriété : Maison a deux étages avec jardin, située dans un quartier résidentiel calme
- Revenu mensuel (en euros) : 5000
- Dépense mensuelle (en euros) : 3000
- Numéro de compte bancaire : 11
- Référence de la propriété : 1

A blue "SOUMETTRE" button is located at the bottom of the form.

Après avoir cliqué sur le bouton "Soumettre", la page de confirmation s'ouvre comme ceci:

The screenshot shows a web browser window with a blue header bar. The main content is a confirmation page titled "Confirmation de la soumission du dossier de demande de prêt". The message on the page is:

Votre dossier a bien été soumis à notre service d'évaluation sous le numéro de dossier suivant :
439971

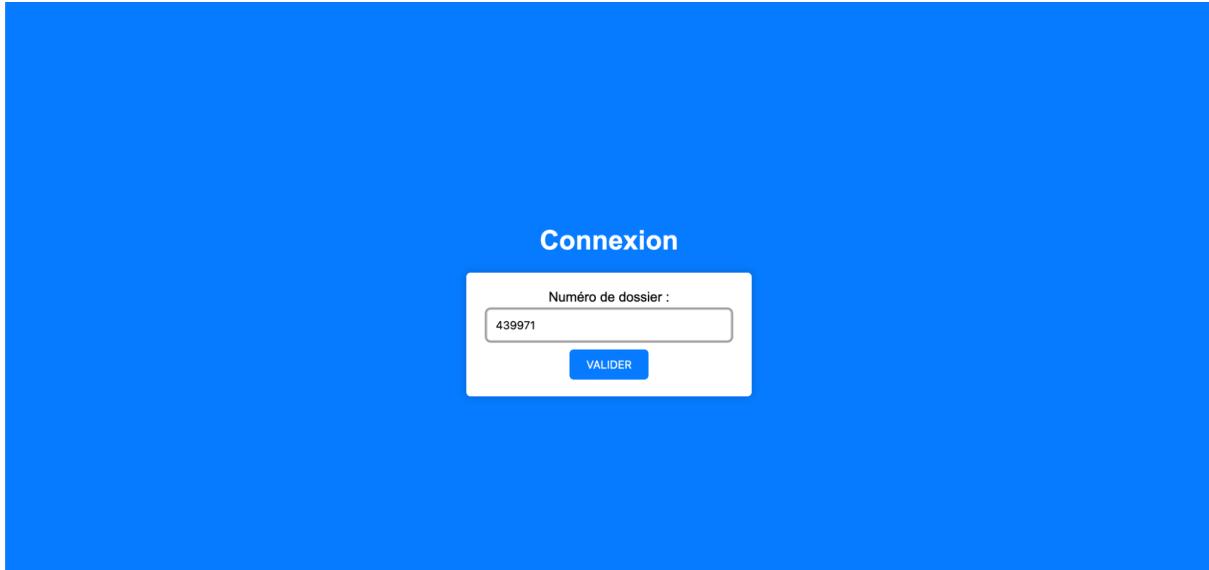
Gardez bien ce numéro pour vous connecter la prochaine fois pour pouvoir récupérer vos résultats.

A blue "RETOUR" button is located at the bottom of the confirmation message.

Elle nous indique le numéro de dossier de la demande qui est à conserver pour pouvoir consulter les résultats par la suite. Si l'utilisateur clique sur "Retour", il revient à la page d'accueil

Cas 2 : VISUALISATION DU RESULTAT

Si il clique sur le bouton "Voir le résultat", la page de connexion s'affiche

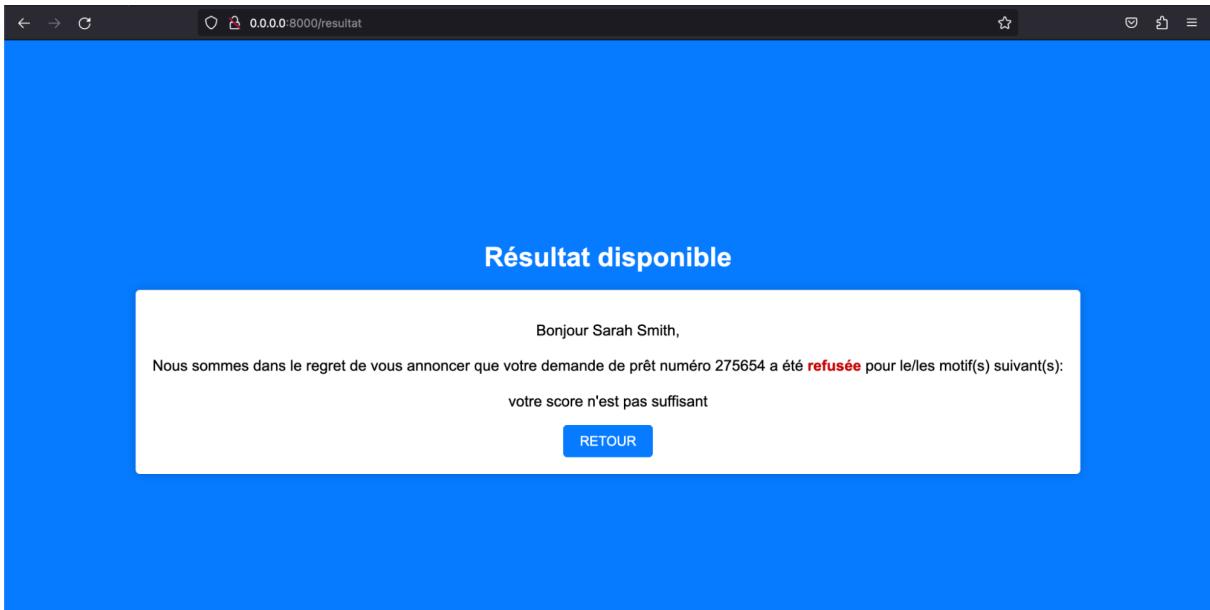


L'utilisateur saisit donc son numéro de dossier qu'il a copié lors de la confirmation de la soumission du formulaire. L'utilisateur accède à la page de résultat.

Voici un exemple lorsque le dossier est accepté, on trouve les modalités à respecter :



Voici un exemple lorsque le dossier est refusé, on trouve les raisons du refus :



8.2.3 Côté Serveur : Terminal

Voici un exemple de ce qui se passe sur le serveur lorsque le client fait sa demande d'évaluation sur l'interface :

```
Démarrage du serveur...
127.0.0.1 - - [05/Dec/2023 21:54:27] "POST /enregistrer HTTP/1.1" 200 840
127.0.0.1 - - [05/Dec/2023 21:54:27] "GET /static/css/confirmation.css HTTP/1.1" 200 1033
127.0.0.1 - - [05/Dec/2023 21:54:27] "GET /serviceExtraction?wsdl HTTP/1.1" 200 2636
127.0.0.1 - - [05/Dec/2023 21:54:27] "GET /serviceExtraction?wsdl HTTP/1.1" 200 2636
127.0.0.1 - - [05/Dec/2023 21:54:27] "POST /serviceExtraction HTTP/1.1" 200 373
127.0.0.1 - - [05/Dec/2023 21:54:27] "GET /serviceSolvabilite?wsdl HTTP/1.1" 200 2862
127.0.0.1 - - [05/Dec/2023 21:54:27] "POST /serviceSolvabilite HTTP/1.1" 200 414
127.0.0.1 - - [05/Dec/2023 21:54:27] "GET /serviceEvaluation?wsdl HTTP/1.1" 200 2765
127.0.0.1 - - [05/Dec/2023 21:54:27] "POST /serviceEvaluation HTTP/1.1" 200 397
127.0.0.1 - - [05/Dec/2023 21:54:27] "GET /serviceDecision?wsdl HTTP/1.1" 200 2755
127.0.0.1 - - [05/Dec/2023 21:54:27] "POST /serviceDecision HTTP/1.1" 200 395
127.0.0.1 - - [05/Dec/2023 21:54:39] "GET / HTTP/1.1" 200 1053
127.0.0.1 - - [05/Dec/2023 21:54:39] "GET /static/css/accueil.css HTTP/1.1" 200 958
127.0.0.1 - - [05/Dec/2023 21:54:40] "GET /connexion.html HTTP/1.1" 200 703
127.0.0.1 - - [05/Dec/2023 21:54:40] "GET /static/css/connexion.css HTTP/1.1" 200 1246
127.0.0.1 - - [05/Dec/2023 21:54:42] "POST /resultat HTTP/1.1" 200 847
127.0.0.1 - - [05/Dec/2023 21:54:42] "GET /static/css/disponible.css HTTP/1.1" 200 1087
127.0.0.1 - - [05/Dec/2023 21:54:52] "POST /enregistrer HTTP/1.1" 200 840
127.0.0.1 - - [05/Dec/2023 21:54:52] "GET /serviceExtraction?wsdl HTTP/1.1" 200 2636
127.0.0.1 - - [05/Dec/2023 21:54:52] "GET /static/css/confirmation.css HTTP/1.1" 200 1033
127.0.0.1 - - [05/Dec/2023 21:54:52] "POST /serviceExtraction HTTP/1.1" 200 373
127.0.0.1 - - [05/Dec/2023 21:54:52] "POST /serviceSolvabilite HTTP/1.1" 200 414
127.0.0.1 - - [05/Dec/2023 21:54:52] "POST /serviceEvaluation HTTP/1.1" 200 397
127.0.0.1 - - [05/Dec/2023 21:54:52] "POST /serviceDecision HTTP/1.1" 200 395
127.0.0.1 - - [05/Dec/2023 21:55:00] "GET / HTTP/1.1" 200 1053
127.0.0.1 - - [05/Dec/2023 21:55:00] "GET /static/css/accueil.css HTTP/1.1" 304 0
127.0.0.1 - - [05/Dec/2023 21:55:00] "GET /connexion.html HTTP/1.1" 200 703
127.0.0.1 - - [05/Dec/2023 21:55:00] "GET /static/css/connexion.css HTTP/1.1" 304 0
127.0.0.1 - - [05/Dec/2023 21:55:00] "GET /static/css/connexion.css HTTP/1.1" 200 1246
127.0.0.1 - - [05/Dec/2023 21:55:03] "POST /resultat HTTP/1.1" 200 791
127.0.0.1 - - [05/Dec/2023 21:55:03] "GET /static/css/disponible.css HTTP/1.1" 200 1087
```

Dépôt du formulaire : création du fichier .txt dans demandeTxt/

```
(base) thivani@MacBook-de-Thiva ProjetSOAP % cat EvaluationDemandePretImmobilier/services/demandeTxt/188190.txt
Nom du Client: John Doe
Adresse: 123 Rue de la Liberte, 75001 Paris, France
Email: johndoe@email.com
Numéro de Téléphone: 0123456789
Montant du Prêt Demandé: 200000
Durée du Prêt: 20
Description de la Propriété: Maison a deux etages avec jardin, situee dans un quartier residentiel calme
Revenu Mensuel: 5000
Dépenses Mensuelles: 3000
Numero de compte bancaire: 11
Référence de la propriété: 1
```

Début de l'évaluation : création du fichier .xml dans demandeXml/

```
(base) thivani@MacBook-de-Thiva ProjetSOAP % cat EvaluationDemandePretImmobilier/services/demandeXml/188190.xml
<clientDemandePrêt><idClient>1</idClient><adresse>123 Rue de la Liberte, 75001 Paris, France</adresse><email>johndoe@email.com</email><numTelephone>0123456789</numTelephone><montantPrêt>200000</montantPrêt><dureePrêt>20</dureePrêt><descriptionPropriete>Maison a deux étages avec jardin, située dans un quartier résidentiel calme</descriptionPropriete><revenuMensuel>5000</revenuMensuel><depensesMensuelles>3000</depensesMensuelles><idBanque>11</idBanque><idPropriete>1</idPropriete><score>20</score><decisionScore>Sous conditions</decisionScore><decisionConforme>Admissible a un prêt immobilier</decisionConforme><estimationValeur>154000</estimationValeur></client>%
```

Fin de l'évaluation : création du fichier .txt dans reponseTxt/

```
(base) thivani@MacBook-de-Thiva ProjetSOAP % cat EvaluationDemandePretImmobilier/services/reponseTxt/188190.txt
Bonjour John Doe,
Félicitations, votre demande de prêt numéro 188190 a bien été <vert>approuvée</vert>.
Veuillez trouver les modalités suivantes à respecter:
- Montant de prêt accordé : 200000 euros
- Durée du prêt : 20 ans
```

```
(base) thivani@MacBook-de-Thiva ProjetSOAP % cat EvaluationDemandePretImmobilier/services/reponseTxt/369789.txt
Bonjour Sarah Smith,
Nous sommes dans le regret de vous annoncer que votre demande de prêt numéro 369789 a été <rouge>refusée</rouge> pour le/les motif(s) suivant(s):
votre score n'est pas suffisant
```

9 Démonstration avec SOAP

9.1 Scénario 1 : Demande de prêt acceptée

9.1.1 Saisie dans le formulaire

Voici les informations à saisir pour ce scénario :

- Nom : John Doe
- Adresse : 123 Rue de la Liberte, 75001 Paris, France
- Email : johndoe@email.com
- Numéro de téléphone : 0123456789
- Montant du prêt : 200000
- Durée du prêt : 20
- Description de la propriété : Maison a deux etages avec jardin, situee dans un quartier résidentiel calme
- Revenus mensuel : 5000
- Dépenses mensuelles : 3000
- Compte bancaire : 11
- Identifiant propriété : 1

9.1.2 Résultat attendu

Voici les résultats attendus pour ce scénario :

- Score : 28
- Décision Score : Sous condition
- Décision Conformité : Admissible a un pret immobilier
- Raisons : /
- EstimationValeur: 154000

9.1.3 Page résultat attendue

Voici la page résultat attendue pour ce scénario:



9.2 Scénario 2 : Demande de prêt refusée

9.2.1 Saisie dans le formulaire

Voici les informations à saisir pour ce scénario :

- Nom : Sarah Smith
- Adresse : 45 avenue des Etats-Unis, 78000 Versailles, France
- Email : sarahsmith@email.com
- Numéro de téléphone : 0123456788
- Montant du prêt : 150000
- Durée du prêt : 30
- Description de la propriété : Appartement située dans un quartier d'affaire
- Revenus mensuel : 2000
- Dépenses mensuelles : 1500
- Compte bancaire : 22
- Identifiant propriété : 2

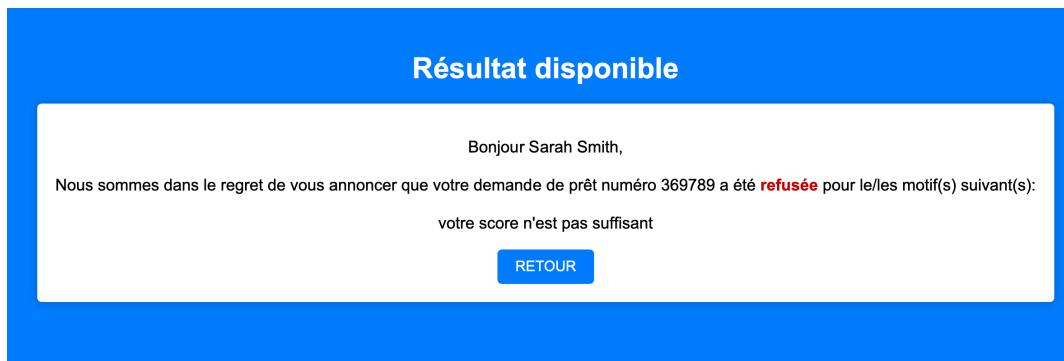
9.2.2 Résultat attendu

Voici les résultats attendus pour ce scénario :

- Score : -1
- Décision Score : Non admissible
- Décision Conformité : Non admissible a un pret immobilier
- Raisons : Normes non reglementaires! Il y a au moins 1 litige en cours
- EstimationValeur: 0

9.2.3 Page résultat attendue

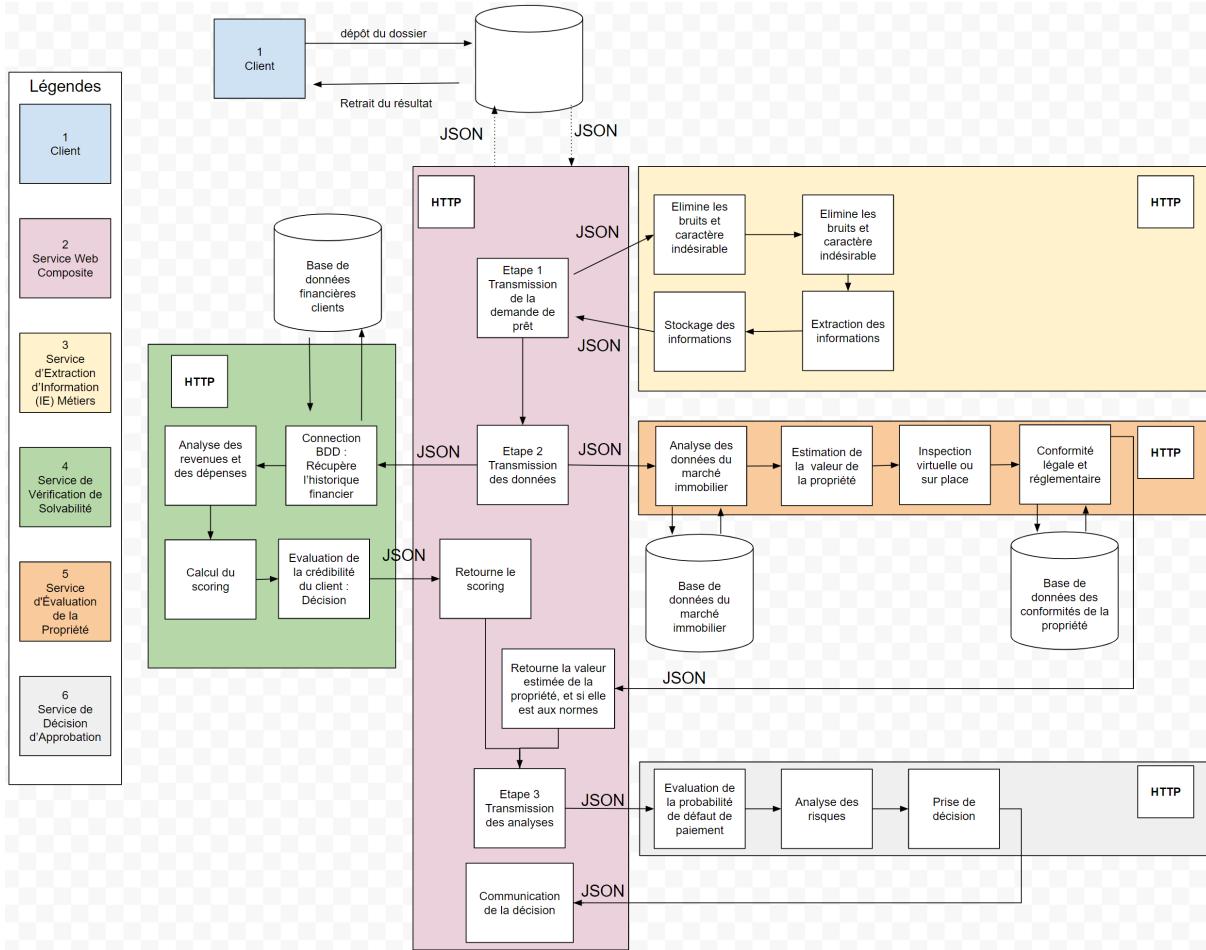
Voici la page résultat attendue pour ce scénario :



10 Modélisation des services sous REST

10.1 Modélisation

Voici une architecture à base de service pour mettre en œuvre le processus d'évaluation de demande de prêt immobilier :



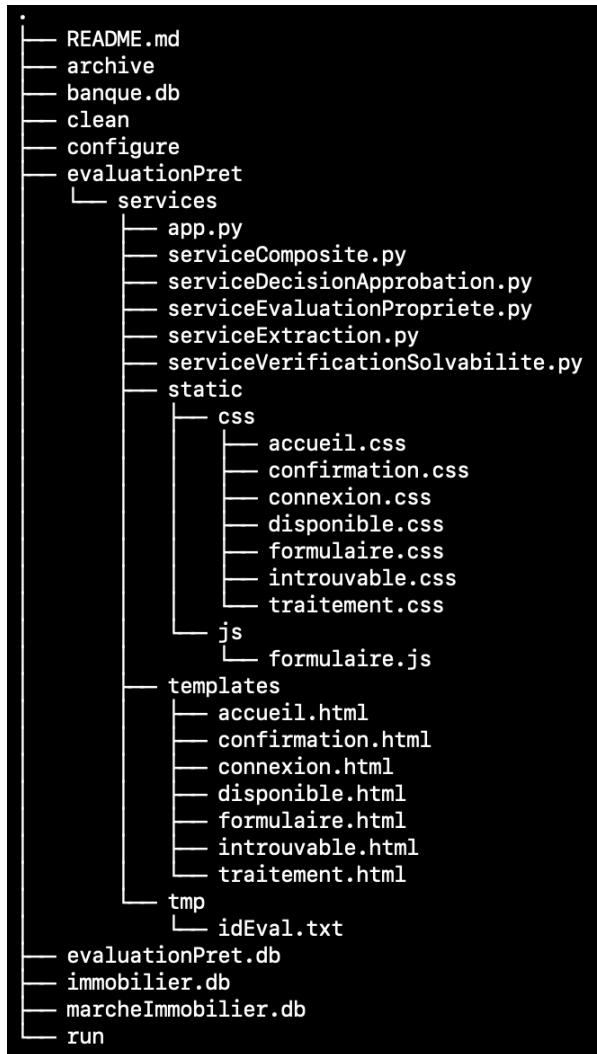
10.2 Différences par rapport au SOAP

Le fonctionnement interne des services se déroule de la même manière que le SOAP à quelques différences près. Voici les éléments qui changent par rapport à la partie sur le SOAP :

- les bases de données Banque, Immobilier, marcheImmobilier changent de format : les fichiers JSON disparaissent et deviennent des BDD SQLite `banque.db`, `immobilier.db` et `marcheImmobilier.db`;
- les fichiers XML et txt disparaissent et deviennent des tables dans la BDD `evaluationPret.db`;
- apparition d'un fichier `App.py` qui contient toutes les routes y compris celles pour communiquer entre les différents services;
- les requêtes faisant appel au fichier WSDL des services n'existent plus;
- les requêtes SOAP disparaissent et laissent place aux requêtes REST à l'aide du module `requests`;
- les messages sont envoyés sous format JSON au lieu du XML entre les services.

11 Implémentation avec REST

11.1 Structure du projet



- Dans le dossier ProjetREST, nous avons:
 - un dossier **evaluationPret**, contenant le coeur du projet;
 - un **README.md**, donnant les commandes et les scénarios à exécuter;
 - des scripts (**archive**, **clean**, **configure**, **run**);
 - et les fichiers **.db** contenant les bases de données SQLite : **evaluationPret**, **banque**, **immobilier**, **marcheImmobilier**.
- Dans le dossier **evaluationPret**, nous avons un dossier **services** contenant le coeur du projet.
- Le projet se situant dans le dossier **services**, il est composé de:
 - le fichier **app.py** qui contient toutes les routes entre les services;
 - les fichiers **.py** pour chaque service tels que : **serviceComposite.py**, **serviceDecisionApprobation.py**, **serviceEvaluationPropriete.py**, **serviceExtraction.py**, **serviceVerificationSolvabilite.py**;
 - un dossier **static** qui contient l'ensemble des fichiers **CSS** et **JS** nécessaire pour l'interface web;
 - un dossier **templates** qui contient l'ensemble des fichiers **HTML** contenant les pages web de l'interface;
 - ainsi qu'un dossier **tmp** qui contient des fichiers temporaires qui sont utilisés lors de l'évaluation tel que **idEval.txt**

11.2 Spécifications des services

11.2.1 Service Composite

```
class serviceComposite:  
>     def creerBD(): ...  
  
>     def lancerEvaluationPret(idDossier, formulaire): ...  
  
>     def recupererDossier(idDossier): ...
```

Le **service Composite** est l'orchestrateur de ce projet. Il permet de lancer l'évaluation de la demande de prêt et de faire appel aux autres services (Extraction, VerificationSolvabilité, EvaluationPropriété, Décision). Il contient les fonctions suivantes:

- **creerBD** : permet de créer la BD **EvaluationPret** avec les tables **DEMANDE**, **EVALUATION**, **RESULTAT** afin de stocker les données utiles lors de l'évaluation de la demande de prêt;
- **lancerEvaluationPret** : permet de lancer l'évaluation de prêt et de communiquer avec les autres services;
- **recupererDossier** : permet de récupérer le formulaire stocké dans la base de données **evaluationPret.db** avec l'**idDossier** mentionné.

11.2.2 Service Extraction

```
class serviceExtraction:  
>     def extraire(idDossier: int, formulaire: str): ...
```

Le **service Extraction** contient une fonction principale **extraire** qui permet de récupérer le formulaire pour un **idDossier** donné, de l'extraire en entités et de créer un nouveau tuple dans la table **EVALUATION**.

11.2.3 Service VerificationSolvabilite

```
class serviceVerificationSolvabilite:  
>     def creationDBBanque(): ...  
  
>     def recupererDonnees(idEvaluation): ...  
  
>     def calculScoring(donnees, idEvaluation): ...  
  
>     def verifier(idEvaluation : int): ...
```

Le **service VerificationSolvabilité** permet de vérifier la solvabilité en fonction de la situation financière du demandeur de prêt. Pour cela, il va recueillir les données nécessaires du client et calculer le scoring avec ces données. Il contient les fonctions suivantes:

- **creationDBBanque** : permet de créer la base de données **Banque** avec une table **BANQUE** afin d'avoir la situation financière des clients;
- **recupererDonnees** : permet de récupérer les données concernant le client et sa situation financière provenant de la base de données **Banque**;
- **calculScoring** : permet de calculer le scoring du client en fonction des données récupérées dans la fonction précédente et d'écrire le score ainsi que la décision du score dans la table **EVALUATION**;
- **verifier** : fonction principale qui permet d'appliquer les 3 fonctions précédentes.

11.2.4 Service EvaluationPropriete

```
class serviceEvaluationPropriete:  
>     def creationDBImmobilier(): ...  
  
>     def creationDBMarcheImmobilier(): ...  
  
>     def recuperDonneesImmobilier(idEvaluation): ...  
  
>     def verificationConformite(donnees, idEvaluation): ...  
  
>     def valeurMarche(donnees, idEvaluation): ...  
  
>     def evaluer(idEvaluation : int): ...
```

Le `serviceEvaluationPropriete` permet d'évaluer la propriété pour laquelle le client demande son prêt. Pour cela, il va dans un premier temps vérifier la conformité de la propriété et ensuite vérifier la valeur de cette propriété dans le marché. Il contient les fonctions suivantes:

- `creationDBImmobilier` : permet de créer la base de données `Immobilier` avec une table `IMMOBILIER` afin d'avoir plus d'informations sur la propriété;
- `creationDBMarcheImmobilier` : permet de créer la base de données `MarcheImmobilier` avec une table `MARCHEIMMOBILIER` afin d'avoir des informations sur les propriétés similaires situées dans la même ville que celle évaluée;
- `recuperDonneesImmobilier` : permet de récupérer les données concernant les informations de la propriété provenant de la base de données `IMMOBILIER`;
- `verificationConformite` : permet de vérifier si la situation de la propriété est conforme et légale grâce aux données récupérées dans la fonction précédente et d'écrire la décision de conformité dans la table `EVALUATION`;
- `valeurMarche` : permet de calculer la moyenne des valeurs des propriétés similaires situées dans la même ville en récupérant les données nécessaires dans la base de données `MARCHEIMMOBILIER` et d'écrire l'estimation valeur dans la table `EVALUATION`;
- `evaluer` : fonction principale qui permet d'appliquer les 5 fonctions précédentes.

11.2.5 Service DecisionApprobation

```
class serviceDecisionApprobation:  
>     def recuperBD(idEvaluation): ...  
  
>     def decision(donnees, idEvaluation): ...  
  
>     def decider(idEvaluation : int): ...
```

Le `serviceDecisionApprobation` permet de générer une décision suite à l'évaluation du prêt effectué par le `serviceVerificationSolvabilite` et le `serviceEvaluationPropriete`. Il contient les fonctions suivantes:

- `recuperBD` : permet de récupérer les données nécessaires pour pouvoir générer une décision dans la table `EVALUATION`;
- `decision` : permet de générer un résultat suite aux évaluations des services précédentes. On crée un tuple dans la table `RESULTAT` qui sera récupéré par le `service Composite` pour pouvoir l'afficher sur l'interface lorsque le client lui demande les résultats.
- `decider` : fonction principale qui permet d'appliquer les 2 fonctions précédentes.

11.2.6 App

```
class App():
    def creerInterfaceWeb(): ...

> if __name__ == '__main__': ...
```

Le fichier App.py regroupe toutes les routes de ce projet: les routes pour chaque page web, les routes pour enregistrer le formulaire ainsi que pour afficher le résultat, ainsi que les routes permettant de communiquer entre 2 services sous l'API REST.

11.3 Interaction entre les services sous l'API REST

- Service Composite ↔ Service Extraction

Request POST afin d'envoyer l'idDossier et le formulaire au service Extraction

```
# Envoi de l'idDossier et formulaire à Extraction
print("— Service Extraction commencée —")

serviceExtractionUrl = "http://127.0.0.1:8000/serviceExtraction"
jsonPost = {"idDossier": idDossier, "formulaire": formulaire}
headers = {"Content-Type": "application/json"}
reponse = requests.post(serviceExtractionUrl, json=jsonPost, headers=headers)

if reponse.status_code != 200:
    print("Endpoint serviceExtraction non OK")
```

Route permettant d'appeler la fonction principale du service Extraction avec idDossier et formulaire récupéré du service Composite

```
@app.route('/serviceExtraction', methods=['POST'])
def getDemande():
    recuServiceComposite = request.get_json()
    serviceExtraction.extraire(recuServiceComposite.get('idDossier'), recuServiceComposite.get('formulaire'))

    return jsonify(recuServiceComposite)
```

Request GET afin de recevoir l'idEvaluation du service Extraction

```
# Reception de l'idEvaluation d'Extraction
serviceCompositeEvalUrl = "http://127.0.0.1:8000/serviceCompositeEval"
reponse = requests.get(serviceCompositeEvalUrl, headers=headers)

if reponse.status_code != 200:
    print("Endpoint serviceComposite non OK")

idEvaluationReponse = reponse.json()
idEvaluation = idEvaluationReponse.get('idEvaluation')

print("— Service Extraction terminée —")
```

Route permettant la communication entre les services Composite et Extraction pour l'idEvaluation

```
@app.route('/serviceCompositeEval', methods=['GET', 'POST'])
def getEvaluation():
    if request.method == 'POST':
        recuperServiceExtraction = request.get_json()
        idEvaluation = recuperServiceExtraction.get('idEvaluation')

        with open(CHEMIN_RACINE+"tmp/idEval.txt", "w") as f:
            f.write(f'{idEvaluation}')
        f.close()

    elif request.method == 'GET':

        with open(CHEMIN_RACINE+"tmp/idEval.txt", "r") as f:
            idEvaluation = f.read()
        f.close()
        return {"idEvaluation" : idEvaluation}

    else:
        print("Méthode non autorisée")

    return jsonify(recuperServiceExtraction)
```

- Service Composite ↔ Service VerificationSolvabilite

Request POST afin d'envoyer l'idEvaluation au service VerificationSolvabilite

```
# Envoi de l'idEvaluation à Verification Solvabilité
print("-- Service Verification Solvabilité commencée --")

serviceVerificationSolvabiliteUrl = "http://127.0.0.1:8000/serviceVerificationSolvabilite"
jsonPost = {"idEvaluation": idEvaluation}
reponse = requests.post(serviceVerificationSolvabiliteUrl, json=jsonPost, headers=headers)

if reponse.status_code != 200:
    print("Endpoint serviceVerificationSolvabilite non OK")

print("-- Service Verification Solvabilité terminée --")
```

Route permettant d'appeler la fonction principale du service VerificationSolvabilite avec idEvaluation récupéré du service Composite

```
@app.route('/serviceVerificationSolvabilite', methods=['POST'])
def getEvaluation1():
    recuperServiceComposite = request.get_json()
    serviceVerificationSolvabilite.verifier(recuperServiceComposite.get('idEvaluation'))

    return jsonify(recuperServiceComposite)
```

- Service Composite ↔ Service EvaluationPropriete

Request POST afin d'envoyer l'idEvaluation au service EvaluationPropriete

```
# Envoi de l'idEvaluation à Evaluation Propriété
print("-- Service Evaluation Propriete commencée --")

serviceEvaluationProprieteUrl = "http://127.0.0.1:8000/serviceEvaluationPropriete"
jsonPost = {"idEvaluation": idEvaluation}
reponse = requests.post(serviceEvaluationProprieteUrl, json=jsonPost, headers=headers)

if reponse.status_code != 200:
    print("Endpoint serviceEvaluationPropriete non OK")

print("-- Service Evaluation Propriété terminée --")
```

Route permettant d'appeler la fonction principale du service EvaluationPropriete avec idEvaluation récupéré du service Composite

```
@app.route('/serviceEvaluationPropriete', methods=['POST'])
def getEvaluation2():
    recuServiceComposite = request.get_json()
    serviceEvaluationPropriete.evaluer(recuServiceComposite.get('idEvaluation'))

    return jsonify(recuServiceComposite)
```

- Service Composite ↔ Service DecisionApprobation

Request POST afin d'envoyer l'idEvaluation au service DecisionApprobation

```
# Envoi de l'idEvaluation à Decision
print("-- Service Décision commencée --")

serviceDecisionApprobationUrl = "http://127.0.0.1:8000/serviceDecisionApprobation"
jsonPost = {'idEvaluation': idEvaluation}
reponse = requests.post(serviceDecisionApprobationUrl, json=jsonPost, headers=headers)

if reponse.status_code != 200:
    print("Endpoint serviceDecision non OK")

print("-- Service Décision terminée --")
```

Route permettant d'appeler la fonction principale du service DecisionApprobation avec idEvaluation récupéré du service Composite

```
@app.route('/serviceDecisionApprobation', methods=['POST'])
def getEvaluation3():
    recuServiceComposite = request.get_json()
    serviceDecisionApprobation.decider(recuServiceComposite.get('idEvaluation'))

    return jsonify(recuServiceComposite)
```

12 Démonstration avec REST

12.1 Scénario 1 : Profil parfait

12.1.1 Saisie dans le formulaire

Voici les informations à saisir pour ce scénario :

- Nom : Paul Gauthier
- Adresse : 12 Rue du Pont Neuf, 91120 Palaiseau, France
- Email : paul.gauthier@email.com
- Numéro de téléphone : 01 12 23 34 45
- Montant du pret : 160000
- Duree du pret : 10
- Description de la propriete : Appartement a 6 etages avec un petit balcon
- Revenus mensuel : 12000
- Depenses mensuelles : 2000
- Compte bancaire : 33
- Identifiant propriété : 3

12.1.2 Résultat attendu

Voici les résultats attendus pour ce scénario :

- Score : 32
- Décision Score : Très favorable
- Décision Conformité : Admissible a un pret immobilier
- Raisons : /
- EstimationValeur: 148000

12.1.3 Page résultat attendu

Voici la page résultat attendue pour ce scénario:



12.2 Scénario 2 : Profil emploi instable

12.2.1 Saisie dans le formulaire

Voici les informations à saisir pour ce scénario :

- Nom : Jack Daniel
- Adresse : 52 Avenue de la gare, 91120 Palaiseau, France
- Email : jack.daniel@email.com
- Numéro de téléphone : 01 12 23 34 46
- Montant du pret : 130000
- Duree du pret : 15
- Description de la propriete : Appartement de 5 etages
- Revenus mensuel : 7000
- Depenses mensuelles : 3000
- Compte bancaire : 44
- Identifiant propriété : 4

12.2.2 Résultat attendu

Voici les résultats attendus pour ce scénario :

- Score : 27
- Décision Score : Sous condition
- Décision Conformité : Admissible a un pret immobilier
- Raisons : /
- EstimationValeur: 135000

12.2.3 Page résultat attendu

Voici la page résultat attendue pour ce scénario:



12.3 Scénario 3 : Profil score trop faible

12.3.1 Saisie dans le formulaire

Voici les informations à saisir pour ce scénario :

- Nom : Victor Wolf
- Adresse : 39 Rue Geais Padidee, 91120 Palaiseau, France
- Email : victor.wolf@email.com
- Numéro de téléphone : 01 12 23 34 47
- Montant du pret : 70000
- Duree du pret : 25
- Description de la propriété : Maison avec deux étages et un grand jardin
- Revenus mensuel : 10000
- Dépenses mensuelles : 6000
- Compte bancaire : 55
- Identifiant propriété : 5

12.3.2 Résultat attendu

Voici les résultats attendus pour ce scénario :

- Score : 12
- Décision Score : Non Admissible
- Décision Conformité : Admissible à un prêt immobilier
- Raisons : Score pas suffisant
- EstimationValeur: 200000

12.3.3 Page résultat attendu

Voici la page résultat attendue pour ce scénario:



12.4 Affichage côté Serveur

12.4.1 Scénario 1 : Profil parfait

```
*****Nouvelle demande n°181477*****
-- Service Extraction commencée --
127.0.0.1 -- [05/Dec/2023 22:38:58] "GET /static/css/confirmation.css HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:38:58] "POST /serviceCompositeEval HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:38:58] "POST /serviceExtraction HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:38:58] "GET /serviceCompositeEval HTTP/1.1" 200 -
-- Service Extraction terminée --
-- Service Verification Solvabilité commencée --
127.0.0.1 -- [05/Dec/2023 22:38:58] "POST /serviceVerificationSolvabilite HTTP/1.1" 200 -
-- Service Verification Solvabilité terminée --
-- Service Evaluation Propriété commencée --
127.0.0.1 -- [05/Dec/2023 22:38:58] "POST /serviceEvaluationPropriete HTTP/1.1" 200 -
-- Service Evaluation Propriété terminée --
-- Service Décision commencée --
127.0.0.1 -- [05/Dec/2023 22:38:58] "POST /serviceDecisionApprobation HTTP/1.1" 200 -
-- Service Décision terminée --
127.0.0.1 -- [05/Dec/2023 22:39:05] "GET / HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:05] "GET /static/css/accueil.css HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:05] "GET /static/css/accueil.css HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:06] "GET /connexion.html HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:06] "GET /static/css/connexion.css HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:06] "GET /static/css/connexion.css HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:09] "POST /resultat HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:09] "GET /static/css/disponible.css HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:13] "POST /enregistrer HTTP/1.1" 200 -
```

12.4.2 Scénario 2 : Profil emploi instable

```
*****Nouvelle demande n°139359*****
-- Service Extraction commencée --
127.0.0.1 -- [05/Dec/2023 22:39:13] "GET /static/css/confirmation.css HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:13] "POST /serviceCompositeEval HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:13] "POST /serviceExtraction HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:13] "GET /serviceCompositeEval HTTP/1.1" 200 -
-- Service Extraction terminée --
-- Service Verification Solvabilité commencée --
127.0.0.1 -- [05/Dec/2023 22:39:13] "POST /serviceVerificationSolvabilite HTTP/1.1" 200 -
-- Service Verification Solvabilité terminée --
-- Service Evaluation Propriété commencée --
127.0.0.1 -- [05/Dec/2023 22:39:13] "POST /serviceEvaluationPropriete HTTP/1.1" 200 -
-- Service Evaluation Propriété terminée --
-- Service Décision commencée --
127.0.0.1 -- [05/Dec/2023 22:39:13] "POST /serviceDecisionApprobation HTTP/1.1" 200 -
-- Service Décision terminée --
127.0.0.1 -- [05/Dec/2023 22:39:21] "GET / HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:21] "GET /static/css/accueil.css HTTP/1.1" 304 -
127.0.0.1 -- [05/Dec/2023 22:39:21] "GET /static/css/accueil.css HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:22] "GET /connexion.html HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:22] "GET /static/css/connexion.css HTTP/1.1" 304 -
127.0.0.1 -- [05/Dec/2023 22:39:23] "GET /static/css/connexion.css HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:25] "POST /resultat HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:25] "GET /static/css/disponible.css HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:32] "POST /enregistrer HTTP/1.1" 200 -
```

12.4.3 Scénario 3 : Profil score trop faible

```
*****Nouvelle demande n°563799*****
-- Service Extraction commencée --
127.0.0.1 -- [05/Dec/2023 22:39:32] "POST /serviceCompositeEval HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:32] "POST /serviceExtraction HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:32] "GET /static/css/confirmation.css HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:32] "GET /serviceCompositeEval HTTP/1.1" 200 -
-- Service Extraction terminée --
-- Service Verification Solvabilité commencée --
127.0.0.1 -- [05/Dec/2023 22:39:32] "POST /serviceVerificationSolvabilite HTTP/1.1" 200 -
-- Service Verification Solvabilité terminée --
-- Service Evaluation Propriete commencée --
127.0.0.1 -- [05/Dec/2023 22:39:32] "POST /serviceEvaluationPropriete HTTP/1.1" 200 -
-- Service Evaluation Propriété terminée --
-- Service Décision commencée --
127.0.0.1 -- [05/Dec/2023 22:39:32] "POST /serviceDecisionApprobation HTTP/1.1" 200 -
-- Service Décision terminée --
127.0.0.1 -- [05/Dec/2023 22:39:38] "GET / HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:38] "GET /static/css/accueil.css HTTP/1.1" 304 -
127.0.0.1 -- [05/Dec/2023 22:39:38] "GET /static/css/accueil.css HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:39] "GET /connexion.html HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:39] "GET /static/css/connexion.css HTTP/1.1" 304 -
127.0.0.1 -- [05/Dec/2023 22:39:39] "GET /static/css/connexion.css HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:42] "POST /resultat HTTP/1.1" 200 -
127.0.0.1 -- [05/Dec/2023 22:39:42] "GET /static/css/disponible.css HTTP/1.1" 200 -
```

12.5 Bases de données evaluationPret.db

12.5.1 Table DEMANDE : formulaires saisis par les 3 clients des scénarios précédents

```
sqlite> select * from demande;
139359|Nom du Client: Jack Daniel
Adresse: 52 Avenue de la gare, 91120 Palaiseau, France
Email: jack.daniel@email.com
Numéro de Téléphone: 01 12 23 34 46
Montant du Prêt Demandé: 130000
Durée du Prêt: 15
Description de la Propriété: Appartement de 5 étages
Revenu Mensuel: 7000
Dépenses Mensuelles: 3000
Numéro de compte bancaire: 44
Référence de la propriété: 4

181477|Nom du Client: Paul Gauthier
Adresse: 12 Rue du Pont Neuf, 91120 Palaiseau, France
Email: paul.gauthier@email.com
Numéro de Téléphone: 01 12 23 34 45
Montant du Prêt Demandé: 160000
Durée du Prêt: 10
Description de la Propriété: Appartement à 6 étages avec un petit balcon
Revenu Mensuel: 12000
Dépenses Mensuelles: 2000
Numéro de compte bancaire: 33
Référence de la propriété: 3

563799|Nom du Client: Victor Wolf
Adresse: 39 Rue Geais Padidee, 91120 Palaiseau, France
Email: victor.wolf@email.com
Numéro de Téléphone: 01 12 23 34 47
Montant du Prêt Demandé: 70000
Durée du Prêt: 25
Description de la Propriété: Maison avec deux étages et un grand jardin
Revenu Mensuel: 10000
Dépenses Mensuelles: 6000
Numéro de compte bancaire: 55
Référence de la propriété: 5
```

12.5.2 Table EVALUATION : fin de l'évaluation des prêts pour les 3 clients

```
sqlite> select * from evaluation;
4/7466|181477|Paul Gauthier|12 Rue du Pont Neuf, 91120 Palaiseau, France|paul.gauthier@email.com|01 12 23 34 45|160000|10|Appartement à 6 étages avec un petit balcon |12000|2000|33|3|32|Très favorable|Admissible|181477|139359|Jack Daniel|52 Avenue de la gare, 91120 Palaiseau, France|jack.daniel@email.com|01 12 23 34 46|130000|15|Appartement de 5 étages|7000|3000|44|4|27|Sous conditions|Admissible à un prêt immobilier|139359|563799|Victor Wolf|39 Rue Geais Padidee, 91120 Palaiseau, France|victor.wolf@email.com|01 12 23 34 47|70000|25|Maison avec deux étages et un grand jardin|10000|6000|55|5|12|Non Admissible|Admissible à un prêt immobilier|200000
```

12.5.3 Table RESULTAT : résultats des 3 clients

```
sqlite> select * from resultat;
139359|Bonjour Jack Daniel,$ Félicitations, votre demande de prêt numéro 139359 a bien été <vert>approuvée</vert>. $Veuillez trouver les modalités suivantes à respecter: $- Montant de prêt accordé : 130000
$ euros $- Durée du prêt : 15 ans
181477|Bonjour Paul Gauthier,$ Félicitations, votre demande de prêt numéro 181477 a bien été <vert>approuvée</vert>. $Veuillez trouver les modalités suivantes à respecter: $- Montant de prêt accordé : 160000
$ euros $- Durée du prêt : 10 ans
563799|Bonjour Victor Wolf,$ Nous sommes dans le regret de vous annoncer que votre demande de prêt numéro 563799 a été <rouge>refusée</rouge> pour le/les motif(s) suivant(s): $Votre score n'est pas suffisant
```