

Automatisation des tests d'interfaces

dans l'écosystème Apple

Mathias La Rochelle

Université de Montréal

2025-09-29

Aperçu

1. XCUIAutomation	2
1.1 C'est quoi ?	3
1.2 Pourquoi ?	4
1.3 Assertions XCTest	5
1.4 Composantes essentielles	6
1.5 Flux d'appels	7
1.6 Détection des composantes d'accessibilité	8
2. Demo	9
2.1 Contexte de l'application	10
2.2 Exemples de code	11
3. Annexe	13
3.1 Sources	14

1. XCUIAutomation

1.1 C'est quoi ?

Framework qui permet d'automatiser les interactions utilisateurs

1.1 C'est quoi ?

Framework qui permet d'automatiser les interactions utilisateurs

- Vérifier l'état de l'interface

1.1 C'est quoi ?

Framework qui permet d'automatiser les interactions utilisateurs

- Vérifier l'état de l'interface
- Vérifier le reflet approprié des vues selon le changement des
 - contrôleurs
 - modèles de données

1.1 C'est quoi ?


Framework qui permet d'automatiser les interactions utilisateurs

- Vérifier l'état de l'interface
- Vérifier le reflet approprié des vues selon le changement des
 - contrôleurs
 - modèles de données
- Créer des cas de tests pour simuler des gestes

1.1 C'est quoi ?

Framework qui permet d'automatiser les interactions utilisateurs

- Vérifier l'état de l'interface
- Vérifier le reflet approprié des vues selon le changement des
 - contrôleurs
 - modèles de données
- Créer des cas de tests pour simuler des gestes

Ne possède pas encore son framework moderne comme  *Swift Testing* pour XCTest.

IMPORTANT : Le préfixe *test* doit précéder le nom de chaque fonction à exécuter.

1.2 Pourquoi ?

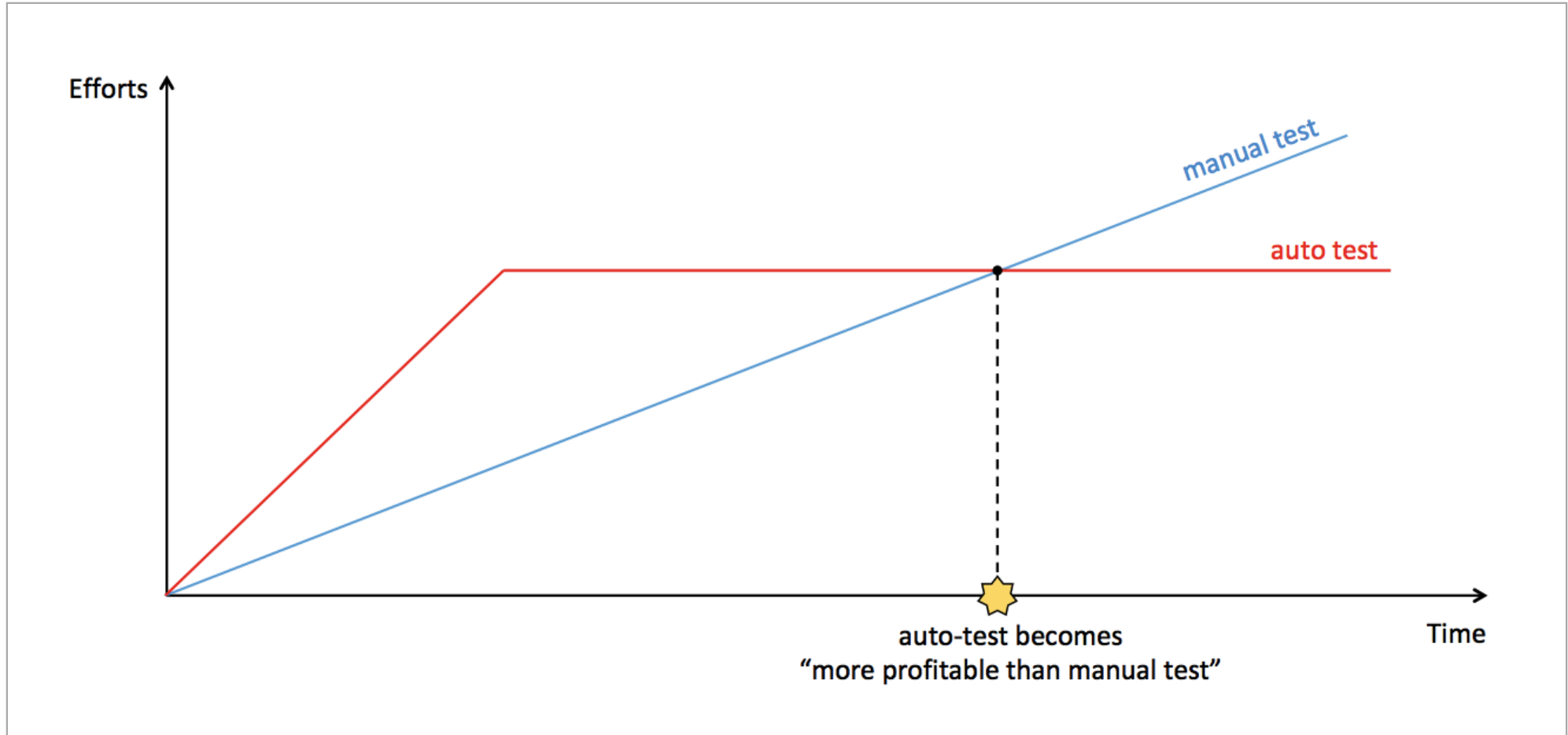


Fig. 1. – Source : [\[1\]](#) en Annexe

1.3 Assertions XCTest

- Booléens
 - XCTAssert
 - XCTAssertTrue
 - XCTAssertFalse
- Égalités et inégalités
 - XCTAssertEqual
 - XCTAssertNotEqual
 - XCTAssertEqual
 - XCTAssertNotIdentical
- Nil et Non-Nil
 - XCTAssertNil
 - XCTAssertNotNil
 - XCTUnwrap
- Comparaison de valeurs
 - XCTAssertGreaterThan
 - XCTAssertGreaterThanOrEqual
 - XCTAssertLessThan
 - XCTAssertLessThanOrEqual

1.4 Composantes essentielles

1. XCUIAutomation

1. XCUIApplication

- Instance devant être appelée avec `launch()` à chaque début de test. Arguments de lancement spécifiés en même temps de son exécution.

1.4 Composantes essentielles

1. XCUIApplication

- Instance devant être appelée avec `launch()` à chaque début de test. Arguments de lancement spécifiés en même temps de son exécution.

2. XCUIElementQuery

- Méthode utilisée par le `RunnerApp` et permettant de faire des requêtes au `UIWindow` du `HostApp` pour obtenir les `XCUIElement`.

1. XCUIApplication

- Instance devant être appelée avec `launch()` à chaque début de test. Arguments de lancement spécifiés en même temps de son exécution.

2. XCUIElementQuery

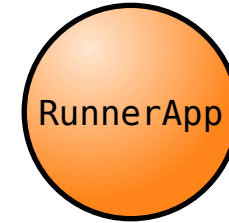
- Méthode utilisée par le `RunnerApp` et permettant de faire des requêtes au `UIWindow` du `HostApp` pour obtenir les `XCUIElement`.

3. XCUIElement

- Composante UI dont le simulateur peut interagir avec grâce à des fonctions comme `tap()`, `doubleTap()`, `press(_:)`, [`swipeLeft()`, `swipeRight()`, etc], `pinch(_:_:)`, `rotate(_:)` et plus.

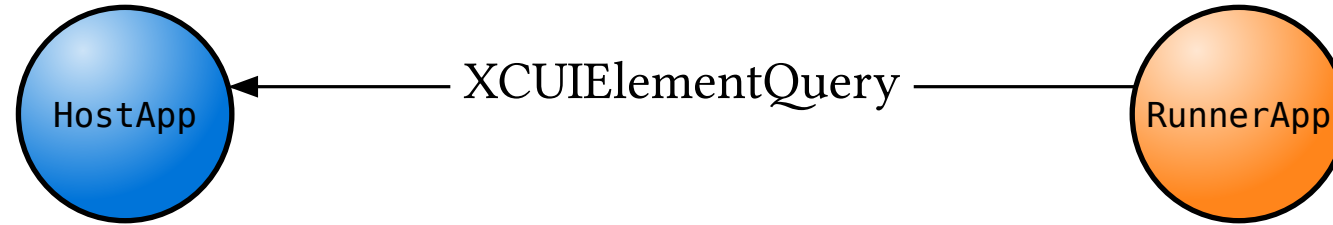
1.5 Flux d'appels

1. XCUIAutomation

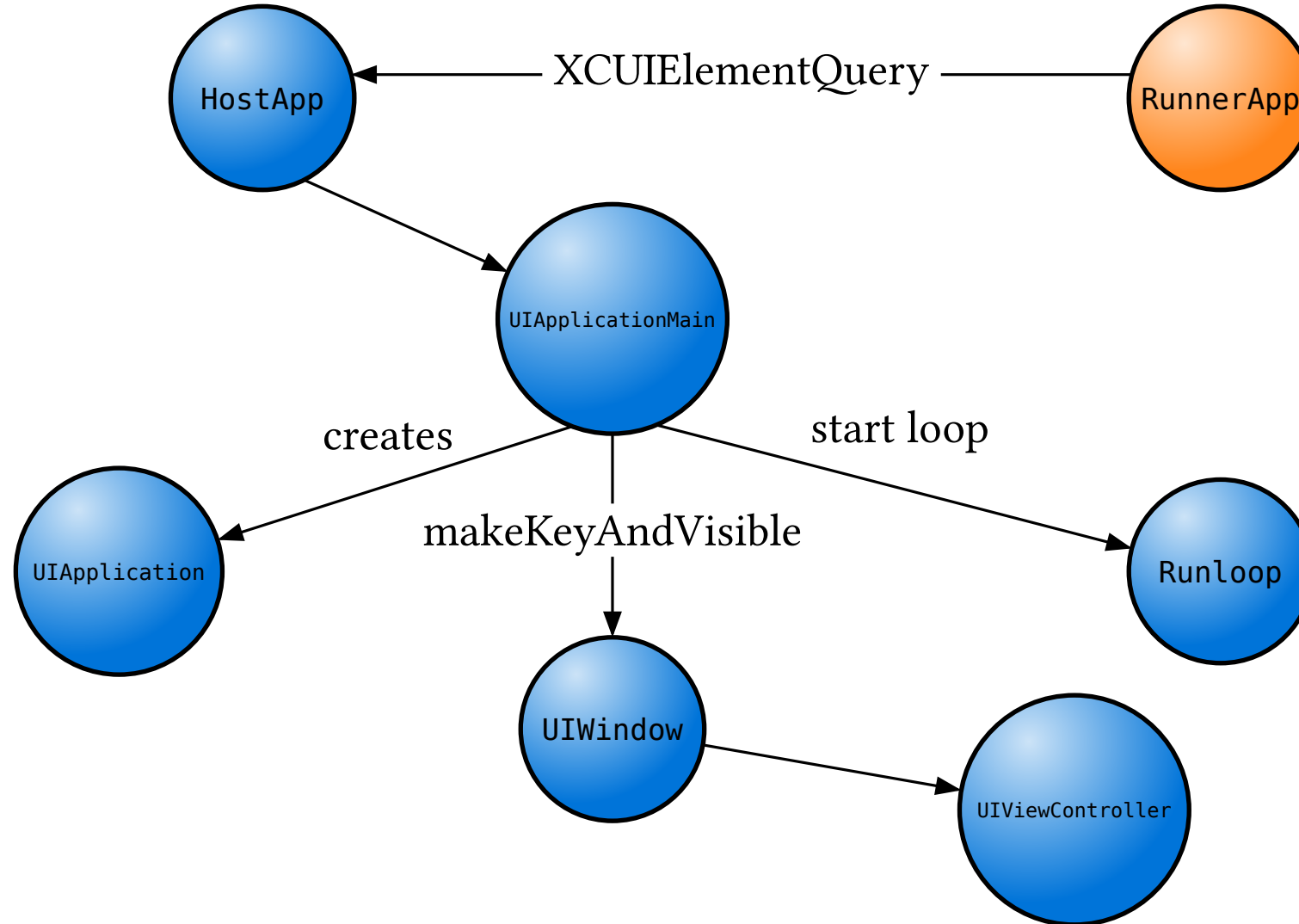


1.5 Flux d'appels

1. XCUIAutomation

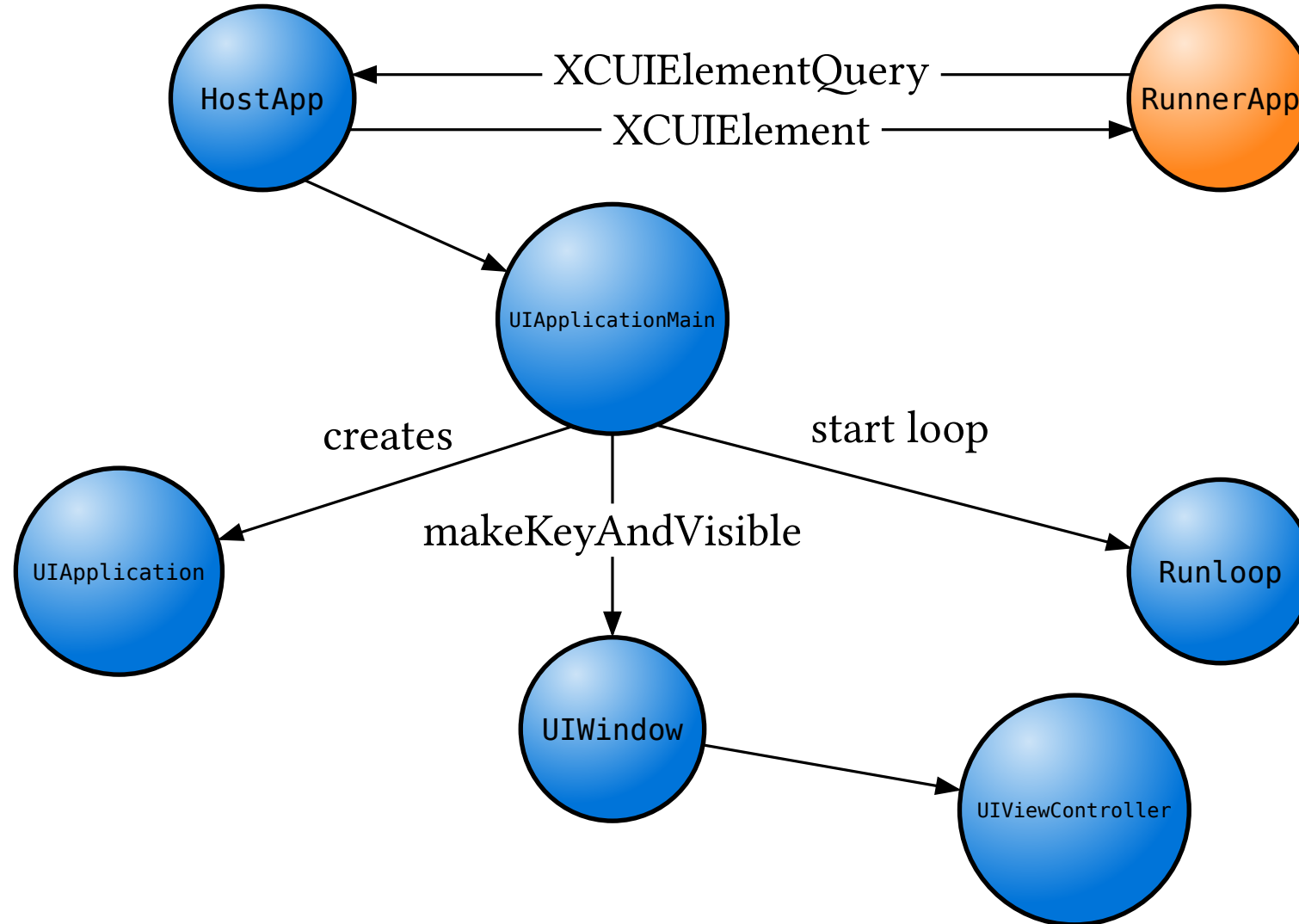


1.5 Flux d'appels



1.5 Flux d'appels

1. XCUIAutomation



1.6 Détection des composants d'accessibilité1. XCUIAutomation

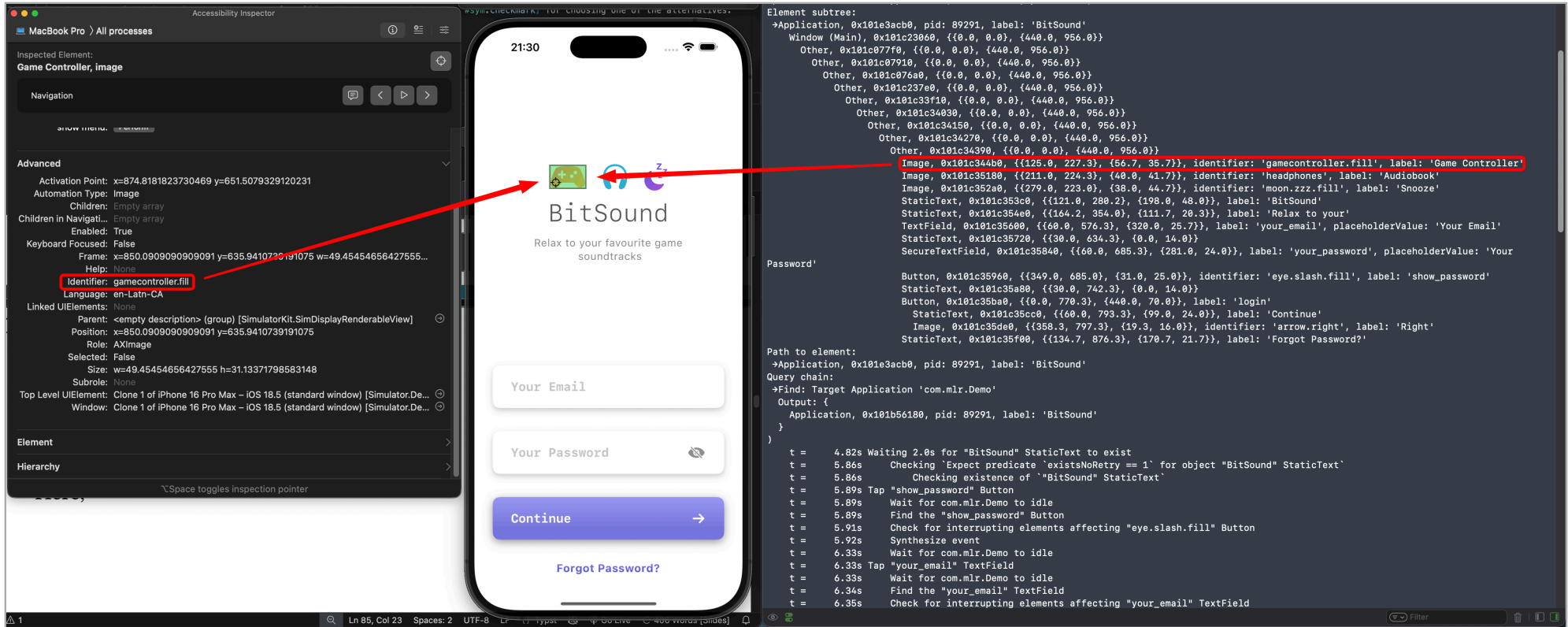
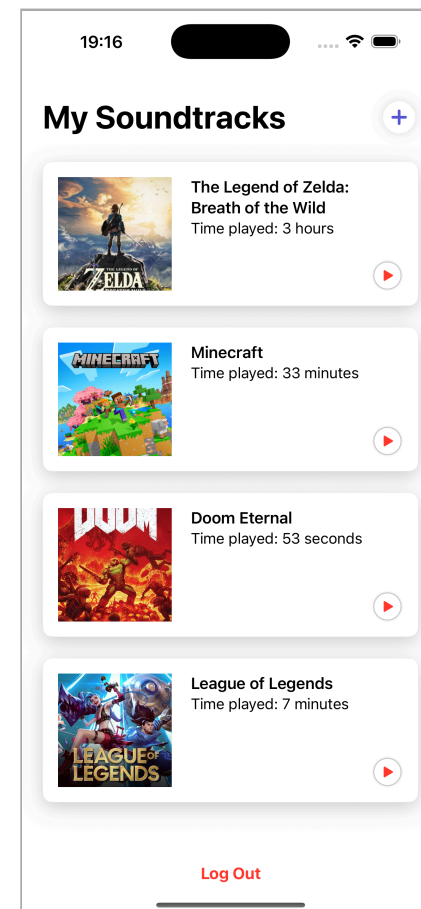
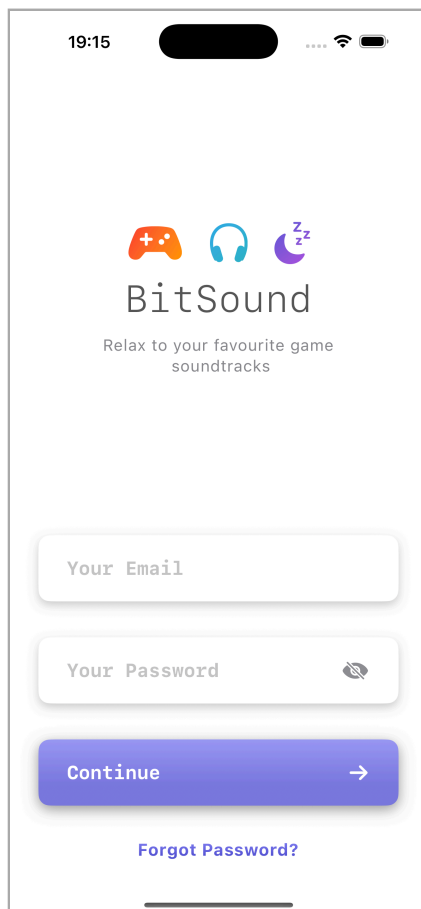


Fig. 2. – Correspondance entre  Accessibility Inspector et l'arbre des composants de l'interface utilisateur

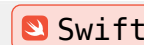
2. Demo

2.1 Contexte de l'application

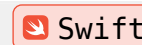
Application qui permet d'ajouter les musiques de nos jeux vidéos favoris.



```
1  @MainActor
2  func testExploreAuthenticationErrors() throws {
3      app.launch()
4
5      // Étape 1 : Vérifier qu'on est bien sur la page d'accueil
6      let nameOfApplication = app.staticTexts["BitSound"]
7      XCTAssertTrue(nameOfApplication.waitForExistence(timeout: 2)) // attendre tout chargement possible
8      // après ouverture de l'app
9
10     // Étape 2 : Localiser les champs cliquables
11     let emailTextField = app.textFields.matching(identifier: "your_email").firstMatch
12     let showPasswordButton = app.buttons.matching(identifier: "show_password").firstMatch;
13     showPasswordButton.tap()
14     let passwordTextField = app.textFields.matching(identifier: "your_password").firstMatch
15     let logInButton = app.buttons.matching(identifier: "login").firstMatch
16
17     // ...
18 }
```



```
1  private func enterCredentials(email: String, password: String, error: String?, expVal: Bool?,  
2                                emailTF: XCUIElement, passwordTF: XCUIElement, loginBT: XCUIElement) throws {  
3      emailTF.tap()  
4      emailTF.typeText(email)  
5      app.keyboards.buttons["Return"].tap()  
6      // Nécessaire sinon le simulateur ne trouve pas le prochain champ de texte  
7      passwordTF.tap()  
8      passwordTF.typeText(password)  
9      app.keyboards.buttons["Return"].tap()  
10     loginBT.tap()  
11     // Vérification de l'apparition du message d'erreur  
12     if let errorMessage = error {  
13         let errorLabel = app.staticTexts[errorMessage]  
14         XCTAssertNotEqual(errorLabel.exists, expVal)  
15     } else {  
16         try AuthError.allCases.forEach { error in  
17             if app.staticTexts[error.message].waitForExistence(timeout: 1) { throw error }  
18         }  
19     }  
20 }
```



3. *Annexe*

3.1 Sources

- [1] Akshay Pai, « Getting Started with XCTest: UI Automation Framework on iOS ». [En ligne]. Disponible sur: <https://www.browserstack.com/guide/getting-started-xcuietest-framework>
- [2] Thuyen Trinh, « Dealing With Flaky UI Tests in iOS: ». [En ligne]. Disponible sur: <https://trinhngocthuyen.com/posts/tech/dealing-with-flaky-ui-tests/>
- [3] Apple Developer, « XCUIAutomation: Replicate sequences of interactions and make sure that your apps user interface behaves as intended. ». [En ligne]. Disponible sur: <https://developer.apple.com/documentation/xcuiautomation>