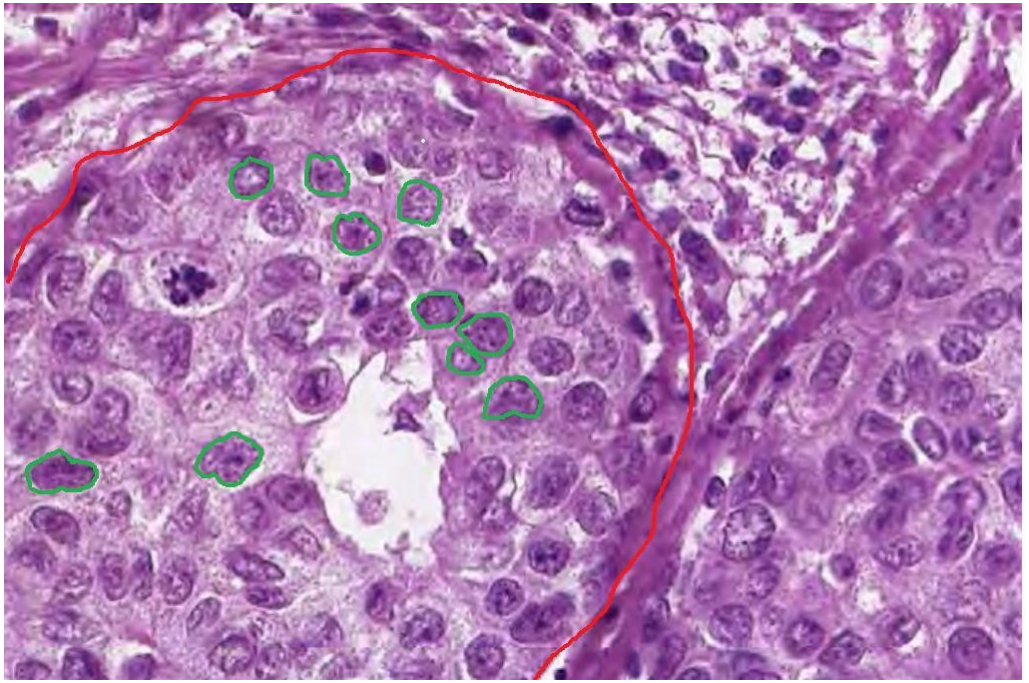
Exemple d’un logiciel de dépistage précoce du cancer à partir de l’analyse d’images de cytologie

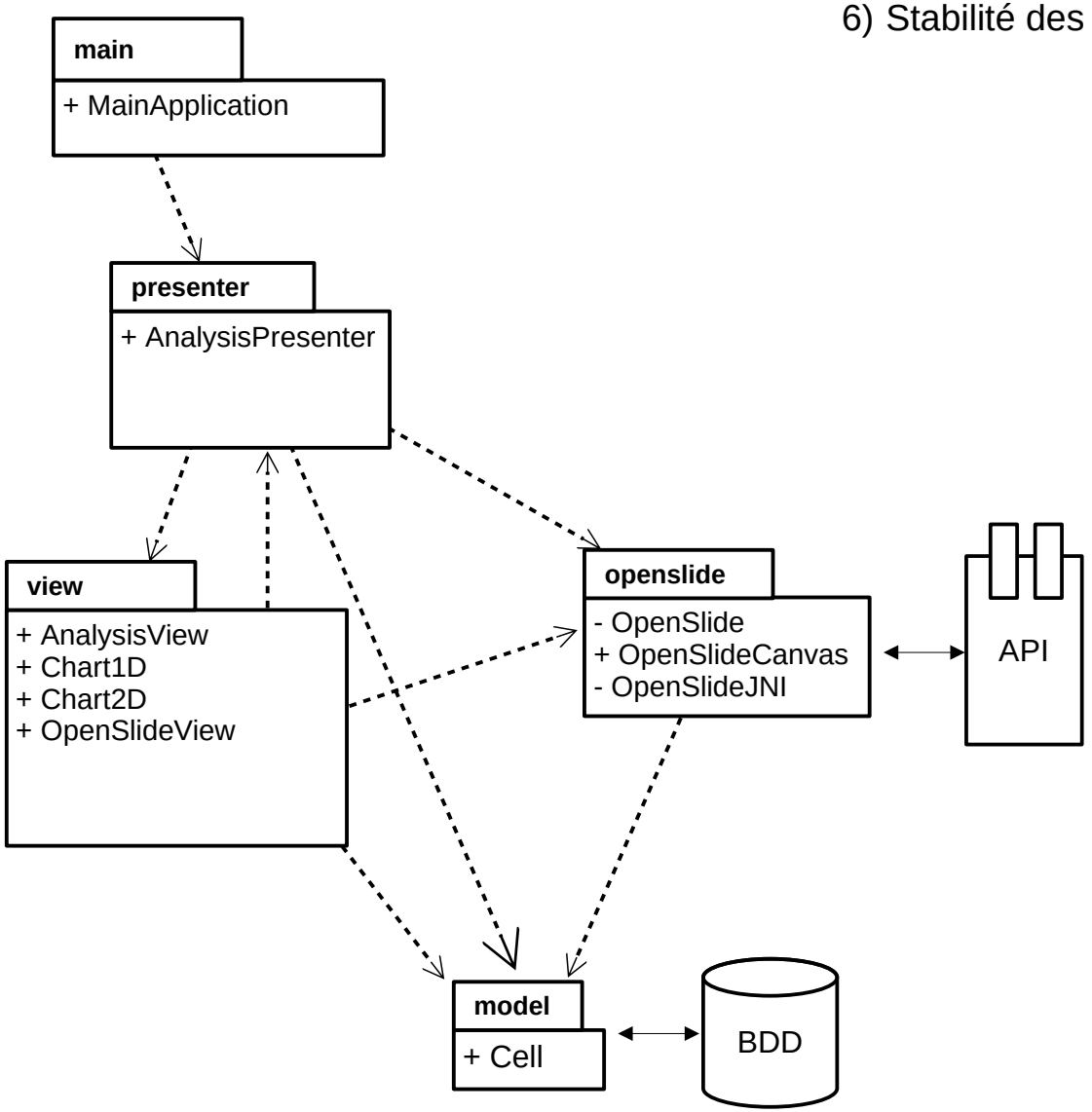
Le logiciel consiste à afficher aux médecins des statistiques concernant des caractéristiques de cellules segmentées dans des images de cytologie (caractéristiques topologiques, morphologiques, colorimétriques, photométriques…) en vue d’un dépistage précoce du cancer.

Le logiciel utilise la bibliothèque OpenSlide qui permet de lire et d’afficher des images médicales de grande taille (100 000 x 100 000 pixels).



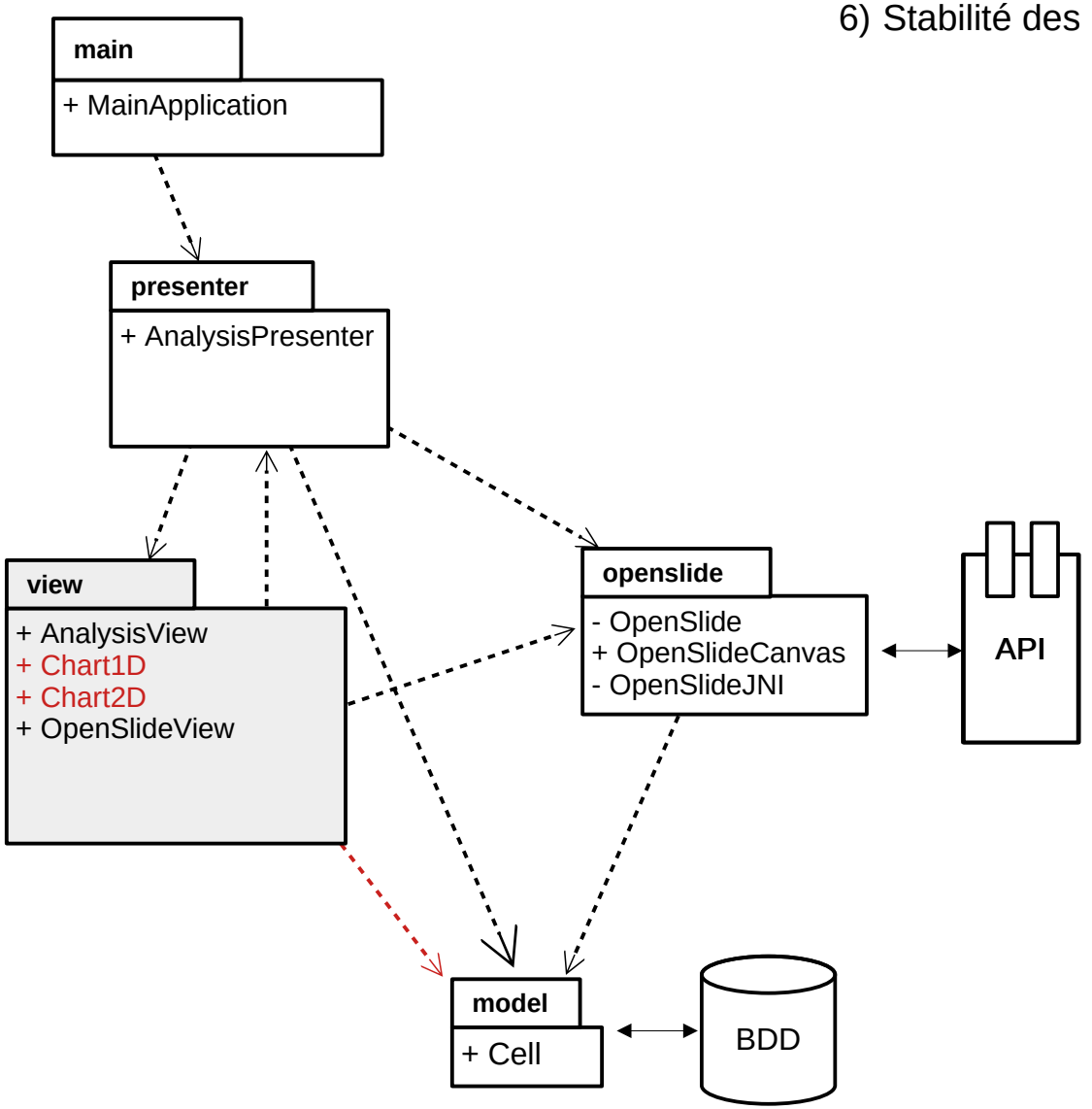
Soit le diagramme de paquets initiale qui reflète une architecture MVP.

Principes
1) Équivalence réutilisation / livraison
2) Fermeture commune
3) Réutilisation commune
4) Dépendances acycliques
5) Relation dépendance / stabilité
6) Stabilité des abstractions



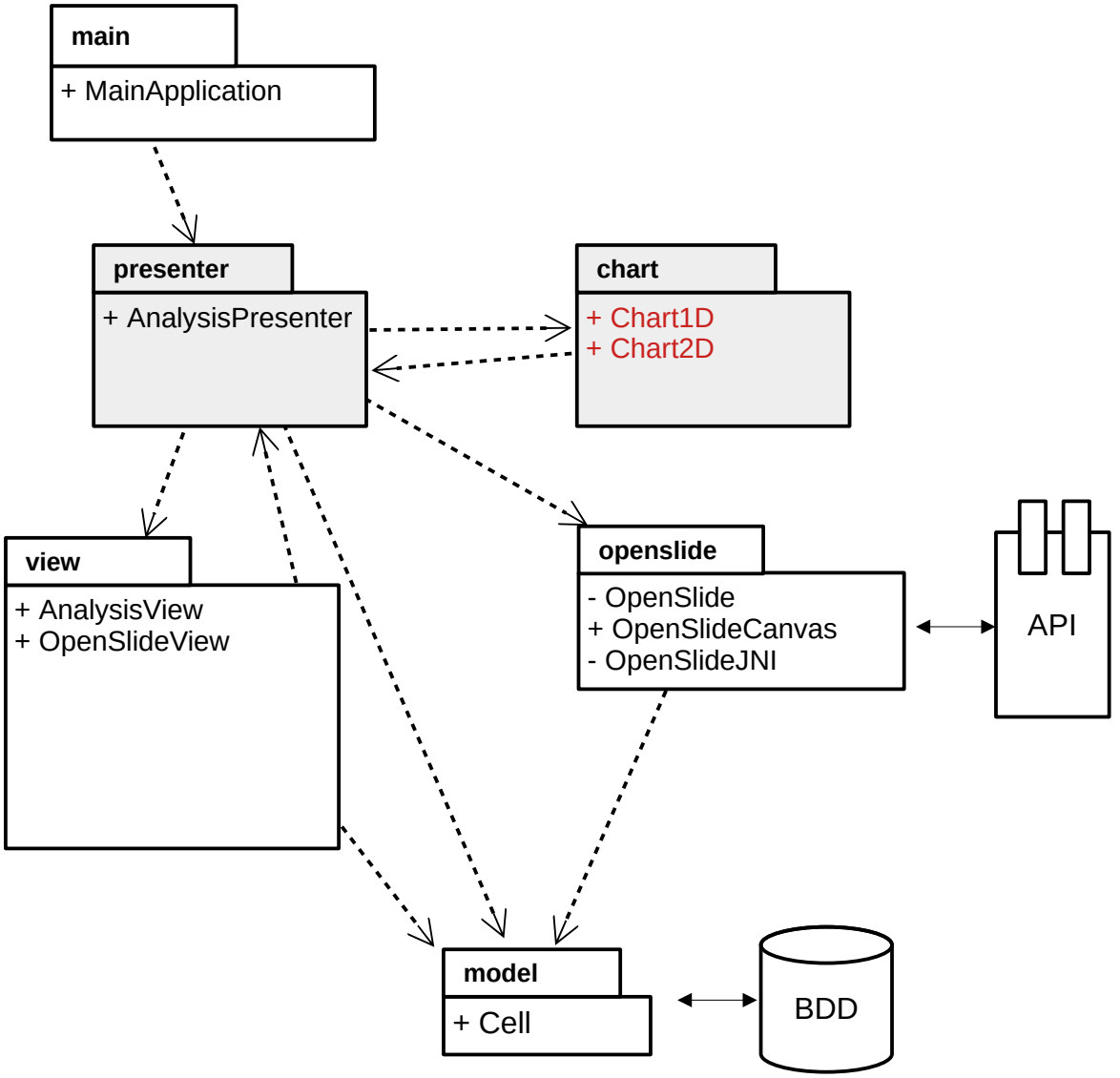
Problèmes :
- Non respect du principe de fermeture commune (ie, responsabilité unique) dans le paquet **view** : analyse + chart
- Dépendance entre la **vue** et le **modèle** (ne respecte pas l'architecture 3-tier)

Principes
1) Équivalence réutilisation / livraison
2) Fermeture commune
3) Réutilisation commune
4) Dépendances acycliques
5) Relation dépendance / stabilité
6) Stabilité des abstractions



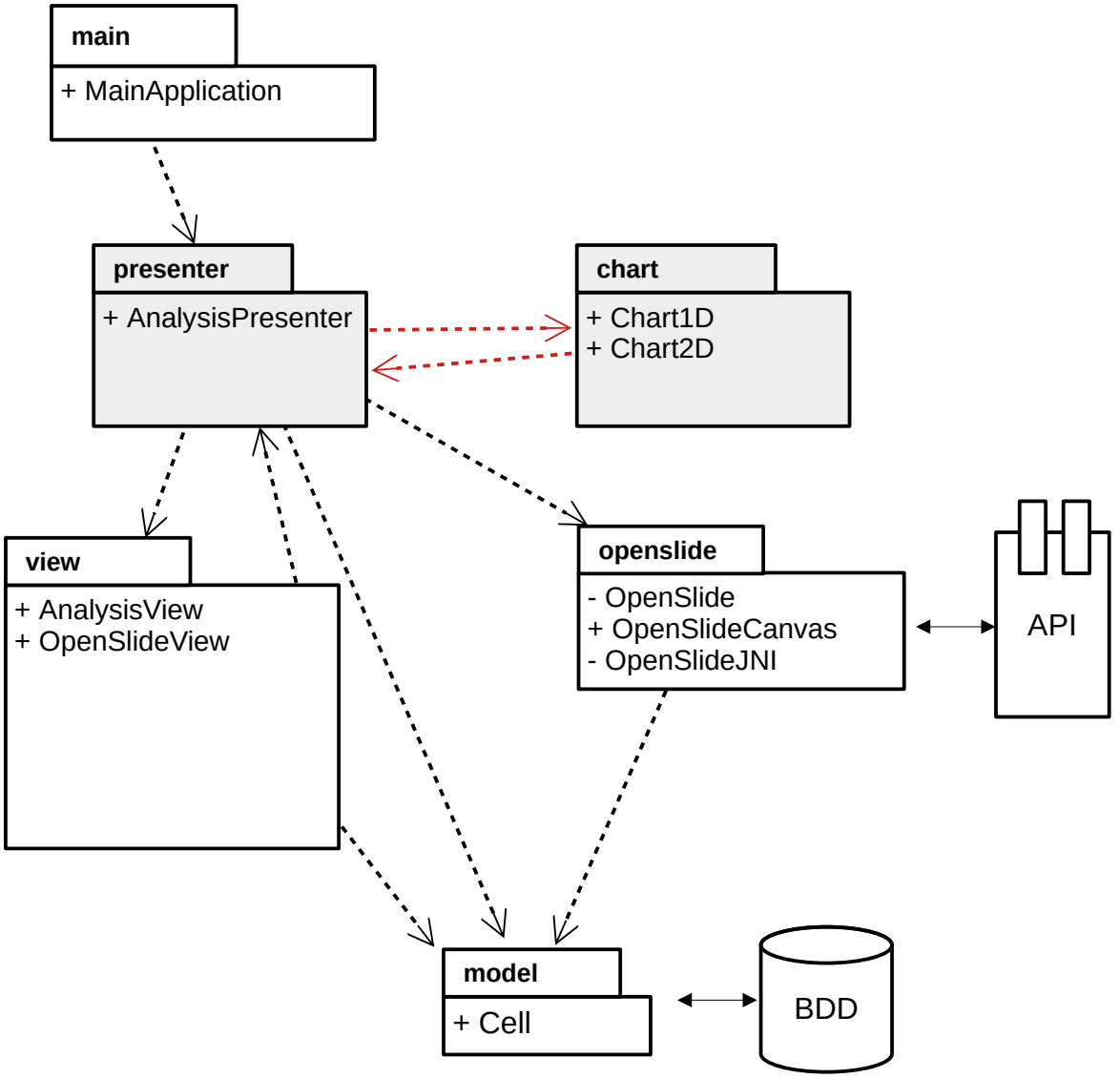
Solution :
- un paquet pour les charts

Principes
1) Équivalence réutilisation / livraison
2) Fermeture commune
3) Réutilisation commune
4) Dépendances acycliques
5) Relation dépendance / stabilité
6) Stabilité des abstractions



Problème :
-Dépendance acyclique entre « **presenter** » et « **chart** » puisque le présentateur crée les charts et que les charts ont besoin des données du présentateur

Principes
1) Équivalence réutilisation / livraison
2) Fermeture commune
3) Réutilisation commune
4) Dépendances acycliques
5) Relation dépendance / stabilité
6) Stabilité des abstractions



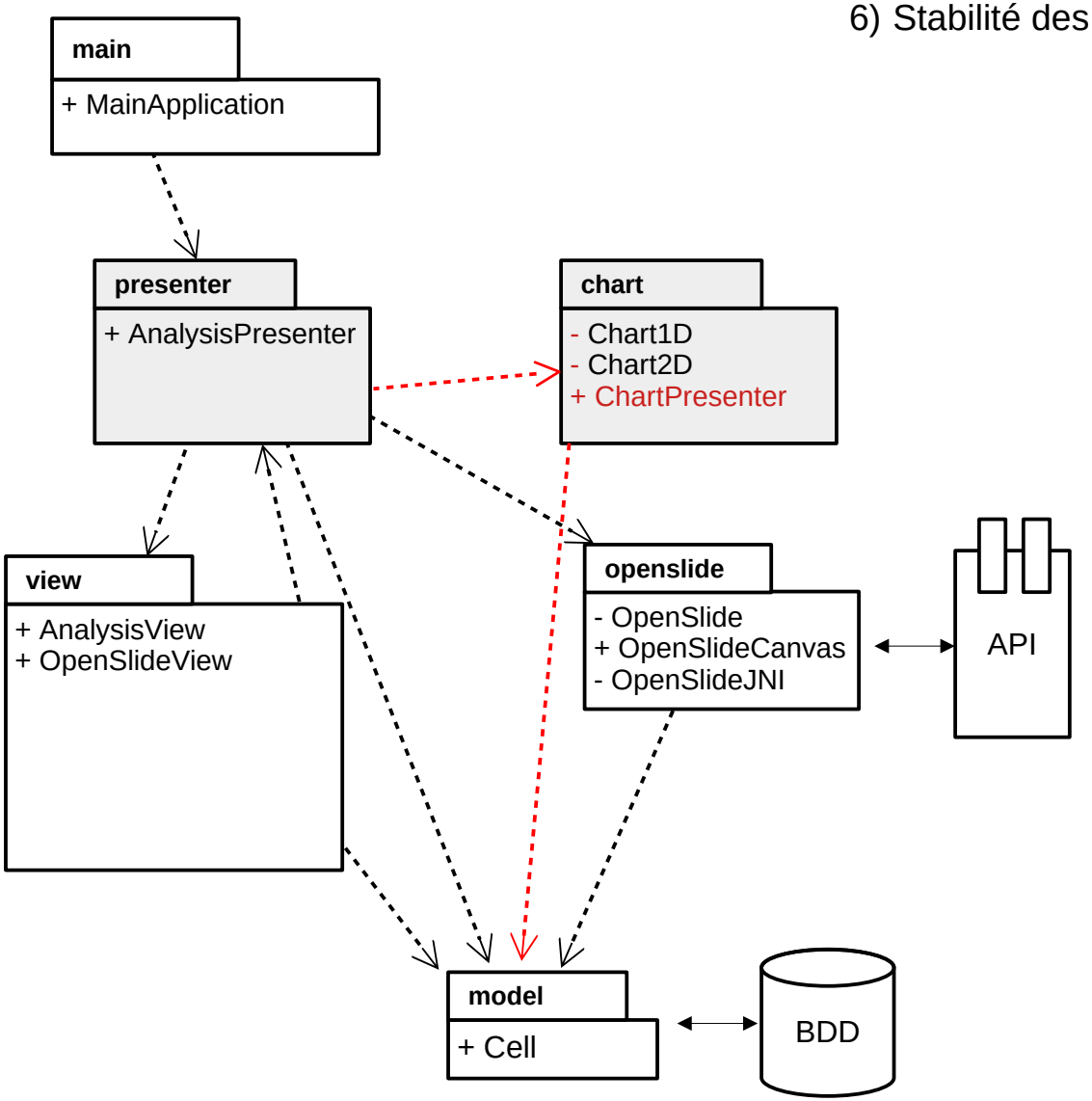
Solution

1/ Extraire du présentateur les fonctions de gestion des charts et faire une classe qui gère les charts → casse la dépendance acyclique.
AnalysisPresenter construit le **ChartPresenter**.

2/ Mettre les classes **Chart1D** et **Chart2D** en private à l'intérieur du module

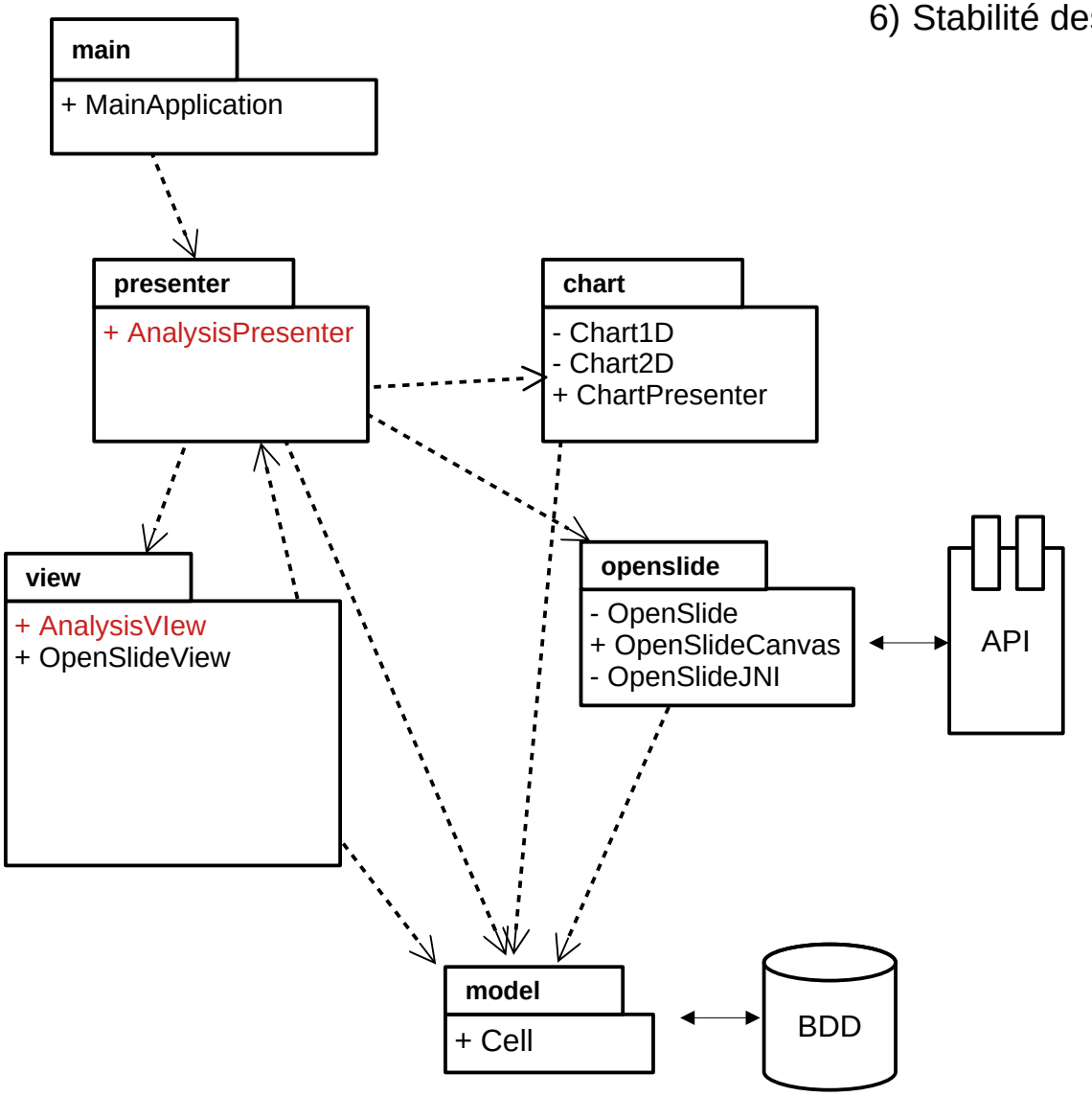
Principes

1) Équivalence réutilisation / livraison
2) Fermeture commune
3) Réutilisation commune
4) Dépendances acycliques
5) Relation dépendance / stabilité
6) Stabilité des abstractions



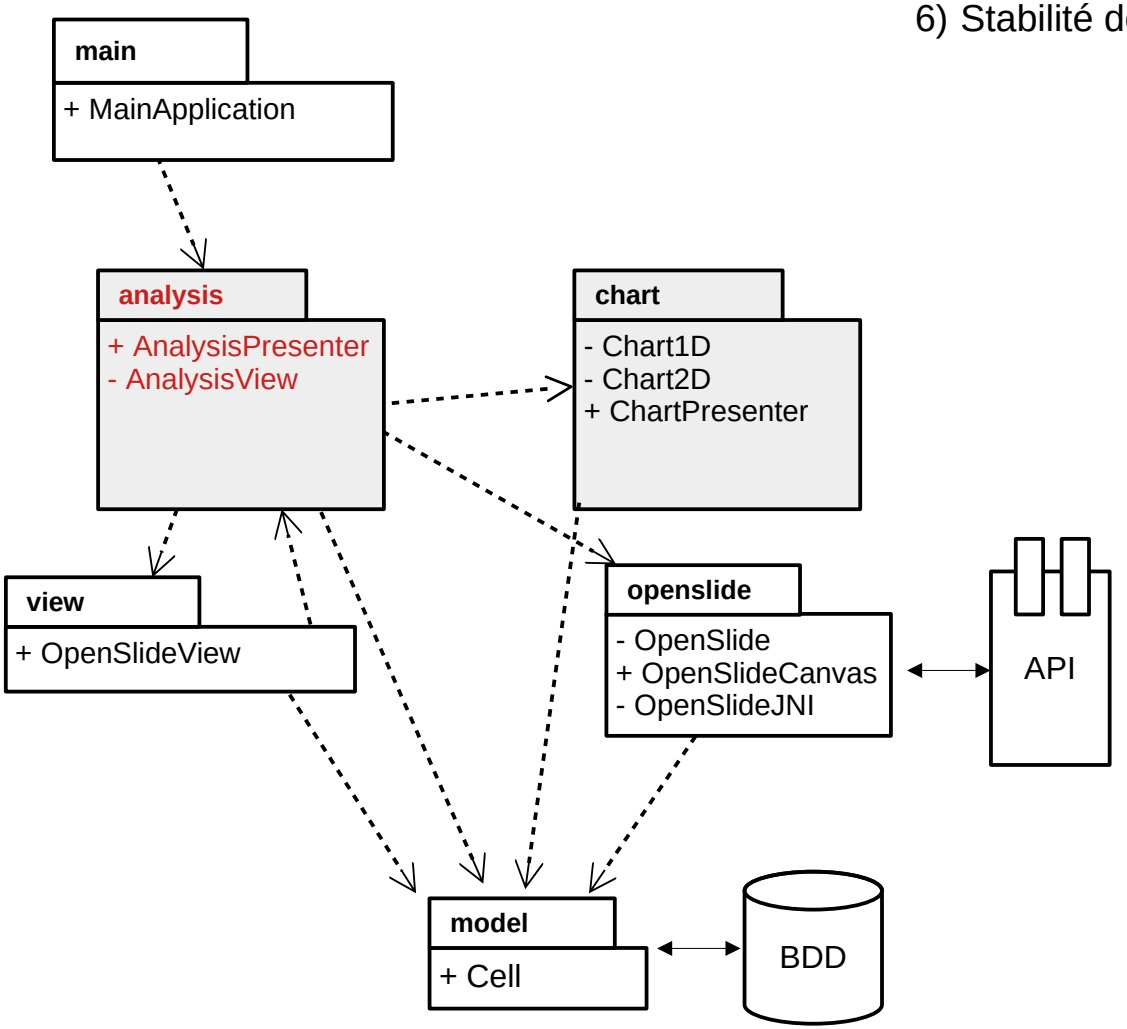
Problèmes
- Non respect du principe de fermeture dans le paquet **view** (ie, responsabilité unique)
- Non respect de la réutilisation commune entre la présentation et la vue de « analysis ».

Principes
1) Équivalence réutilisation / livraison
2) Fermeture commune
3) Réutilisation commune
4) Dépendances acycliques
5) Relation dépendance / stabilité
6) Stabilité des abstractions



Solution
1/ Regrouper le présentateur et la vue dans le même paquet.
2/ Renommer le paquet « **presenter** » en « **analysis** ».

Principes
1) Équivalence réutilisation / livraison
2) Fermeture commune
3) Réutilisation commune
4) Dépendances acycliques
5) Relation dépendance / stabilité
6) Stabilité des abstractions

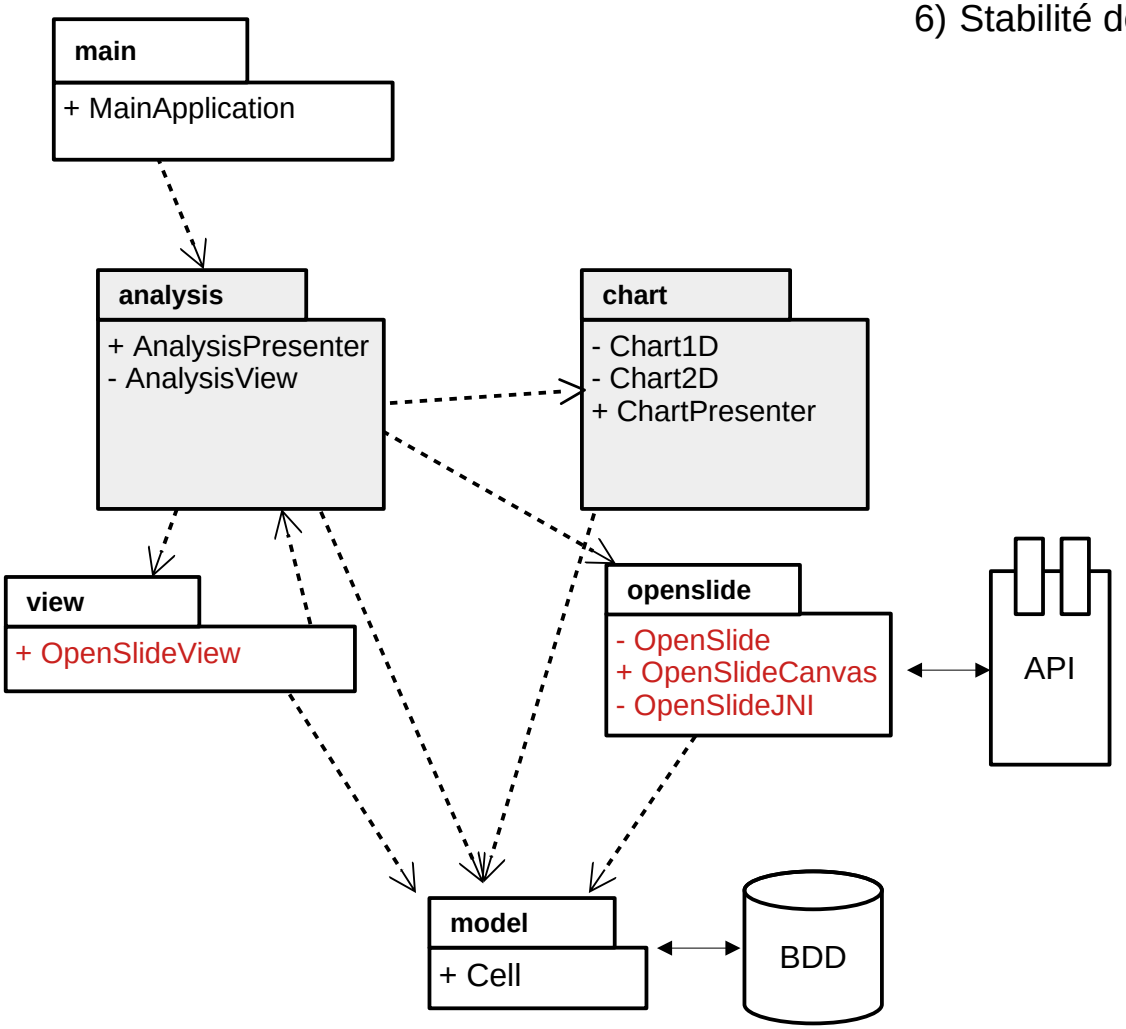


Problèmes

1/ OpenSlide est éclaté dans plusieurs paquets (réutilisation commune).
2/ OpenSlide est une bibliothèque externe qui est amenée à évoluer. Il doit être possible de changer de bibliothèque (équivalence réutilisation / livraison).

Principes

1) Équivalence réutilisation / livraison
2) Fermeture commune
3) Réutilisation commune
4) Dépendances acycliques
5) Relation dépendance / stabilité
6) Stabilité des abstractions



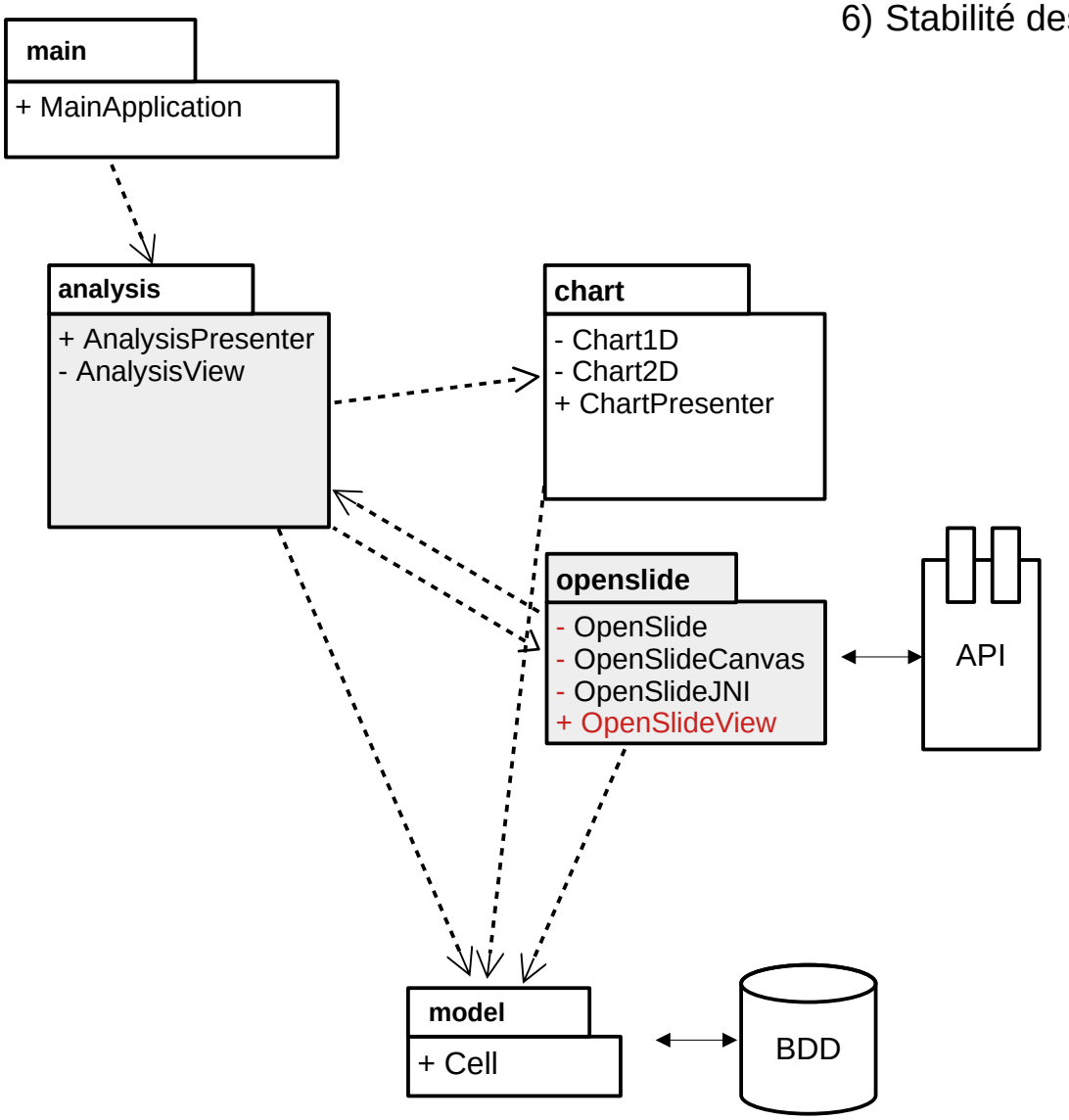
Solution

1/ Regrouper les classes OpenSlide dans un paquet unique

2/ Changer les visibilités

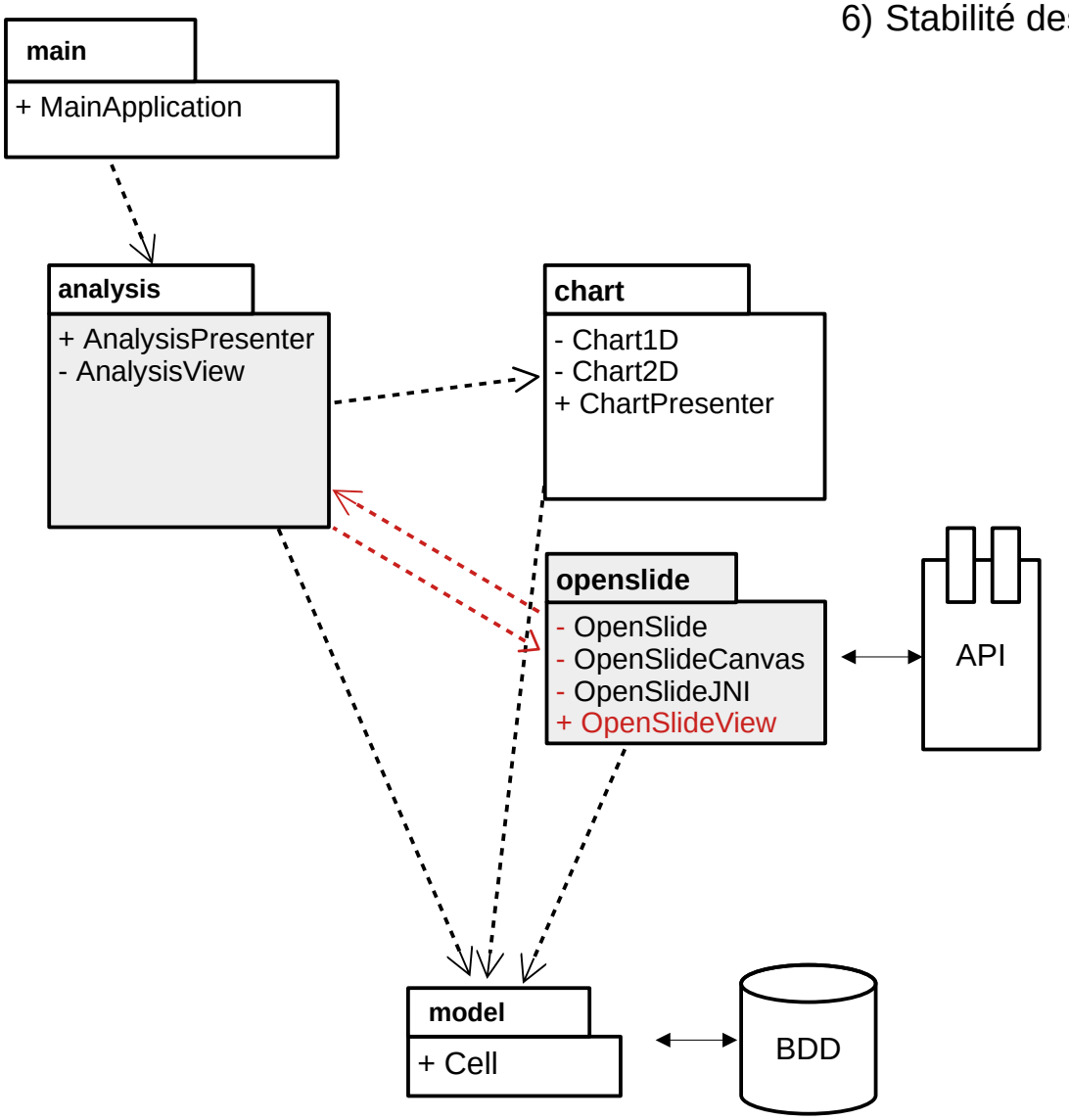
Principes

1) Équivalence réutilisation / livraison
2) Fermeture commune
3) Réutilisation commune
4) Dépendances acycliques
5) Relation dépendance / stabilité
6) Stabilité des abstractions



Problème
1/ Dépendance acyclique

Principes
1) Équivalence réutilisation / livraison
2) Fermeture commune
3) Réutilisation commune
4) Dépendances acycliques
5) Relation dépendance / stabilité
6) Stabilité des abstractions

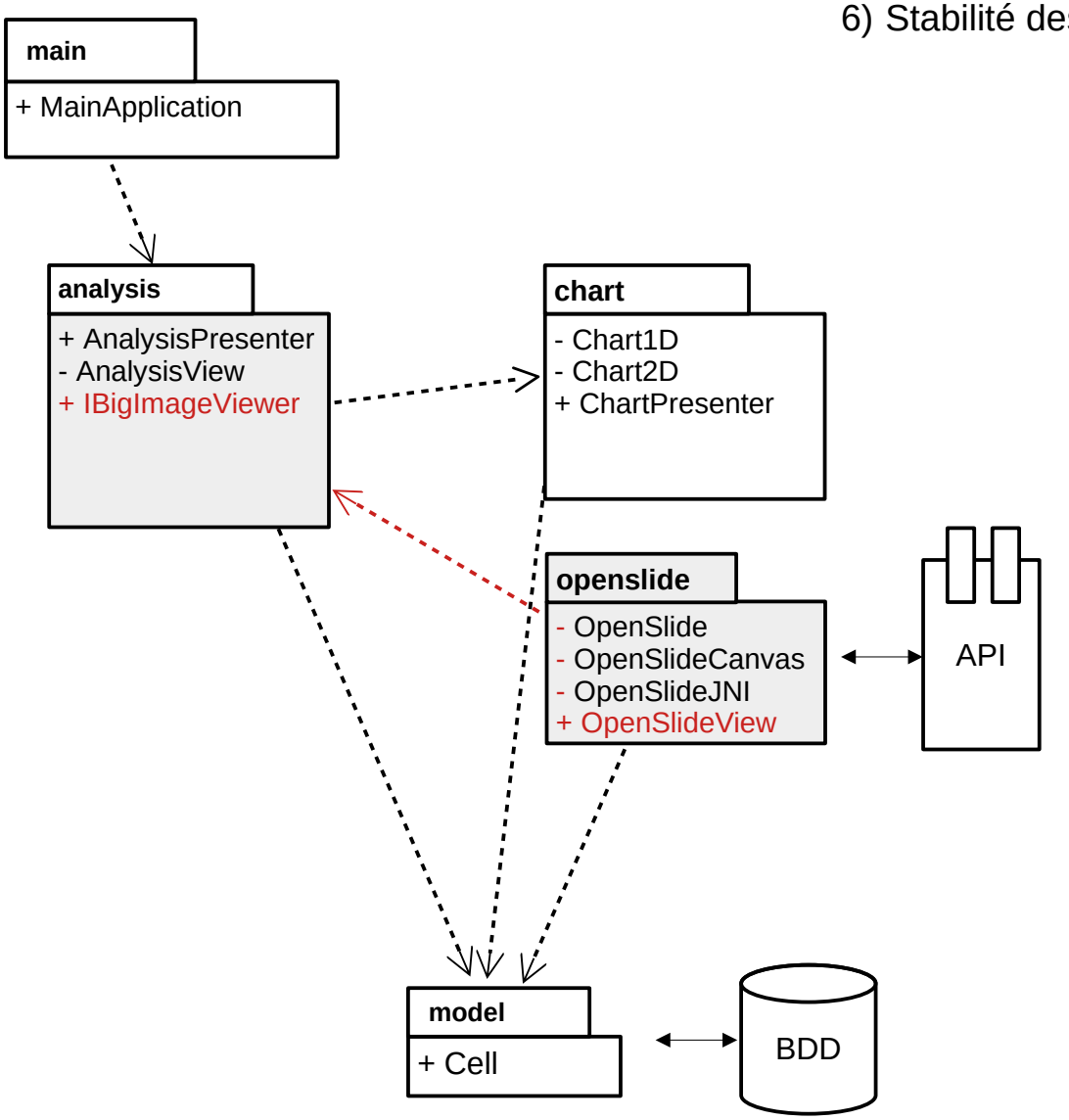


Solution

1/ Ajouter une interface **IBigImageViewer** dans la présentateur de l'analyse pour interfacer la bibliothèque.

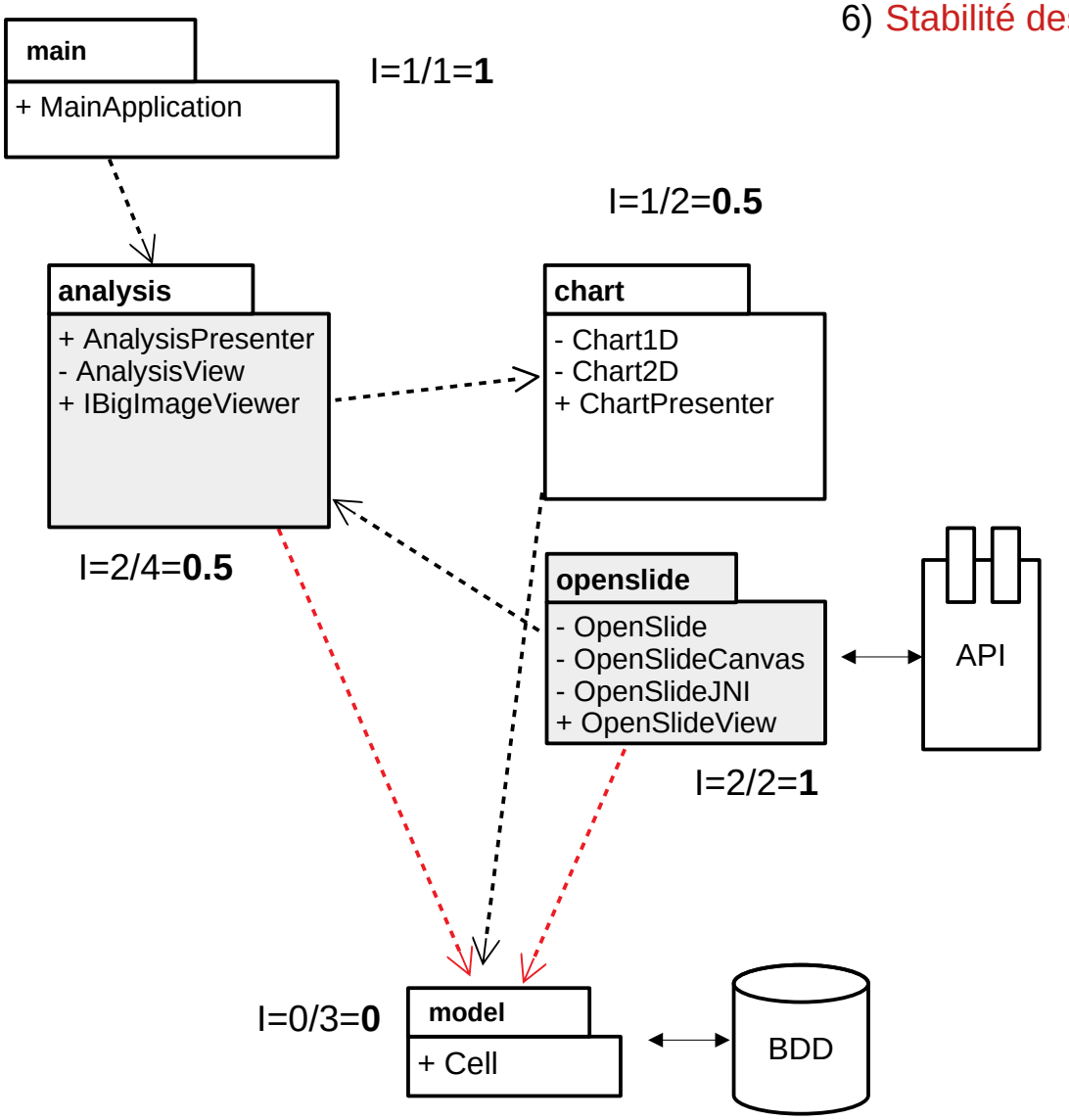
Principes

1) Équivalence réutilisation / livraison
2) Fermeture commune
3) Réutilisation commune
4) Dépendances acycliques
5) Relation dépendance / stabilité
6) Stabilité des abstractions



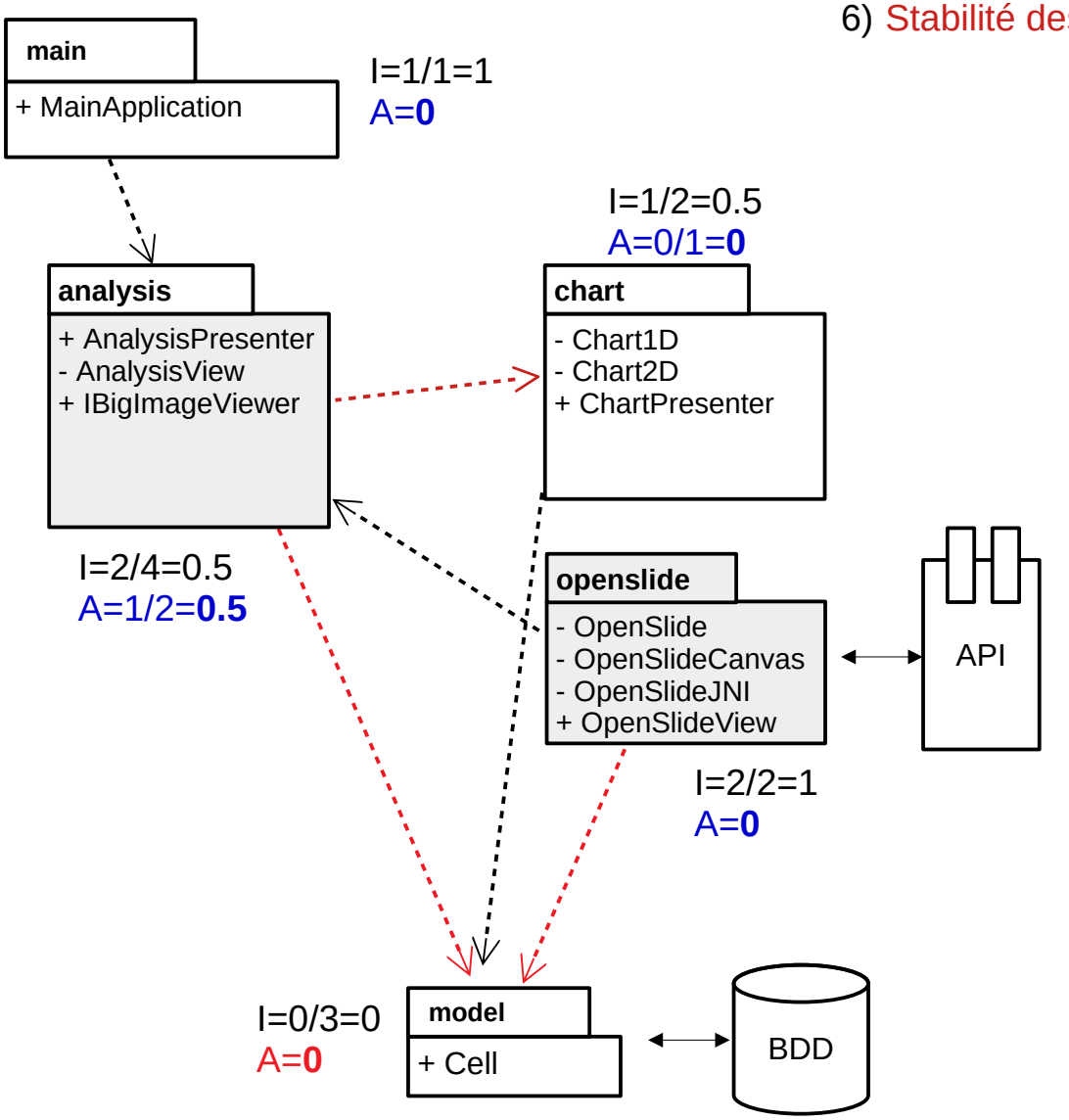
Problème
- Relation dépendance / stabilité + stabilité des abstractions

Principes
1) Équivalence réutilisation / livraison
2) Fermeture commune
3) Réutilisation commune
4) Dépendances acycliques
5) Relation dépendance / stabilité
6) Stabilité des abstractions



Problème
- Relation dépendance / stabilité + stabilité des abstractions

Principes
1) Équivalence réutilisation / livraison
2) Fermeture commune
3) Réutilisation commune
4) Dépendances acycliques
5) Relation dépendance / stabilité
6) Stabilité des abstractions



Principes
1) Équivalence réutilisation / livraison
2) Fermeture commune
3) Réutilisation commune
4) Dépendances acycliques
5) Relation dépendance / stabilité
6) Stabilité des abstractions

Solution
1/ Ajout d'une interface « **AbstractCell** » dans le modèle.