REX : entamer une migration

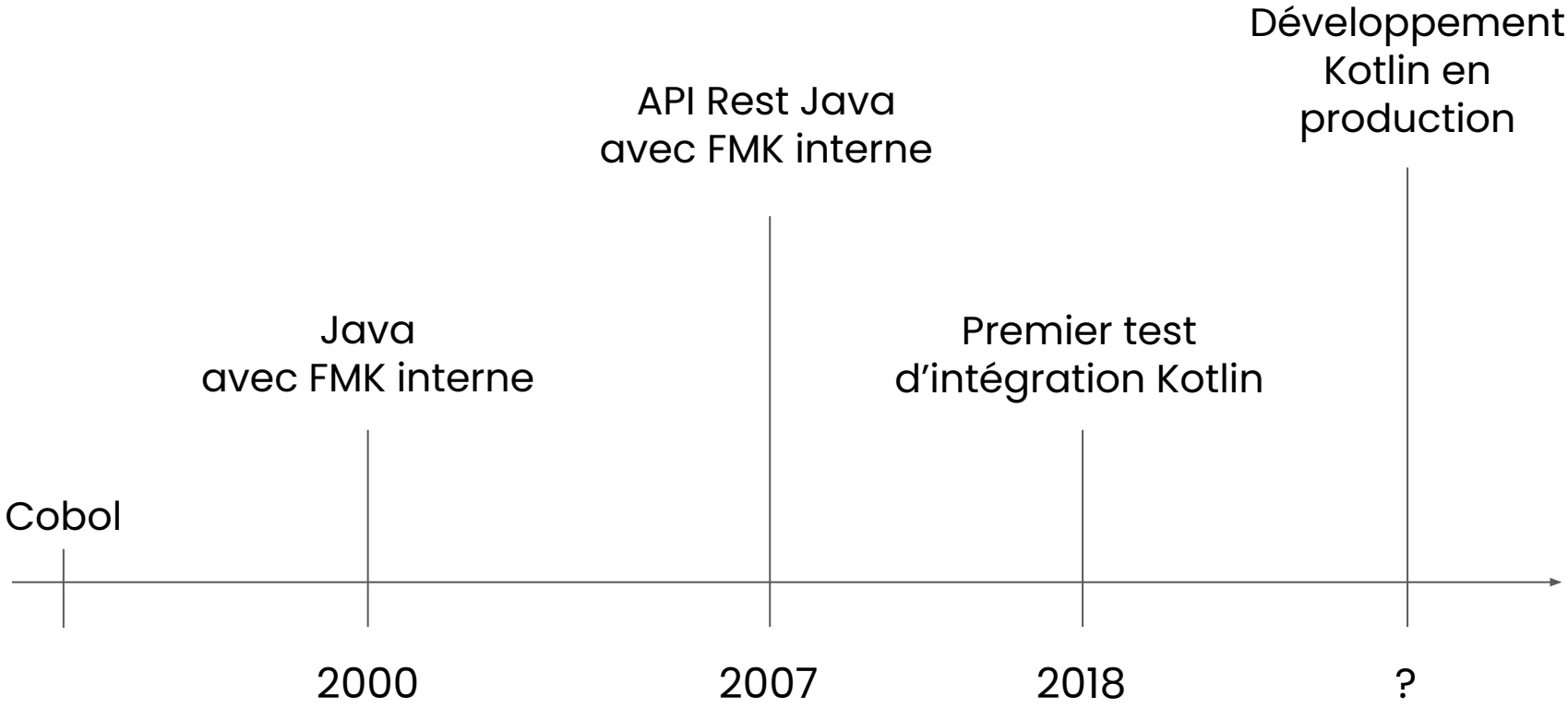
Kotlin dans une DSI Java

pôle emploi

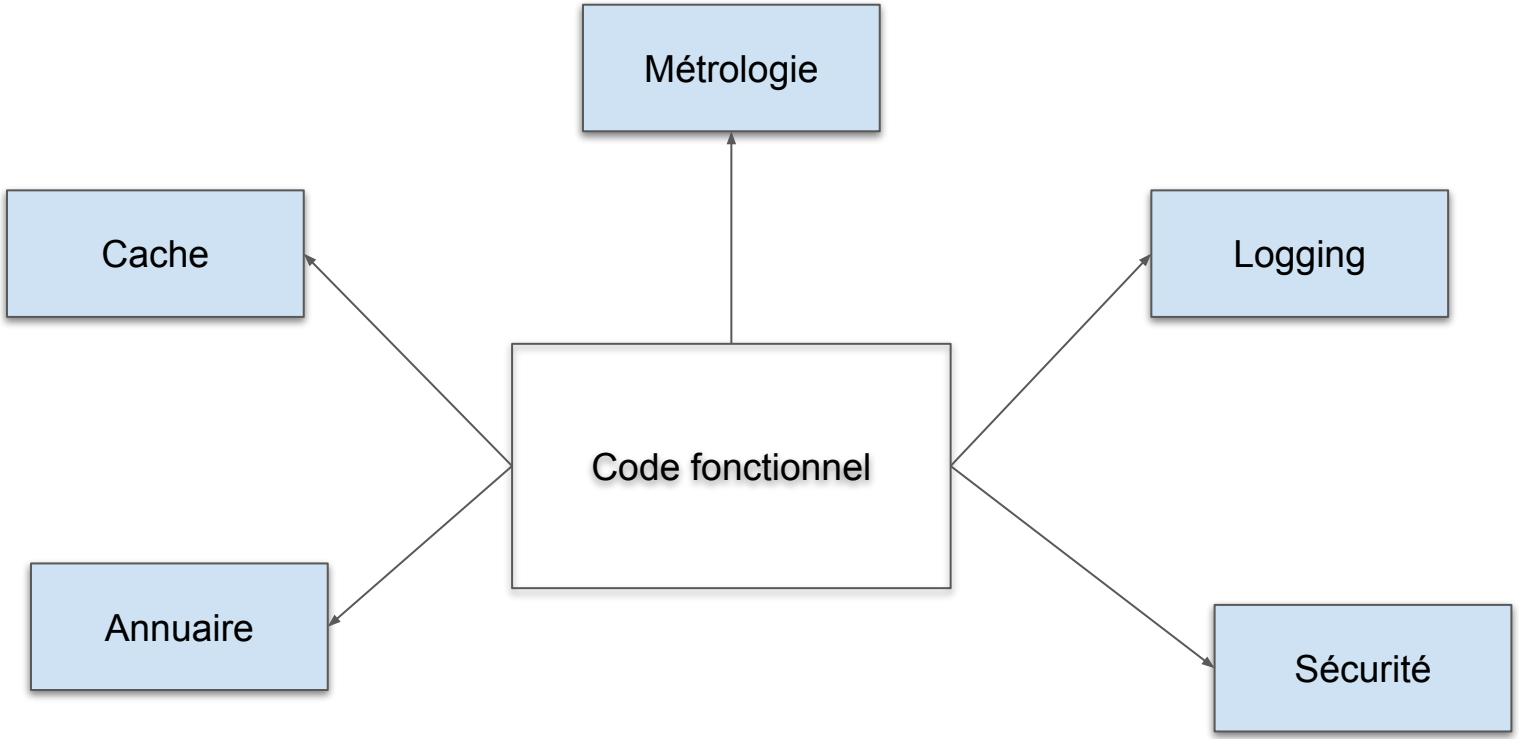
JPhi_Baconnais

Tech Lead à la DSI de Pôle emploi

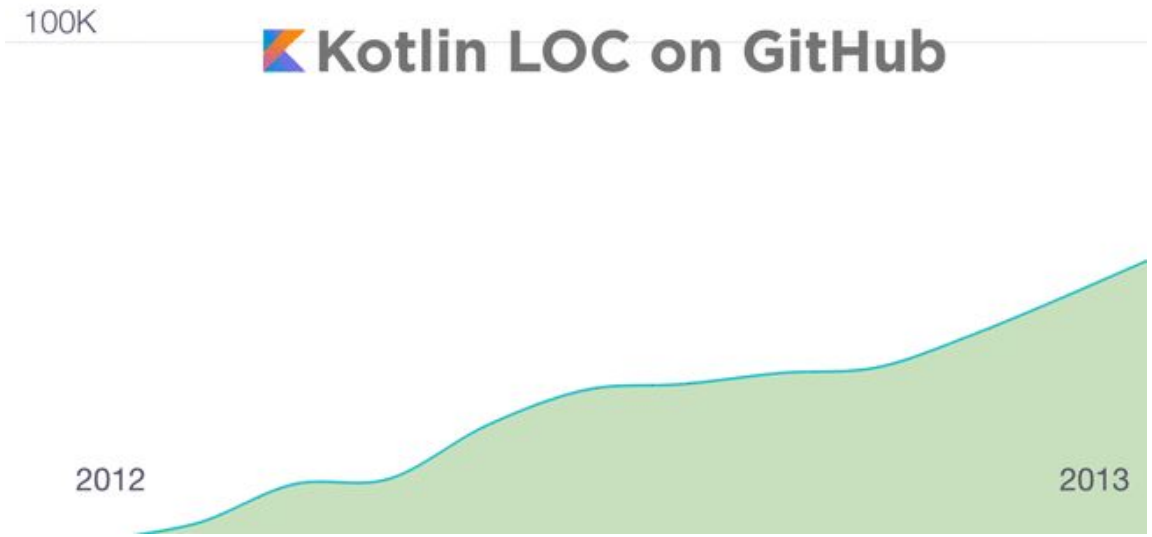
Contexte de la DSI Pôle emploi


Développement
Kotlin en
production
API Rest Java
avec FMK interne
Java
avec FMK interne
Premier test
d'intégration Kotlin
Cobol
2000
2007
2018
?

Notre framework interne


Métrologie
Cache
Logging
Code fonctionnel
Annuaire
Sécurité

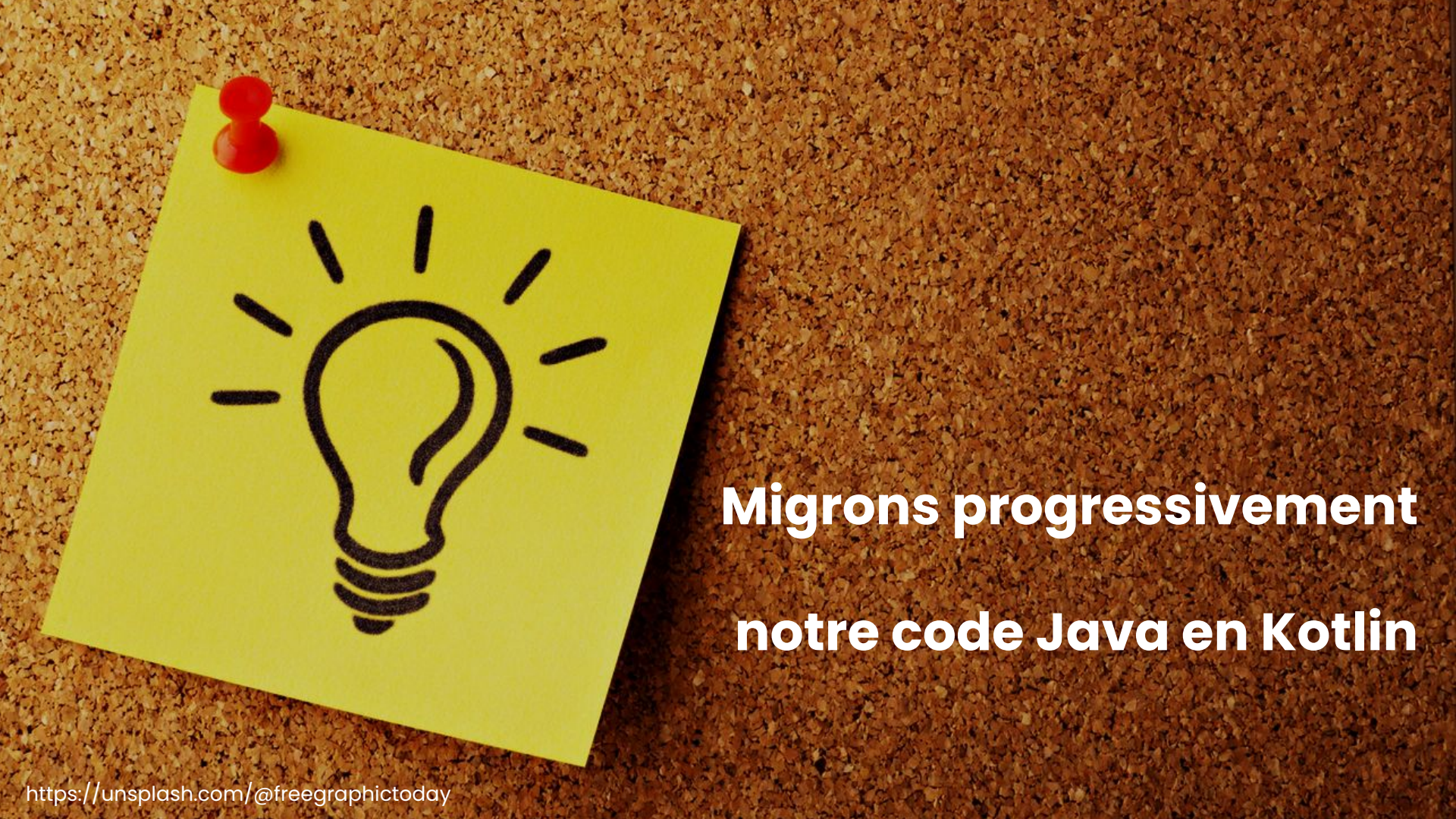
Le Kotlin dans tout ça ?


100K
Kotlin LOC on GitHub
2012
2013

“ Les nouveaux dév seront fait en Kotlin ! ”

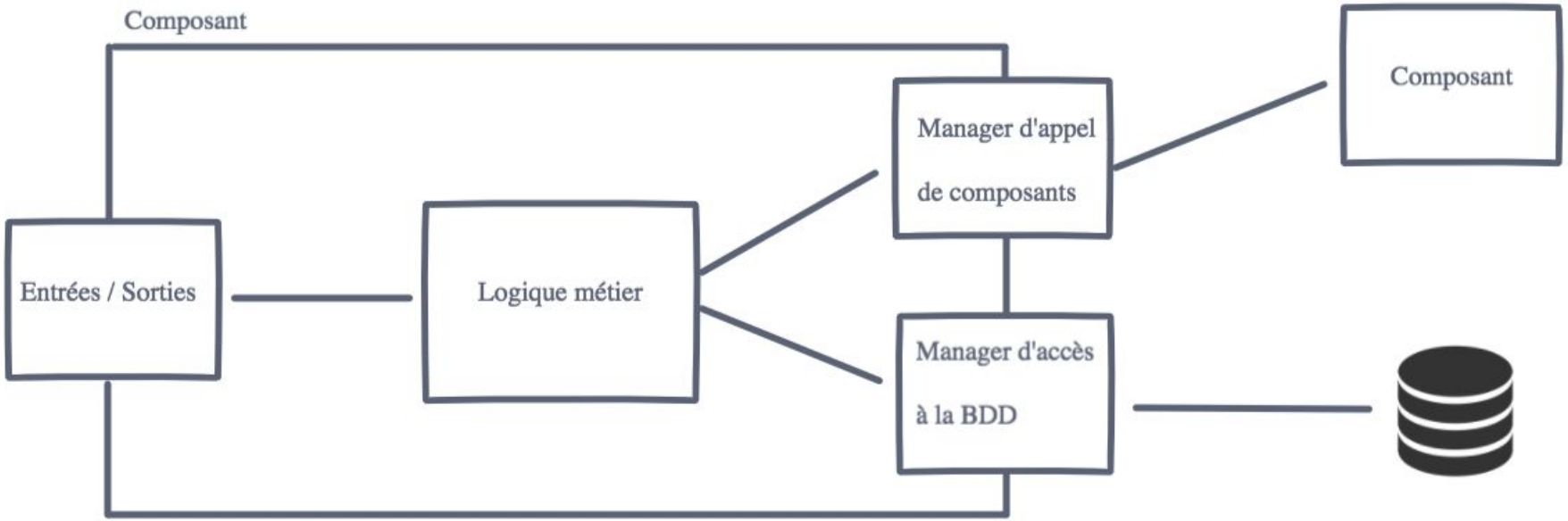
https://unsplash.com/@kaip

L’infrastructure est trop complexe et créer de nouveaux composants en Kotlin serait compliqué !

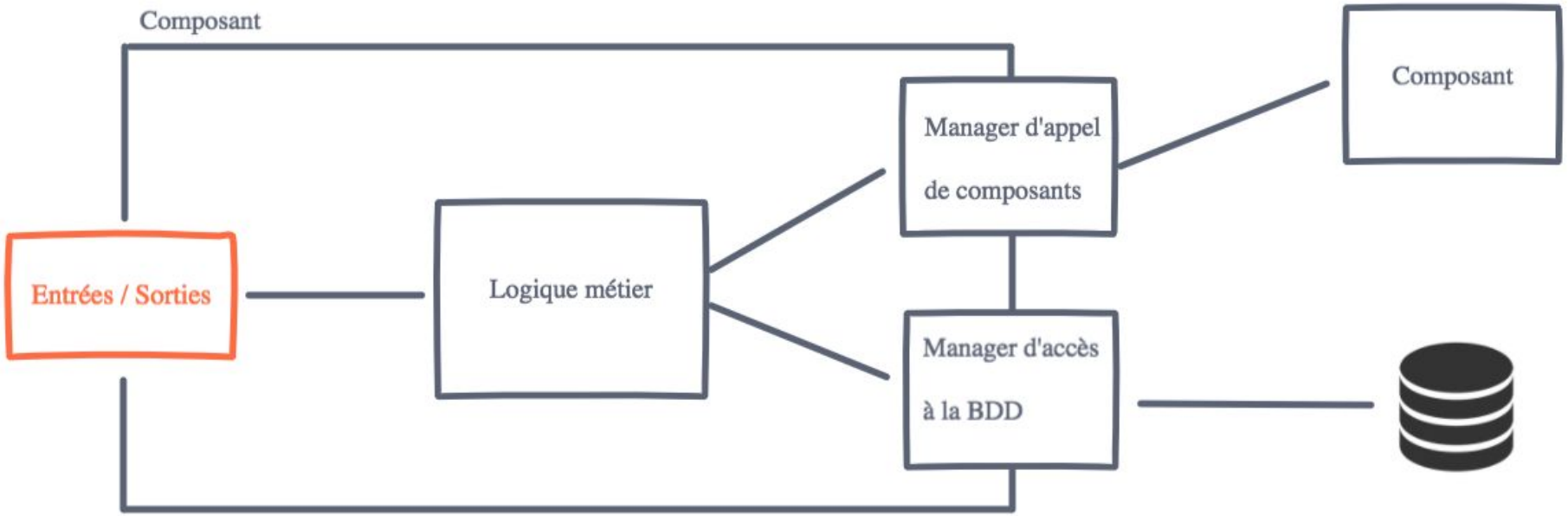

Migrons progressivement
notre code Java en Kotlin
https://unsplash.com/@freegraphictoday

L’intégration de Kotlin dans notre éco-système

Architecture de notre code fonctionnel


Composant
Entrées / Sorties
Logique métier
Manager d'appel
de composants
Manager d'accès
à la BDD
Composant

Choix de migration


Composant
Composant
Manager d'appel
de composants
Entrées / Sorties
Logique métier
Manager d'accès
à la BDD

```
public class Person implements
Cloneable{

  private String name;

  private Integer id;

  public Person(String name,Integer
id){
    this.name = name;
    this.id = id;
  }
  public String getName(){
    return name;
  }

  public void setName(String name){
    this.name = name;
  }

  public Integer getId(){
    return id;
  }

  public void setId(Integer id){
    this.id = id;
  }
```

```
  @Override
  public boolean equals(Object obj) {
    if (obj == this) {
      return true;
    }
    if (obj == null || obj.getClass()
!= this.getClass()) {
      return false;
    }
    Person guest = (Person) obj;
    return id == guest.id && (name ==
guest.getName || (name != null &&
name.equals(guest.getName())));
  }

  @Override
  public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((name
== null) ? 0 : name.hashCode());
    result = prime * result + id;
    return result;
  }
```

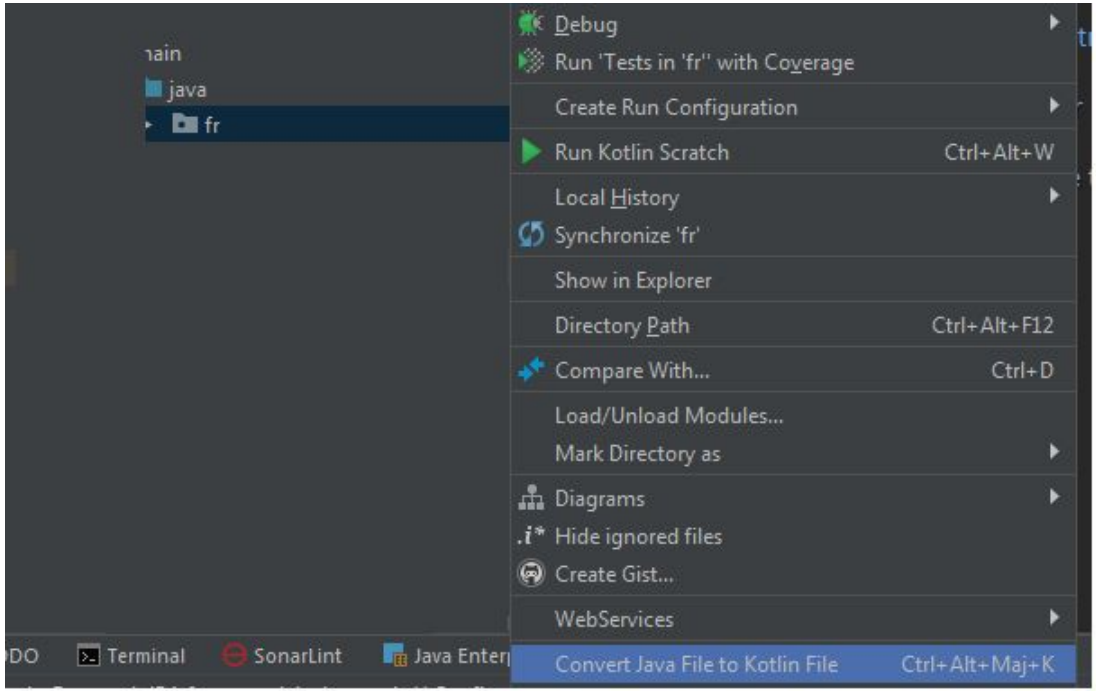
```
 @Override
  public String toString(){

return"Person(name="+name+",id="+id")
";
  }

  @Override
  public Object clone(){
      Person person = null;
          try {
             person = (Person)
super.clone();
          }
catch(CloneNotSupportedException
cnse) {

cnse.printStackTrace(System.err);
          }
          person.name = (String)
name.clone();
          person.id = (Integer)
id.clone();
          return person;
  }
}
```

```
data class User(val name: String, val id: Int)
```

Les étapes de modification de notre composant

1 / Installer le plugin Kotlin dans l’IDE

2 / Clic droit -> convertir en kotlin



3 / Déplacer les sources dans un package Kotlin

4 / Ajout de la dépendance Kotlin

```
<kotlin.version>1.3.21</kotlin.version>


<dependency>
  <groupId>org.jetbrains.kotlin</groupId>
  <artifactId>kotlin-stdlib-jdk8</artifactId>
  <version>${kotlin.version}</version>
</dependency>
```

5 / Déclaration ressources Kotlin

```
<groupId>org.jetbrains.kotlin</groupId>
<artifactId>kotlin-maven-plugin</artifactId>
<version>${kotlin.version}</version>
[...]
    <configuration>
        <sourceDirs>
            <source>src/main/java</source>
            <source>src/main/kotlin</source>
            <source>src/main/resources</source>
        </sourceDirs>
    </configuration>
```

```
<groupId>org.jetbrains.kotlin</groupId>
<artifactId>kotlin-maven-plugin</artifactId>
<version>${kotlin.version}</version>
[...]
    <configuration>
        <sourceDirs>
            <source>src/main/java</source>
            <source>src/main/kotlin</source>
            <source>src/main/resources</source>
        </sourceDirs>
    </configuration>
```

6 / Quelques ajustements dans le code

Erreur de compilation : Property must be initialized or be abstract

```
class ConstantesServices {
   companion object {
       val MEDIA TYPE UTILISE:MediaType! =
MediaType.APPLICATION_JSON_TYPE.withCharset(CharEncoding.UTF_8)
   }
}
```

```
class ConstantesServices {

   val MEDIA TYPE UTILISE: MediaType =
MediaType.APPLICATION_JSON_TYPE.withCharset(CharEncoding.UTF_8)

}
```

```
class ConstantesServices {
   companion object {
       val MEDIA TYPE UTILISE:MediaType! =
MediaType.APPLICATION_JSON_TYPE.withCharset(CharEncoding.UTF_8)
   }
}
```

```
class ConstantesServices {

   val MEDIA TYPE UTILISE: MediaType =
MediaType.APPLICATION_JSON_TYPE.withCharset(CharEncoding.UTF_8)

}
```

7 / Les tests

Mockito 1.9 ⇒ 2

Whitebox

```
org.mockito.exceptions.base.MockitoException: Cannot mock/spy class
data.MiniSite
Mockito cannot mock/spy because :
 - final class
```

mock-maker-inline

Et ça marche !

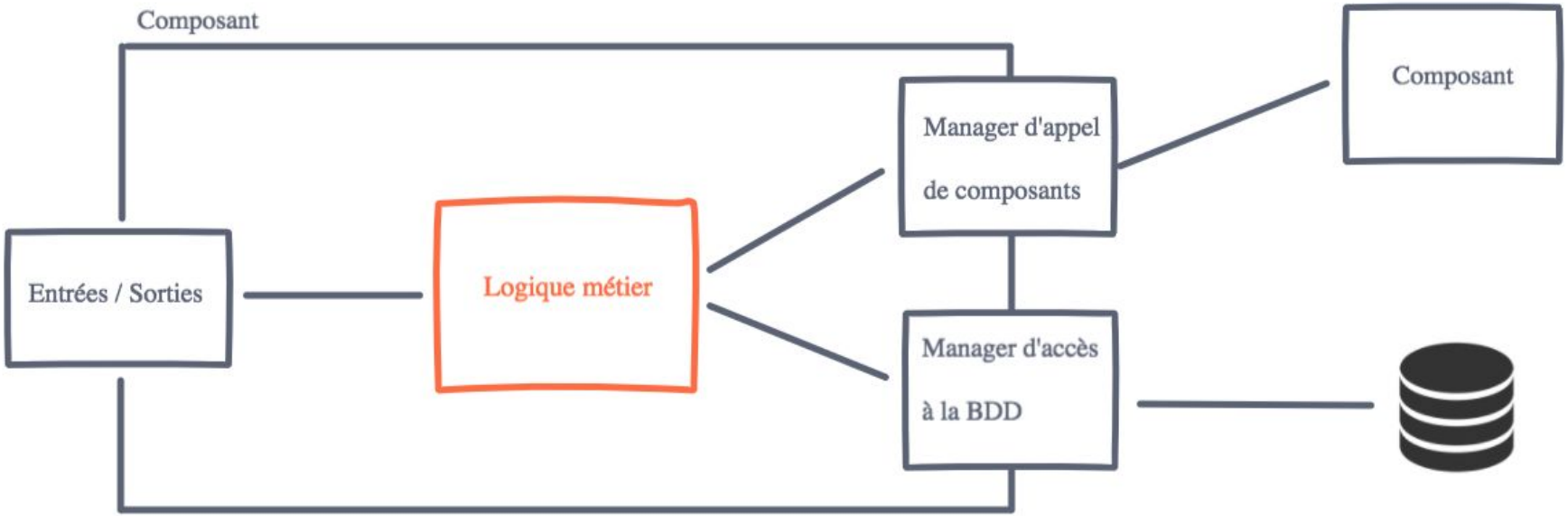
Migration réussie

Le composant fonctionne sans régression

Plus value pas très haute

Mais du Kotlin a été intégré !

La suite ?


Composant
Entrées / Sorties
Logique métier
Manager d'appel
de composants
Manager d'accès
à la BDD
Composant

Faire des tests Kotlin


MockK

```
@Test
fun givenServiceMock_whenCallingMockedMethod_thenCorrectlyVerified() {
    // given
    val service = mockk<TestableService>()
    every { service.getDataFromDb("Expected Param") } returns "Expected
Output"

    // when
    val result = service.getDataFromDb("Expected Param")

    // then
    verify { service.getDataFromDb("Expected Param") }
    assertEquals("Expected Output", result)
}
```

Bencher le composant

Installation en production !

https://unsplash.com/@doran_erickso
n

Merci !

JPhi_Baconnais