

Joseph
ARUL

Jean-Prince
DOTOU-SEGLA

AUTOMATISATION NOCODE AVEC LE
FRAMEWORK WEBENGINE

JOURNÉE
FRANÇAISE DES
TESTS
LOGICIELS



11 JUIN 2024
BEFFROI DE MONTROUGE



Automatisation NoCode avec le Framework WebEngine

1. Pourquoi un Framework pour les tests automatisés ?
2. Le Framework WebEngine : Standard, pérenne, open source et intégrant différentes approches de test
3. WebEngine NoCode
 - a) Persona & Fonctionnement
 - b) Démo
 - c) Déploiement & Usages
 - d) Perspectives

1. Pourquoi un Framework pour les tests automatisés?

Comment une entreprise tend vers l'efficience DevOps ?

Accelerate : Par où commencer

[Rapport sur l'état du DevOps en 2022](#) | [Google Cloud](#)

1

Contrôle de version

2

Intégration continue

Déploiement
continue

Automatisation des
déploiements

Monitoring et
observabilité

3

Automatisation des
tests

Cloud Nord:
[Jean-Rémy REVY -
Practice Manager -
Ippon Technologies](#)
[| LinkedIn](#)

L'automatisation

Facteurs de succès

Le succès d'un projet **global Entreprise** d'automatisation de tests dépend de :

1

Framework

- **Démarrer** facilement
- **Maintenir** facilement
- Se **concentrer** sur le fonctionnel
- Accès et **modification** aisée des **JDD**



2

Approches adoptées

Selon:

- **Organisation**
- **Compétences**
- **Testabilité** de l'application



Pourquoi un framework?

Constat chez AXA avant 2019



- ❑ L'organisation
 - Contexte métier différent (tribu, squad)
 - Automates avec structures et pratiques différentes
 - Appétences techniques diverses
- ❑ Les outils et langages
 - LeanFT
 - Choix de Sélénium (pour les enjeux de test mobiles entre autres) mais nécessite une surcote

Frameworks du marché*



- ❑ UFT Developer (ex: LeanFT)
 - Licences payantes
- ❑ Sélénium :
 - Gestion du driver
 - Gestion du rapport
 - Gestions multi-attributs
 - Gestion de l'accès aux JDD
- ❑ Outil/Infra Test NoCode payant et/ou dépendant de l'éditeur (Katalon, Virtuoso, Kalios)

Enjeux & Ambitions AXA



- ❑ Bonnes pratiques de dev
- ❑ Structure commune des automates
- ❑ CI/CD
- ❑ Augmenter la technicité du collectif
 - Comprendre le HTML/JAVASCRIPT/CSS
 - Conception de modules réutilisables
 - Intégration dans notre CI/CD

2. Framework WebEngine

Framework AXA : WebEngine

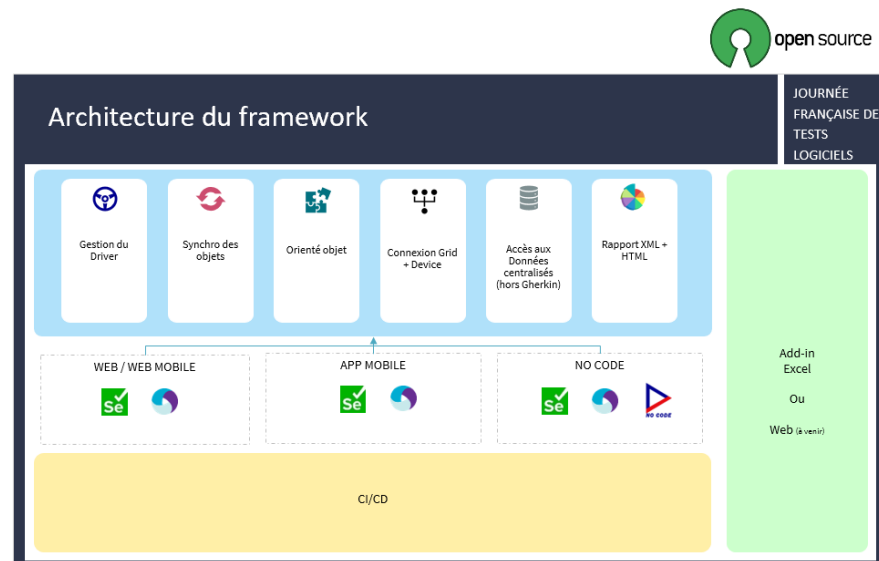
✓ Besoins couverts:

- ❑ **Centralisation** (Gestion des JDD, Rapport, CI/CD)
- ❑ **Uniformisation**
- ❑ **Meilleure synergie** entre automaticiens
- ❑ **Maintenance facilitée**

- ❑ Exécution sans IDE (**Orienté testeur non-automaticien**)

✓ Enjeux du WebEngine :

- ❑ Couvrir les **tests Web/Mobile et App**
- ❑ Solutions **open-source** (Sélénium WebDriver, Appium)
- ❑ **Bonnes pratiques** by design



- ❑ [AxaFrance/webengine-dotnet \(github.com\)](https://github.com/AxaFrance/webengine-dotnet) (.NET)
- ❑ [AxaFrance/webengine-java \(github.com\)](https://github.com/AxaFrance/webengine-java) (Java)

Les approches en lien avec notre organisation

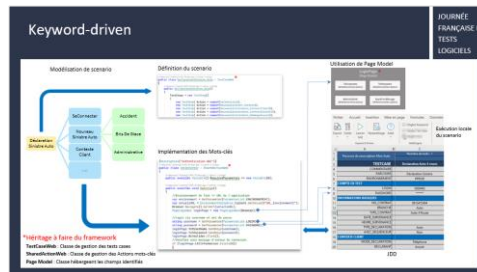
Ces approches sont prévues pour couvrir tous types de tests IHM (test local, d'acceptance ou parcours clés)

BDD*

```
1 Fonctionnalité: Servir un café
2 En tant qu'utilisateur
3 Je veux consommer un café
4 dont le prix fixe est de 40 centimes
5
6 Scénario: Servir un café court sans sucre quand je fais l'appoint
7 Etant donné que j'ai inséré 0,40 €
8 Quand je demande un "café court sans sucre"
9 Alors la machine me remplit un gobelet de "café court sans sucre"
```

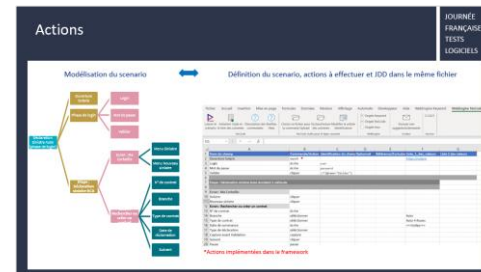
```
01 [Binding]
02 public class ServirUnCafeSteps
03 {
04     private decimal amount;
05     private string drinkName;
06
07     [Given(@"que j'ai inséré (.*) €")]
08     public void SoitQueJaiinsere(decimal amount)
09     {
10         this.amount = amount;
11     }
12
13     [When(@"je demande un ""(.+)""")]
14     public void QuandJeDemandeUn(string drinkName)
15     {
16         this.drinkName = CoffeeManager.Fill(drinkName, amount);
17     }
18
19     [Then(@"la machine me remplit un gobelet de ""(.+)""")]
20     public void AlorsLaMachineMeRemplitUnGobeletDe(string drinkName)
21     {
22         Assert.AreEqual(drinkName, this.drinkName);
23     }
24 }
```

Keyword-driven*



- Test Suite
→ n Tests cases
→ n Tests Steps
→ n Actions (mots-clés)

Actions



- Scénario
→ n Steps
→ n Actions



Automaticien(ne) dédié(é)



Test ingénieur / PO / BA

*BDD : Behaviour Driven Développement / Piloté par le comportement

*Keyword-driven : piloté par mots clés

3. WebEngine NoCode

WebEngine NoCode

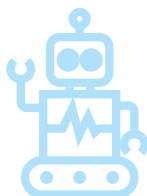
Persona & Fonctionnement

Description du scenario

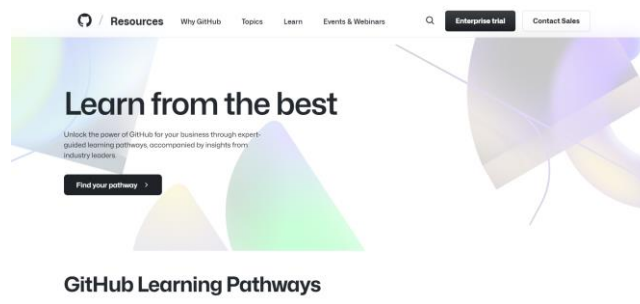
+
JDD



Testeurs / BA / PO



Sélénium Java



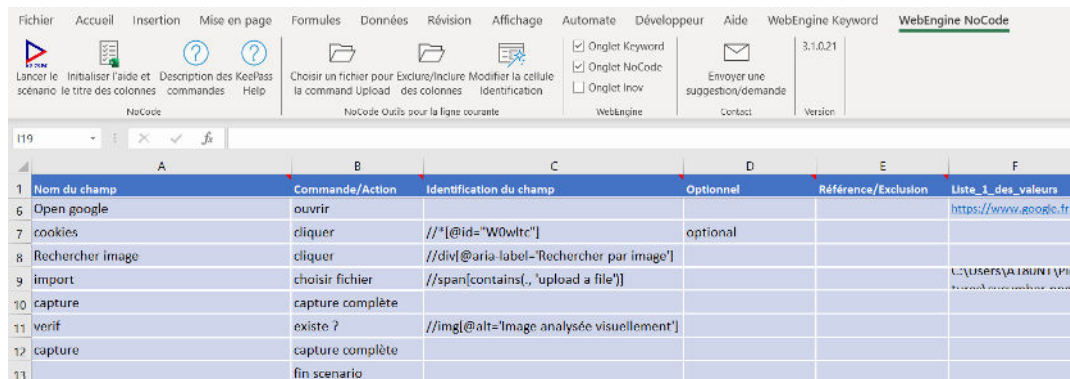
GitHub Learning Pathways

WebEngine NoCode

Démo sur un site/application (15 min)

Déroulé

- ❑ Développer un scénario avec le NoCode
 - Définir mon scénario (étape + résultat attendu + JDD)
 - Récupération des identifiants
 - Renseigner les différentes étapes Cf tableau
 - Exécuter
- ❑ Analyser le rapport généré
- ❑ *Intégration dans notre CI/CD**



The screenshot shows the WebEngine NoCode interface. At the top is a menu bar with options: Fichier, Accueil, Insertion, Mise en page, Formules, Données, Révision, Affichage, Automate, Développeur, Aide, WebEngine Keyword, and WebEngine NoCode. Below the menu bar is a toolbar with icons for launching the scenario, initializing help, and description of commands. The main area contains a table with the following data:

	A	B	C	D	E	F
1	Nom du champ	Commande/Action	Identification du champ	Optionnel	Référence/Exclusion	Liste_1_des_valeurs
6	Open google	ouvrir				https://www.google.fr
7	cookies	cliquer	//*[@id="W0wlte"]	optional		
8	Rechercher image	cliquer	//div[@aria-label="Rechercher par image"]			
9	import	choisir fichier	//span[contains(., 'upload a file')]			C:\Users\A130N1\ync
10	capture	capture complète				
11	verif	existe ?	//img[@alt="Image analysée visuellement"]			
12	capture	capture complète				
13		fin scenario				

Déploiement



Organisation et rythme de delivery



 **Groupe de travail et POs**

 **DEVs & POs**

 **Testeurs / BA / PO**



Apports, atteinte de nos ambitions

- ❑ Temps et accessibilité :
 - Facilité de prise en main (1j de test manuel = 1 jour de développement du scénario automatisé)

- ❑ Technique :
 - Montée en compétence technique des utilisateurs
 - Changement de mindset sur la conception (modules réutilisables)

- ❑ Qualité :
 - Démultiplication du nombre de scénarios automatisés
 - Approche complémentaire et intégrée entre code et nocode (actions développées bycode pour le nocode)

Quelques chiffres

Pilote en S2 2022 :

- 100 scénarios en 3 mois sur 1 périmètre
- > 10 utilisateurs nocode
- 1 semaine gagnée sur une release de 9 semaines

Aujourd'hui :

- 80% testeurs acculturés (300 testeurs)
- 30% d'utilisateurs réguliers
- > 20 périmètres
- 92% de promoteurs et 0% de détracteurs

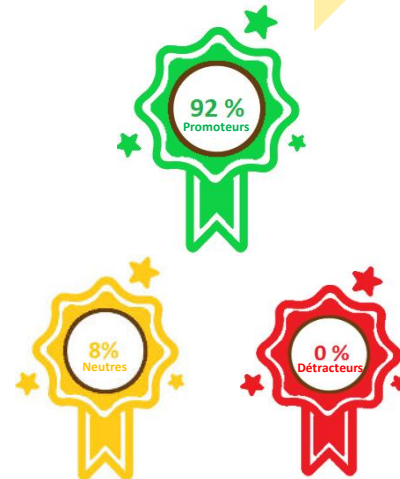
Ambition pour fin 2024

- > 50% d'utilisateurs réguliers
- > 40 périmètres
- Aide/Expertise de la communauté du test

Verbatims

Modulable
Fluidité
Assistant
Réutilisable
Rapidité
Collaboratif
Reporting
Installation
Simplicité
Clair
Familier
Reactivité
Ajustable

Customisable
Générique
Web
Interface
Identification
Logs
Plugin
Rapport



Framework



Identification des
champs par
libellés



Prise en compte
de l'Accessibilité

No Code

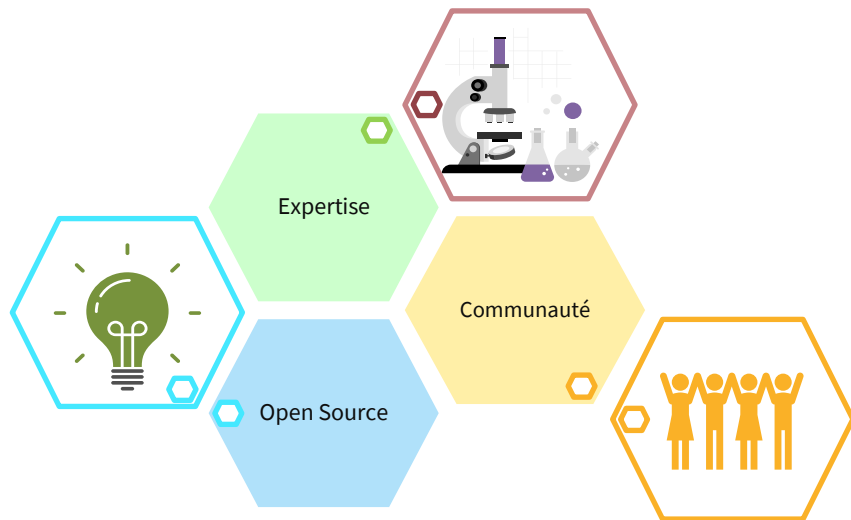


Version WEB



Appel API

On compte sur vous !



[Manuel Utilisateur + Installation](#)

Joseph
ARUL

Jean-Prince
DOTOU-SEGLA

AUTOMATISATION NOCODE AVEC LE
FRAMEWORK WEBENGINE

JOURNÉE
FRANÇAISE DES
TESTS
LOGICIELS



MERCI DE VOTRE ÉCOUTE
N'oubliez pas de voter



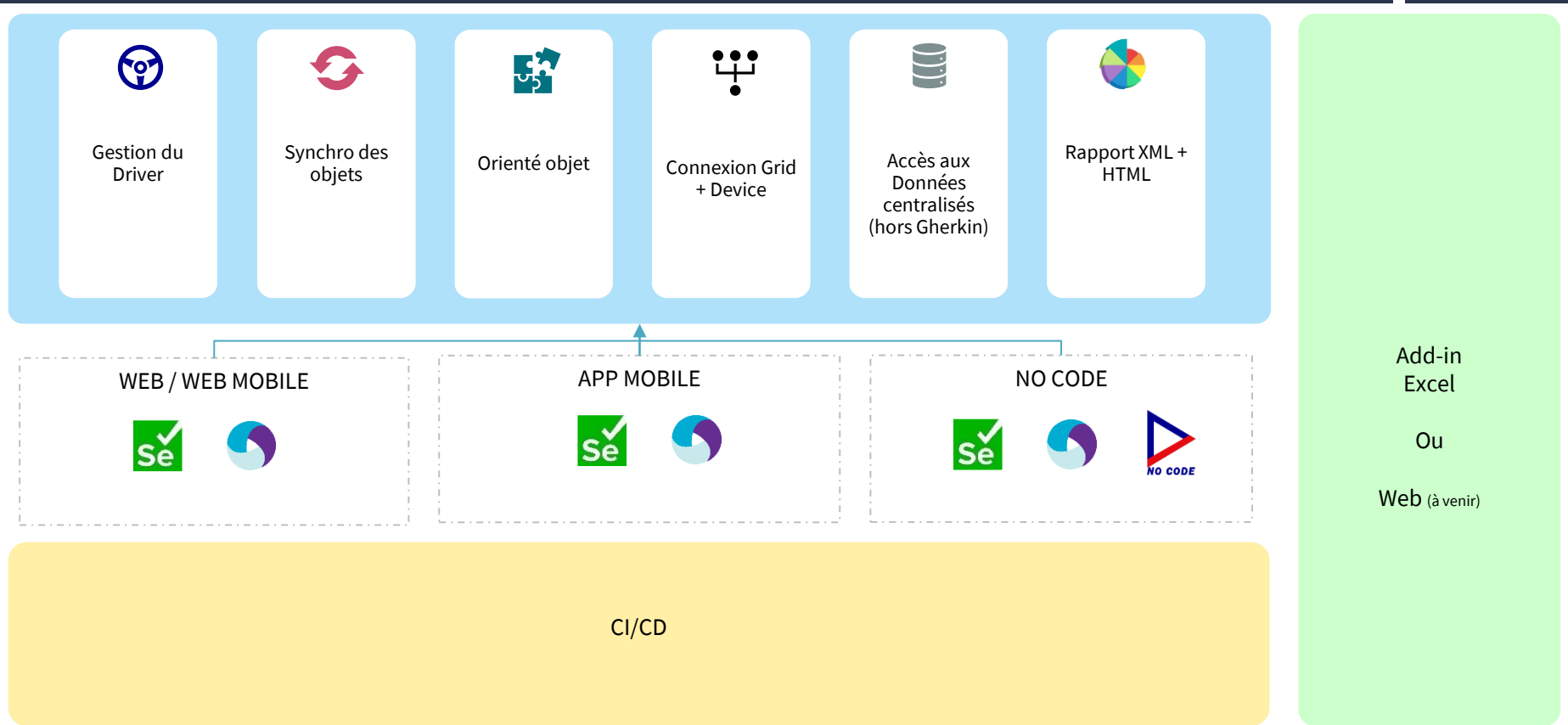
JOURNÉE
FRANÇAISE
DES TESTS
LOGICIELS

**VOTEZ POUR LA
MEILLEURE
PRÉSENTATION**



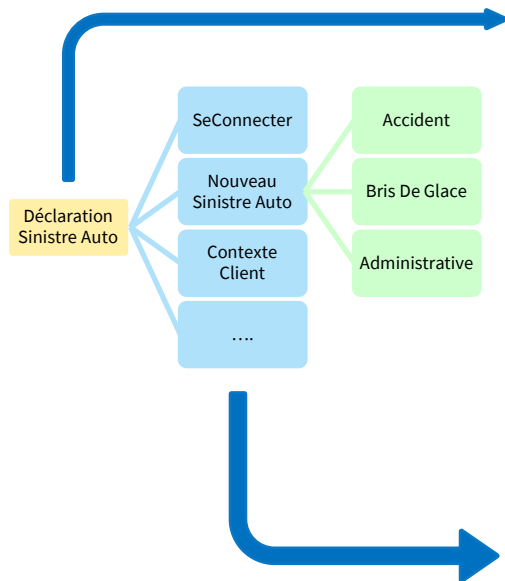
Annexes

Architecture du framework



Keyword-driven

Modélisation de scenario



Définition du scenario

```

2 references | Huaxing YUAN, 29 days ago | 1 author, 1 change
public class DeclarationSinistre_Auto : TestCaseWeb *
{
1 reference | Huaxing YUAN, 29 days ago | 1 author, 1 change
public DeclarationSinistre_Auto()
{
    TestSteps = new TestStep[]
    {
        new TestStep{ Action = nameof(SeConnecter)},
        new TestStep{ Action = nameof(NouveauSinistre_Contraire)},
        new TestStep{ Action = nameof(NouveauSinistreAuto_ContextClient)},
        new TestStep{ Action = nameof(NouveauSinistreAuto_ContextSinistre)},
        new TestStep{ Action = nameof(NouveauSinistreAuto_DommageAssure)},
    }
}
}
  
```

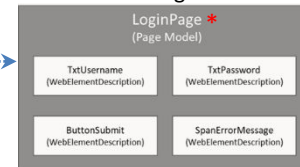
Implémentation des Mots-clés

```

[Description("Authentication WAC")]
3 references | Huaxing YUAN, 29 days ago | 1 author, 1 change
public class SeConnecter : SharedActionWeb *
{
0 references | Huaxing YUAN, 29 days ago | 1 author, 1 change
public override Variable[] RequiredParameters => new Variable[0];

0 references | Huaxing YUAN, 29 days ago | 1 author, 1 change
public override void DoAction()
{
    //Environnement de Test => URL de l'application
    var environment = GetParameter(ParameterList.ENVIRONNEMENT);
    var solarisURL = EnvironmentVariables.Current.GetValue($"*URL_{environment}");
    Browser.Navigate().GoToUrl(solarisURL);
    PageLoginWac loginPage = new PageLoginWac(Browser);
    //Login via username et mot de passe.
    string username = GetParameter(ParameterList.LOGIN);
    string password = GetParameter(ParameterList.PASSWORD);
    loginPage.TxtUserName.SendKeys(username);
    loginPage.TxtPassword.SendKeys(password);
    loginPage.BtnValider.Click();
    //Verifier sans message d'erreur de connexion.
    if (loginPage.LblInformation.Exists(4))
    {
}
}
  
```

Utilisation de Page Model



Exécution locale
du scenario

	A	B	C
1	Parcours de souscription Mon Auto		Nombre de tests : 1
2		TESTCASE	Declaration Auto 4 roues
3		COMMENTAIRE	
4		PARCOURS	Declaration Sinistre
5		ENVIRONNEMENT	PPROD
6		COMPTE DE TEST	
7		LOGIN	S006465
8		PASSWORD	*****
9		INFORMATIONS BASIQUES	
10		NO_CONTRAT	9915472404
11		BRANCHE	Auto
12		TYPE_CONTRAT	Auto 4 Roues
13		DATE_SURVENANCE	
14		HEURE_SURVENANCE	
15		TYPE_DECLARATION	Auto
16		AVEC_SEQUENCEUR	Non
17		CONTEXTE CLIENT	
18		MODE_DECLARATION	Téléphone
19		DECLARANT	Assuré
20			

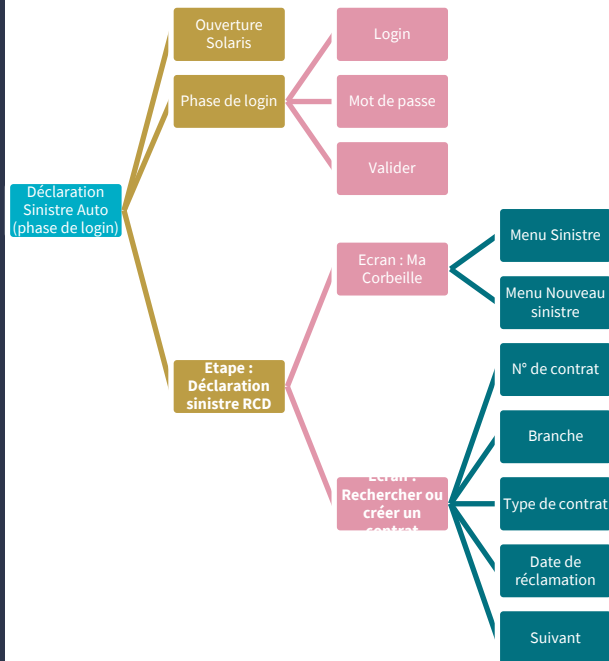
*Héritage à faire du framework

TestCaseWeb : Classe de gestion des tests cases

SharedActionWeb : Classe de gestion des Actions mots-clés

Page Model : Classe hébergeant les champs identifiés

Modélisation du scénario



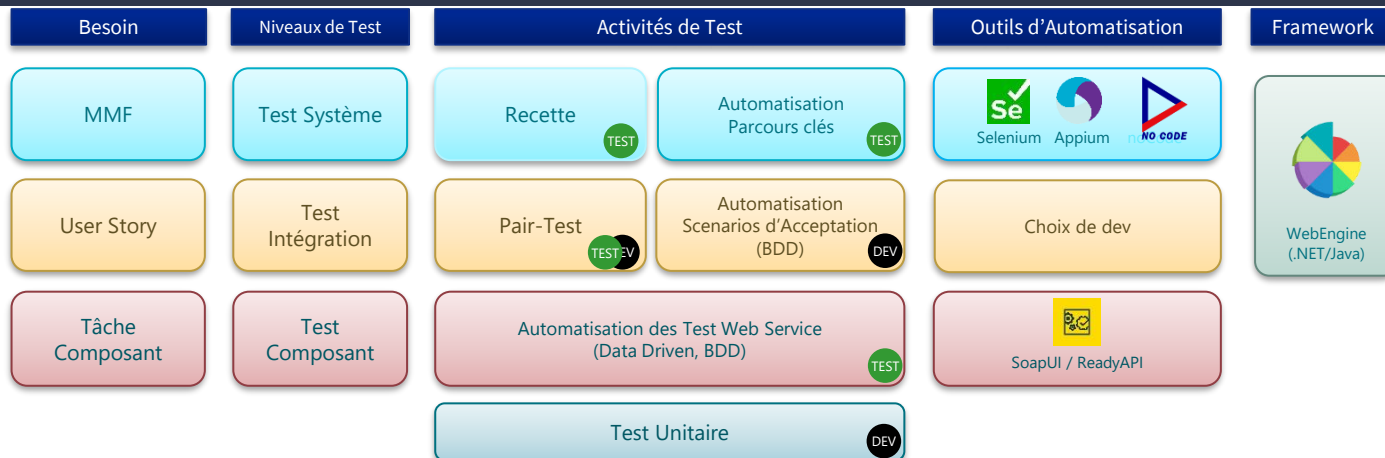
Définition du scénario, actions à effectuer et JDD dans le même fichier

WebEngine NoCode

Nom du champ	Commande/Action	Identification du champ	Optionnel	Référence/Exclusion	Liste 1 des valeurs	Liste 2 des valeurs
1 Ouverture Solaris	ouvrir *				https://solaris	
3 Login	écrire	user				
4 Mot de passe	écrire	password				
5 Valider	cliquer	//*[@name="Valider"]				
7 Etape : Déclaration sinistre Auto Accident 1 véhicule						
8						
9 Ecran : Ma Corbeille						
10 Sinistre	cliquer					
11 Nouveau sinistre	cliquer					
12 Ecran : Rechercher ou créer un contrat						
13 N° de contrat	écrire					
14 Branche	sélectionner				Auto	
15 Type de contrat	sélectionner				Auto 4 Roues	
16 Date de survenance	écrire				<<<{today}>>>	
17 Type de déclaration	sélectionner					
18 Capture avant Validation	capture					
19 Suivant	cliquer					
20 Pause	pause					

*Actions implémentées dans le framework

Stratégie globale d'automatisation de test



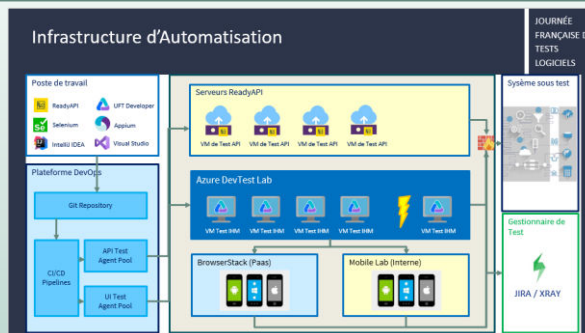
Méthodologies

- Script Linéaire (non recommandé)
- Gherkin (BDD)
- Piloté par Mot-clés
- Piloté par Action

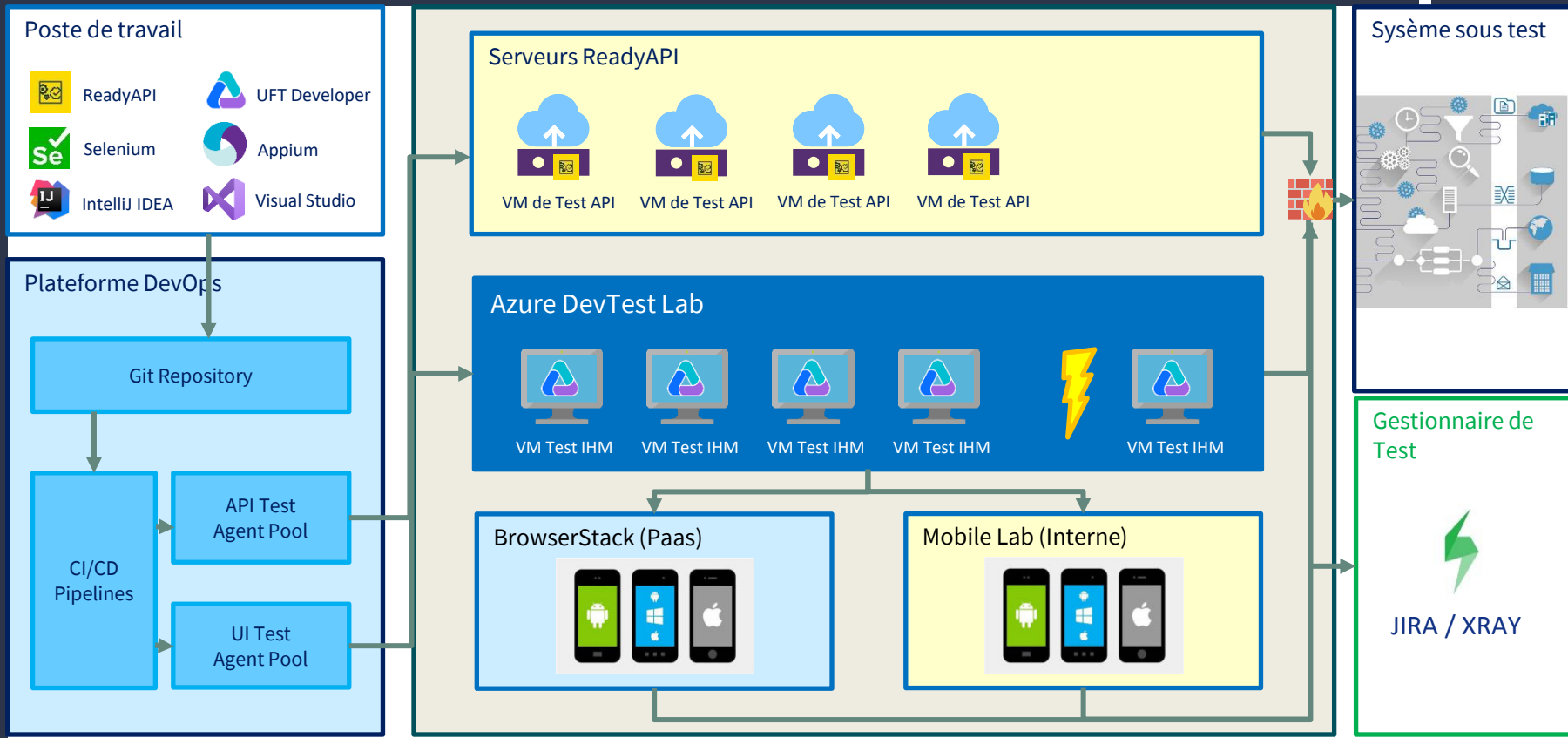
Couvertures de tests

- Couverture par branche
- Couverture par chemin
- Couverture des conditions
- MC/DC
- Pair-wise

Infrastructure d'Automatisation



Infrastructure d'Automatisation



Niveau Composant

Tests WebServices

TEST

- **Quoi** : Tester les WebServices développés par la Squad.
- **Quand** : Dès que la fonctionnalité est testable et que les développements sont terminés.
- **Qui** : Testeur ou Automaticiens
- **Comment** : Se référer à l'approche de test API ci-dessous

Niveau US

Scenarios d'acceptation

DEV

- **Quoi** : Automatiser les scenarios d'acceptations.
- **Quand** : Les scenarios sont documentés durant les sessions 3-amigos (PO, Dev, Test), sous forme de Gherkin
- **Qui** : Développeur Front
- **Comment** : Se référer à l'approche de tests d'Acceptation ci-dessous

Niveau MMF

Parcours clés

TEST

- **Quoi** : Automatiser les parcours utilisateurs de bout-en-bout ayant une forte valeur ajoutée métier.
- **Quand** : Identification au début de cadence (ex : PI Planning), Développement en parallèle du logiciel.
- **Qui** : Testeur ou Automaticien
- **Comment** : Se référer à l'approche de tests de Parcours ci-dessous

Approche de test API

JOURNÉE
FRANÇAISE DES
TESTS
LOGICIELS

- ❑ **Objectif**
 - Prouver démarrer le test au plus tôt et relever un maximum d'anomalies au plus tôt (Shift-Left)
- ❑ **Avantage**
 - Pouvoir utiliser des techniques de conceptions de tests pour des cas de tests "efficaces"
 - Cas de tests plus ciblés et moins complexes.
 - Analyse des résultats facilitée et de localisation plus aisées des anomalies.
 - Coût d'automatisation et maintenance inférieure aux tests IHM.
 - Tests exécutés plus rapidement.
 - Plus de confiance au niveau composant, moins de tests bout-en-bout nécessaires.
- ❑ **Connaissances et compétences à prévoir**
 - Besoin de connaissances techniques sur les protocoles Webservice (SOAP, REST, HTTP, ...)
 - Besoin de connaissances sur les outils de tests API

Approche de test Scenarios d'Acceptation

JOURNÉE
FRANÇAISE DES
TESTS
LOGICIELS

- ❑ **Identification**
 - Qui : Développeur Front, PO, Testeur
 - Quand : Durant la session des 3-Amigos, Définir les scenarios d'acceptance pour US et sélectionner les scenarios à automatiser.
- ❑ **Développement et Déploiement**
 - L'automatisation des scenarios d'acceptance est en générale assurée par le développeur front, via la méthode BDD (Behavior Driven Development)
 - Les scénarios sont souvent exprimés en Gherkin et automatisés avec l'outil habituel des développeurs
 - L'exécution de ces tests est intégrée dans le process DevOps

Approche de test Parcours clés

JOURNÉE
FRANÇAISE DES
TESTS
LOGICIELS

- ❑ **Objectif**
 - Automatiser le test de parcours clés pour assurer l'utilisabilité du Système Informatique
- ❑ **Définition**
 - Un parcours est un processus métier contenant un enchaînement d'actions permettant à un client de répondre à son besoin.
 - Un parcours est jugé clé lorsqu'il représente une forte valeur pour le client et pour l'entreprise.
 - La stratégie d'automatisation vise à couvrir prioritairement ces parcours clés.
- ❑ **Processus**



Approche de test API

❑ Objectif

- Pouvoir démarrer le test au plus tôt et relever un maximum d'anomalies au plus tôt (Shift-Left)

❑ Avantage

- Pouvoir utiliser des techniques de conceptions de tests pour des cas de tests “efficaces”
- Cas de tests plus ciblés et moins complexes.
- Analyse des résultats facilitées et de localisation plus aisées des anomalies.
- Coût d'automatisation et maintenance inférieure aux tests IHM.
- Tests exécutés plus rapidement.
- Plus de confiance au niveau composant, moins de tests bout-en-bout nécessaires.

❑ Connaissances et compétences à prévoir

- Besoin de connaissances techniques sur les protocoles Webservice (SOAP, REST, HTTP, ...)
- Besoin de connaissances sur les outils de tests API

Approche de test Scenarios d'Acceptation

❑ Identification

- Qui : Développeur Front, PO, Testeur
- Quand : Durant la session des 3-Amigos, Définir les scenarios d'acceptance pour US et sélectionner les scenarios à automatiser.

❑ Développement et Déploiement

- L'automatisation des scenarios d'acceptance est en générale assurée par le développeur front, via la méthode BDD (Behavior Driven Development)
- Les scénarios sont souvent exprimés en Gherkin et automatisés avec l'outil habituel des développeurs
- L'exécution de ces tests est intégrée dans le process DevOps

Approche de test Parcours clés

❑ Objectif:

- Automatiser le test de parcours clés pour assurer l'utilisabilité du Système Informatique

❑ Définition

- Un parcours est un processus métier contenant un enchainement d'actions permettant à un client de répondre à son besoin.
- Un parcours est jugé clé lorsqu'il représente une forte valeur pour le client et pour l'entreprise.
- La stratégie d'automatisation vise à couvrir prioritairement ces parcours clés.

❑ Processus:

