



Rapport de stage

Création de tests pour les nouvelles compagnies aériennes à ajouter en Java.

Fait par :

Cyprien BONS

Sous la direction de :

M. COLETTA

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué à la réussite de mon stage.

Encadrement et accompagnement

M. Valentin CLAUDEL, mon maître de stage, pour son soutien, son accompagnement et ses précieux conseils tout au long de cette expérience.

Mme Cécile GOZLAN, pour son suivi attentif lors du processus de recrutement et son engagement à garantir les meilleures conditions possibles pour mon stage.

M. Christian SABBAGH, [PDG](#) d'Orchestra, pour m'avoir accueilli au sein de son entreprise et permis d'évoluer dans un environnement professionnel enrichissant.

Équipe QA - Transport

Un grand merci à toute l'équipe [QA](#) pour leur accueil et leur aide précieuse durant mon stage, en particulier :

- **Mme Daria SINGAEVSKAYA**
- **M. Mejd LIMEN**
- **Mme Thanh-Van LARONCE**

Leur collaboration et leur bienveillance m'ont permis de travailler dans les meilleures conditions possibles.

Encadrement académique

M. Rémi COLETTA, mon responsable de stage à l'IUT, pour son suivi et son encadrement tout au long de cette expérience.

Merci à tous pour votre aide et votre confiance.

Résumé

Au cours de mon stage chez **Orchestra** [2], entreprise du groupe Travelsoft, j'ai intégré l'équipe **QA - Transport** en tant que **stagiaire software engineer**. Mon travail a principalement porté sur **l'automatisation des tests** de validation des données affichées sur la plateforme de réservation lors de l'intégration de nouvelles compagnies aériennes.

Jusqu'à présent, ces tests étaient effectués manuellement par un membre de l'équipe **QA**, ce qui représentait une charge importante et un délai d'environ dix jours pour chaque intégration. Ce processus, bien que fonctionnel, présentait plusieurs inconvénients, notamment un risque d'erreurs humaines et une difficulté à absorber des périodes de forte activité. L'objectif du projet était donc de mettre en place une solution permettant d'automatiser ces vérifications, afin de réduire le temps de traitement à une dizaine de minutes et de libérer du temps pour l'équipe **QA**.

L'automatisation repose sur une comparaison des informations affichées sur la plateforme avec les données XML fournies par les API des compagnies aériennes, comme Air France ou Etihad. Ces tests permettent non seulement de s'assurer de la conformité des informations affichées aux données reçues, mais aussi de prévenir les régressions lors des évolutions du système.

Pour réaliser ce projet, j'ai travaillé en **Java** et utilisé le framework **Selenium**, qui permet d'interagir avec l'interface utilisateur et de récupérer les données affichées sur les différentes pages du processus de réservation. J'ai également exploité plusieurs outils internes à Orchestra, comme **Tracer** et **Devtools**, pour accéder aux logs et aux statuts des requêtes envoyées aux compagnies aériennes. La gestion du projet s'est faite avec **JIRA** pour le suivi des tâches et **GitLab** pour la gestion du code source.

Ce stage m'a permis de développer mes compétences en développement Java, en manipulation de données XML et en mise en place de tests automatisés. J'ai également appris à travailler dans un environnement structuré avec des outils professionnels et une méthodologie orientée qualité, tout en découvrant les enjeux techniques du secteur du tourisme numérique.

Sommaire

Remerciements	1
Encadrement et accompagnement.....	1
Équipe QA - Transport.....	1
Encadrement académique.....	1
Résumé	2
Sommaire	3
Table des figures	4
Glossaire	5
Introduction	6
I - Contexte du stage	7
1.1 - Présentation de ORCHESTRA.....	7
1.2 Dimension commerciale.....	8
1.3 Enjeux du stage.....	9
II – Analyse des missions	10
2.1 Technologies et outils utilisés.....	11
2.2 Spécifications fonctionnelles et techniques.....	12
III – Rapport technique	13
3.1 - Étape 1 : Initialisation des données et formulaire de recherche.....	13
3.2 - Étape 2 : Récupération des données de la page web.....	15
3.3 - Étape 3 : Récupération de la réponse API dans Tracer.....	18
3.4 - Étape 4 : Comparaison des données.....	22
3.5 - Étape 5 : Vérification sur Allure-Sandbox puis merge sur Master.....	23
IV - Méthodologie et organisation du projet	26
4.1 - Organisation du travail dans l'entreprise.....	26
4.2 - Méthodes de développement.....	28
4.3 - Travail effectué en plus du projet principal.....	29
Conclusion	30
Bibliographie	31
Annexes	32
Table des annexes.....	32

Table des figures

Figure 1 - Présentation Groupe TravelSoft.....	7
Figure 2 - Diagramme connectivité Orchestra.....	10
Figure 3 - Formulaire Réservation Transport.....	13
Figure 4 - Base de données.....	14
Figure 5 - Code d'utilisation de la base de données.....	14
Figure 6 - Code de récupérer des données par XPath.....	16
Figure 7 - Code HTML où on utilise un XPath.....	16
Figure 8 - Informations récupérées avec le XPath.....	17
Figure 9 - Liste Objects TransportSolution en Debug.....	17
Figure 10 - Tracer, recherche de la réponse de la source via ID.....	18
Figure 11 - Tracer, réponse de la source (en jaune le xml a récupérer).....	19
Figure 12 - Exemple de Xml récupéré dans Tracer.....	19
Figure 13 - Code de récupération du PaxJourney dans le xml.....	20
Figure 14 - Code de marshallng du XML en objet <U>.....	20
Figure 15 - DebugMode affichant l'objet AirShopping reconstitué.....	21
Figure 16 - Code de comparaison des deux objets.....	22
Figure 17 - Merge request sur GitLab.....	23
Figure 18 - Pipeline Scheduler sur GitLab.....	24
Figure 19 - Allure-Sandbox avec des tests en échec.....	24
Figure 20 - Repository graph GitLab montrant la fusion des commits.....	25
Figure 21 - Allure, branche Master.....	25
Figure 22 - Jira, fiche projet test auto airline intégration.....	26
Figure 23 - Tableau de suivi Excel.....	27
Figure 24 - Interface Slack.....	27

Glossaire

- **API (Application Programming Interface)** : Interface permettant à des logiciels de communiquer entre eux et d'échanger des données.
- **B2C (Business to Consumer)** : Modèle commercial où une entreprise vend directement ses produits ou services aux particuliers.
- **BUT (Bachelor Universitaire de Technologie)** : Diplôme en trois ans alliant théorie et pratique dans un domaine spécialisé.
- **IATA (International Air Transport Association)** : Organisation qui standardise et régule le transport aérien mondial.
- **ID (identifiant)** : Valeur unique attribuée à un élément pour le distinguer des autres et permettre son identification dans un système.
- **IUT (Institut Universitaire de Technologie)** : Etablissement d'enseignement supérieur en France proposant des formations professionnalisantes en deux ou trois ans.
- **NDC (New Distribution Capability)** : Norme de [l'IATA](#) permettant une distribution plus flexible et personnalisée des services aériens, facilitant l'échange de données entre les compagnies aériennes, les agences de voyages et les consommateurs.
- **PDG (Président-directeur général)** : Dirigeant principal d'une entreprise, responsable des décisions stratégiques.
- **QA (Assurance qualité)** : Ensemble des processus visant à garantir la qualité et la fiabilité d'un logiciel.
- **SaaS (Software as a Service)** : Logiciel accessible en ligne sans installation locale, souvent via un abonnement.
- **XPath** : Langage de requête utilisé pour localiser des éléments spécifiques sur une page web en analysant son code HTML et CSS, permettant ainsi d'extraire ou d'interagir avec ces éléments.

Introduction

Dans le cadre de ma deuxième année de [BUT](#) Informatique à l'[IUT](#) de Montpellier-Sète, j'ai effectué un stage de 8 semaines au sein de l'entreprise Orchestra, une filiale du groupe Travelsoft, spécialisée dans les technologies du tourisme. Orchestra développe une plateforme logicielle en ligne [SaaS](#) facilitant la communication entre les producteurs et les distributeurs de voyages. Cette solution permet aux agences de voyages de proposer des offres combinant transport, hébergement et services annexes à leurs clients. En tout, environ 400 partenaires utilisent cette plateforme, parmi lesquels des enseignes comme Leclerc Voyages et Carrefour Voyages.

[\[Voir Annexe 1\]](#) - [\[Voir Annexe 2\]](#)

J'ai intégré l'équipe **QA - Transport** en tant que software engineer stagiaire, où ma mission principale consistait à **automatiser les tests de validation des informations** affichées sur les pages de réservation. Jusqu'à présent, ces tests étaient réalisés manuellement, ce qui prenait environ 10 jours pour chaque nouvelle compagnie aérienne intégrée à la plateforme. L'objectif du projet était donc de réduire considérablement ce délai grâce à l'automatisation, permettant ainsi un gain de temps important pour l'équipe, mais aussi une réduction des erreurs humaines et une meilleure fiabilité des données affichées aux utilisateurs.

Ce projet s'inscrit également dans une démarche de prévention des régressions : en automatisant les tests, il devient plus facile de s'assurer que les modifications apportées au système n'engendrent pas d'anomalies sur les données affichées. Pour mener à bien cette mission, j'ai utilisé des technologies telles que Selenium pour l'exécution des tests, GitLab pour la gestion du code et Jira pour le suivi des tâches et des avancées du projet.

Au fil du stage, j'ai dû me familiariser avec l'infrastructure complexe de la plateforme Orchestra et comprendre le fonctionnement des [API](#) des compagnies aériennes, dont les données sont transmises en XML. L'un des défis majeurs a été d'assurer une comparaison fiable entre les résultats renvoyés par l'[API](#) et les informations affichées sur les pages web.

Ce stage a été une expérience enrichissante, tant sur le plan technique qu'organisationnel. Il m'a permis d'approfondir mes compétences en développement, tests et automatisation, tout en découvrant le fonctionnement d'une entreprise évoluant dans le secteur du tourisme numérique.

I - Contexte du stage

1.1 - Présentation de ORCHESTRA

L'entreprise

Créée en 2005, Orchestra répond aux défis croissants du secteur du tourisme, notamment le besoin d'automatisation. En mettant en place une plateforme unique, Orchestra est devenue une solution compétitive permettant aux opérateurs de voyage d'améliorer leur efficacité et de réduire leurs coûts opérationnels.

Orchestra appartient au groupe Travelsoft. Fondée à Paris en 2000, Travelsoft est une entreprise leader dans les logiciels dédiés aux acteurs du tourisme. Ses technologies gèrent chaque année plus de 35 milliards d'euros de réservations de voyages.

Orchestra est un acteur spécialisé dans le domaine du voyage et agit en tant qu'intermédiaire en accompagnant les agences de voyage. Orchestra leur propose une plateforme de réservation personnalisée leur permettant de créer des « packages » comprenant hôtels, transports, prestations, activités, etc...

Le groupe est présent dans plus de 90 pays et opère sous plusieurs marques :

- **Orchestra** (France)
- **Traffics** (Allemagne et Europe de l'Est)
- **Travel Compositor** (Espagne, Europe du Sud, Amérique latine, Asie)
- **Eventiz** (France, Belgique)
- **ATCORE Technology** (Royaume-Uni)
- **TravelgateX** (Espagne)
- **Travel Connection Technology** (Roumanie)



Figure 1 - Présentation Groupe TravelSoft

Sa situation géographique

Le siège social d'Orchestra se situe à Paris, avenue de l'Opéra, juste en face de l'Opéra Garnier. C'est dans ces bureaux que j'ai effectué mon stage. L'entreprise possède également des bureaux à Dijon et à Séville.

J'ai travaillé uniquement dans les bureaux de Paris, mais une partie des salariés est en télétravail. Orchestra compte des collaborateurs en France, en Espagne et à Dubaï. Elle travaille également en étroite collaboration avec une autre filiale de Travelsoft basée en Roumanie. Par ailleurs, elle fait aussi appel à des prestataires externes situés en Colombie.

1.2 Dimension commerciale

Orchestra évolue dans le secteur du tourisme numérique et des technologies de voyage. Sa plateforme SaaS permet aux professionnels du tourisme d'automatiser la gestion des réservations, des stocks et des prix, facilitant ainsi la distribution des offres de voyages sur Internet.

Modèle économique

L'entreprise génère son chiffre d'affaires en fonction du contrat signé avec chaque client. Plusieurs modèles de facturation sont possibles :

- **Un pourcentage sur chaque vente**
- **Un abonnement mensuel fixe,**
- **Une facturation au nombre de requêtes traitées**

Profil des clients

La cible d'Orchestra comprend :

- **Les agences de voyage**
- **Les tour-opérateurs,**
- **Les entreprises souhaitant vendre des voyages** (billets d'avion, train, bus) avec ou sans prestations complémentaires (hôtels, locations de voiture, forfaits de ski, activités touristiques).

Fonctionnalités principales

Orchestra propose principalement des **packs de voyage** combinant transport (avion, train, bus) et services complémentaires (hébergement, location de voiture, activités touristiques). La plateforme facilite ainsi la création, la gestion et la distribution de ces offres aux clients finaux.

En plus de la réservation, Orchestra prend également en charge toute la gestion associée aux voyages, notamment :

- **Facturation,**
- **Modifications de voyage**
- **Envoi de documents**

- **Enregistrement des transactions**

Grâce à ces fonctionnalités, Orchestra **simplifie le travail des professionnels du tourisme**, optimise leurs opérations et garantit une **expérience client fluide et automatisée**.

1.3 Enjeux du stage

Le projet est important pour l'entreprise car il permet un gain de temps significatif, améliorant ainsi la qualité du service pour les clients. En automatisant le processus, on réduit les délais et on évite les erreurs humaines.

À terme, l'équipe QA n'aura plus qu'à appuyer sur un bouton et attendre environ 10 minutes pour exécuter les tests, au lieu des 10 jours nécessaires auparavant.

Le principal enjeu du projet réside dans l'assimilation de toute l'infrastructure d'Orchestra, ce qui représente déjà une étape. Ensuite, il faut développer les différents outils permettant d'automatiser ces tests et de les intégrer efficacement dans l'environnement existant.

Ce projet est basé sur les transports, principalement les compagnies aériennes, je travaille sur un module propre aux tests automatiques de l'équipe QA (Quality Assurance), appelé « TestAuto » regroupant les tests simulant différents scénarios de réservation de vols.

II – Analyse des missions

Mon stage avait pour mission principale la création de tests automatisés visant à garantir la conformité des informations affichées sur les pages de réservation de la plateforme de test. L'objectif était de comparer les données retournées par l'[API](#) de différentes compagnies aériennes (par exemple, Air France ou Etihad) avec celles affichées sur les pages de la plateforme.

[\[Voir Annexe 6\]](#)

Un défi majeur du projet était de rendre les tests aussi génériques que possible. En effet, chaque compagnie aérienne renvoie des données sous des formats différents : certaines utilisent du JSON (qui n'est pas concerné par notre projet), tandis que d'autres renvoient des fichiers XML. De plus, chaque XML suit un format spécifique, qui varie selon la version de la norme [NDC](#). Par exemple, bien que les informations sur les bagages puissent être présentes, elles peuvent être organisées différemment, même au sein d'une même version à l'autre de cette norme.

Les tests seront dans un premier temps sur Allure Sandbox. Ce dernier permet d'exécuter des tests sur une branche spécifique, sur demande, contrairement à Allure qui exécute des tests de manière récurrente sur la branche master (tous les jours chez Orchestra). Allure Sandbox est ainsi utilisé pour les projets en cours et pour tester des branches spécifiques.

Le processus de nos tests se déroule en plusieurs étapes : la première consiste à récupérer les données depuis la page web (vue client), puis à extraire les informations dans les logs de Tracer, qui contiennent les réponses de l'API.

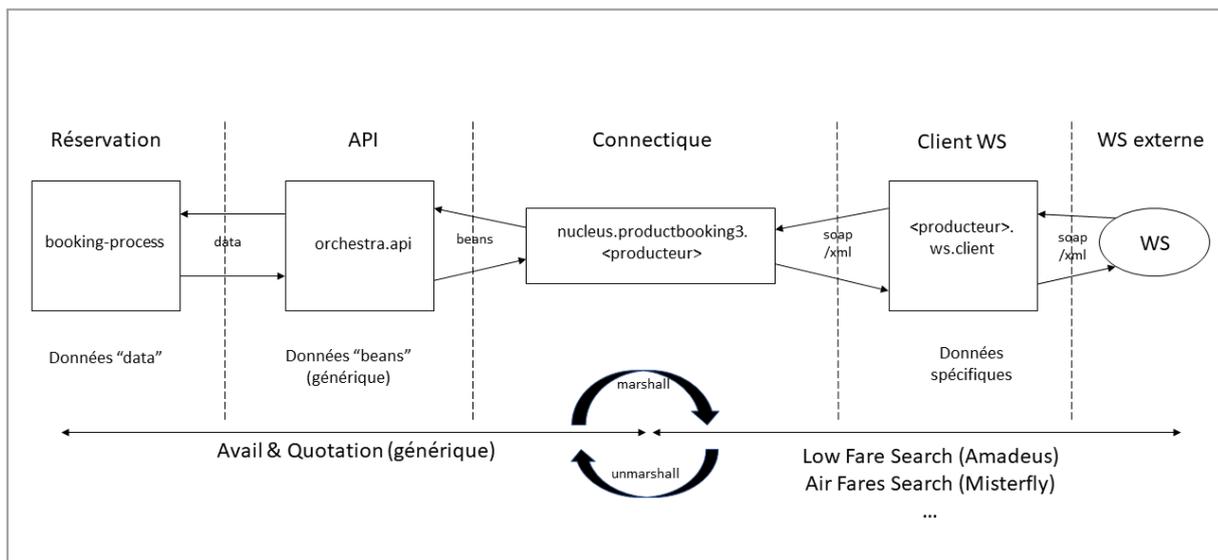


Figure 2 - Diagramme connectivité Orchestra

2.1 Technologies et outils utilisés

Langages de programmation

Lors de ce stage, j'ai utilisé exclusivement le langage **Java**. L'intégralité du projet a été développée en Java, mais j'ai également été amené à lire et manipuler des fichiers **XML**, car toutes les requêtes et réponses échangées avec les systèmes partenaires sont sous ce format.

Frameworks

Pour automatiser les tests, j'ai utilisé **Selenium**, un framework déjà en place au sein de l'équipe. Selenium permet de simuler les actions de l'utilisateur sur l'interface web, comme cliquer sur des boutons, remplir des champs, récupérer des informations affichées, ou encore sélectionner des options dans des menus déroulants.

Logiciels et plateformes utilisés

- **JIRA** : Outil de gestion de projet qui permet de suivre l'avancement des tâches, de décrire les blocages rencontrés et de créer des tickets lorsque j'ai besoin d'aide ou d'une validation.
- **GitLab** : Plateforme de gestion de version utilisée pour sauvegarder le code et collaborer avec les autres membres de l'équipe. Chaque modification était versionnée et documentée.
- **DevTools** : Outil interne propre à Orchestra, qui permet d'effectuer différentes actions comme envoyer une requête de recherche de vol, consulter les logs techniques, vérifier l'état des logiciels ou encore suivre les différentes étapes d'une réservation.
- **Tracer** : Autre outil interne à Orchestra, Tracer permet de récupérer et filtrer les logs, d'analyser l'état de chaque requête ou réponse, de vérifier les temps de traitement et d'identifier d'éventuelles erreurs.
- **Slack** : Plateforme de communication interne, utilisée pour échanger avec les membres de l'équipe, les autres services, ou encore poser des questions rapides. Slack joue un rôle central dans la coordination entre les salariés, qu'ils soient au bureau ou en télétravail.
- **Allure** : Outil de reporting utilisé pour visualiser les résultats des tests automatisés. Il permet d'analyser les éventuels échecs, de consulter des rapports détaillés et de faciliter le suivi des tests effectués sur la plateforme.
- **Confluence** : Outil collaboratif permettant de créer, organiser et partager des documents, des notes et des bases de connaissances au sein d'une équipe ou d'une entreprise.

2.2 Spécifications fonctionnelles et techniques

Besoins fonctionnels

L'objectif principal des tests est de vérifier automatiquement la cohérence des informations de vol entre les données retournées par l'[API](#) des compagnies aériennes et l'affichage sur les pages web de la plateforme. Les tests doivent être aussi génériques que possible, en utilisant des méthodes permettant d'assurer leur fonctionnement même lorsque le XML renvoyé par l'[API](#) n'est pas de la même version de la norme [NDC](#). Cela garantit que les tests fonctionnent quel que soit le format ou la version spécifique des données.

Besoins non fonctionnels

- **Fiabilité des tests** : Les tests doivent être robustes et fonctionner aussi souvent que possible. Cependant, il peut arriver que certains tests échouent pour des raisons externes, comme des problèmes avec l'[API](#) cliente, des défaillances de la plateforme, des pages qui se chargent trop lentement, ou des actions (clics) trop rapides. Il est donc important de prendre en compte ces scénarios à l'avance pour minimiser les erreurs.
- **Maintenabilité du code** : Le code des tests doit être structuré de manière modulaire et claire, afin de pouvoir être facilement adapté aux spécificités de chaque compagnie aérienne, et permettre une évolution du projet dans le temps.
- **Documentation** : La documentation doit être claire et compréhensible pour tous les membres de l'équipe, avec des variables en anglais et des descriptions en anglais pour les méthodes complexes, afin que le code puisse être compris par des prestataires internationaux.
- **Logging et suivi** : L'ajout de *Logger* et de *Steps* pour l'affichage des résultats dans Allure est indispensable afin d'avoir une visibilité sur les étapes des tests et d'assurer une traçabilité des actions effectuées durant l'exécution des tests.

III – Rapport technique

3.1 - Étape 1 : Initialisation des données et formulaire de recherche

Description

La première étape consiste à lancer une recherche via un formulaire web (voir figure ci-dessous). Ce formulaire est rempli avec des informations préalablement stockées dans la base de données.

Cette partie du projet avait déjà été développée avant notre arrivée dans l'entreprise, car elle était utilisée pour tester d'autres fonctionnalités. Je n'ai donc pas eu à le coder moi-même. Cependant, j'ai adapté son utilisation à nos propres tests.

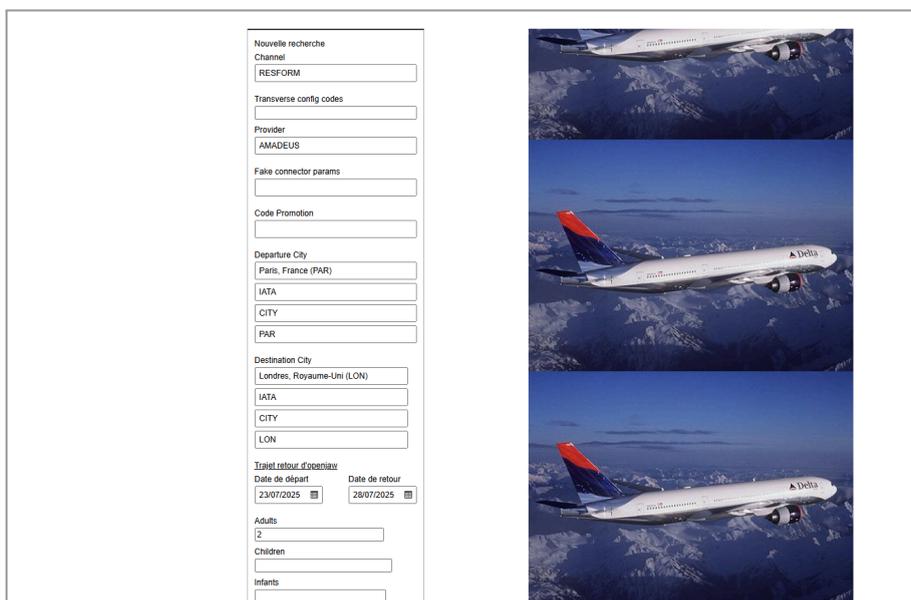
The image shows a web form for a flight search on the left and a photograph of a Delta airplane in flight on the right. The form is titled 'Nouvelle recherche' and contains several input fields: 'Channel' (with 'RESFORM' entered), 'Transverse config codes', 'Provider' (with 'AMADEUS' entered), 'Fake connector params', 'Code Promotion', 'Departure City' (with 'Paris, France (PAR)' entered), 'IATA', 'CITY', 'PAR', 'Destination City' (with 'Londres, Royaume-Uni (LON)' entered), 'IATA', 'CITY', 'LON', 'Trajet retour d'opérateur', 'Date de départ' (with '23/07/2025' selected), 'Date de retour' (with '28/07/2025' selected), 'Adults' (with '2' entered), 'Children', and 'Infants'. The photograph shows a Delta airplane flying over a mountain range.

Figure 3 - Formulaire Réservation Transport

Base de données

Pour nos tests, j'utilise cinq jeux de données par compagnie aérienne. Chaque compagnie couvre des itinéraires différents en fonction de son secteur d'activité.

Par exemple, pour Air France, il y a les scénarios suivants :

- 2 adultes - Aller-Retour (AR)
- 2 adultes, 2 enfants, 1 bébé - Aller-Retour (AR)
- 2 adultes, 1 bébé - Aller-Retour (AR)
- 2 adultes, 2 enfants, 1 bébé - Aller Simple (AS)
- 2 adultes, 2 enfants, 1 bébé - Open Jaw (OJ)

Les données concernant les aéroports de départ et d'arrivée varient pour chaque compagnie aérienne et ne sont pas les mêmes d'une compagnie à l'autre, afin d'assurer une diversité dans les tests.

Il existe trois types de trajets :

- **AR** - Aller-Retour : Exemple : Paris → New York / New York → Paris
- **AS** - Aller Simple : Exemple : Paris → New York
- **OJ** - Open Jaw : Exemple : Paris → New York, puis Miami → Paris
 - (l'aéroport de retour est différent de celui de départ)

Id_test	Departu...	Destina...	Departur...	Anni...	FlightType
INTEGRATION_ETIHAD_005	ABU DHABI	BERLIN	2025-04-10	2025-04-17	2	1	5	1	AS	
INTEGRATION_ETIHAD_004	PARIS	ABUDHABI	2025-04-10	2025-04-17	2	0	0	0	OJ	
INTEGRATION_ETIHAD_003	ABU DHABI	BUDAPEST	2025-04-10	2025-04-17	2	1	0	0	AR	
INTEGRATION_ETIHAD_002	PARIS	ABU DHABI	2025-04-10	2025-04-17	2	1	5	1	AR	
INTEGRATION_ETIHAD_001	ABU DHABI	NEW YORK	2025-04-10	2025-04-17	2	0	0	0	AR	
INTEGRATION_AIRFRANCE_005	LONDRES	MONTREAL	2025-06-12	2025-06-14	2	1	5	1	AS	
INTEGRATION_AIRFRANCE_004	LONDRES	MADRID	2025-06-12	2025-06-25	2	0	0	0	OJ	
INTEGRATION_AIRFRANCE_003	LONDRES	MONTREAL	2025-06-12	2025-06-14	2	1	5	0	AR	
INTEGRATION_AIRFRANCE_002	PARIS	PEKIN	2025-06-12	2025-06-14	2	1	5	1	AR	
INTEGRATION_AIRFRANCE_001	HELSINKI	JOHANNESBURG	2025-04-10	2025-04-17	2	0	0	0	AR	

Figure 4 - Base de données

Apprentissages critiques

Pour réaliser cette tâche, j'ai dû organiser la base de données afin de pouvoir l'utiliser efficacement dans mon code via un appel.

J'ai appliqué la Compétence 4, AC 3 : Organiser la restitution de données.

Dans l'extrait de code ci-dessous, les fonctions en jaune sont celles utilisées pour récupérer les données et les stocker dans une variable de type ResaFlightRequest, une classe développée par Orchestra.

```

24 public class INTEGRATION_002_Etihad extends Integration_NDC {
25     // Sources
26     public static final String TEST_SOURCE_001 = "INTEGRATION_ETIHAD_001";
27     public static final String TEST_SOURCE_002 = "INTEGRATION_ETIHAD_002";
28     public static final String TEST_SOURCE_003 = "INTEGRATION_ETIHAD_003";
29     public static final String TEST_SOURCE_004 = "INTEGRATION_ETIHAD_004";
30     public static final String TEST_SOURCE_005 = "INTEGRATION_ETIHAD_005";
31 }
92 @Epic(EPIC_RESERVATION)
93 @Feature(FEATURE_AIRLINE_MAPPING)
94 @Story(STORY)
95 @Test(description = "Integration_Etihad, TransportResultsPage, 2 Adults,PAR --> AUH", priority = 1, enabled = true)
96 public void INTEGRATION_002_001_Etihad() {
97     initSearchPage(TEST_SOURCE_001);
98     extractDataAndCompareTransportResultsPage();
99 }
77 public void initSearchPage(String TEST_SOURCE) {
78     initTransportData(TEST_SOURCE);
79     ResaSteps.launchTransportSearch(transportSearchPage, driver, flightRequest);
80     Assert.assertTrue(transportResultsPage.areSearchResultsDisplayed(), s: "search results not displayed");
81 }
60 public void initTransportData(String idTest) {
61     flightRequest = HibernateDataProvider.findByIdTest(ResaFlightRequest.class, idTest);
62     client = HibernateDataProvider.findByIdTest(Client.class, id_test: "RESPRO-1");
63 }

```

Figure 5 - Code d'utilisation de la base de données

3.2 - Étape 2 : Récupération des données de la page web

Description

La page que je teste est la première page de résultats de recherche qui s'affiche après la sélection d'une destination et de dates de voyage. Elle affiche une liste des vols disponibles, avec les informations principales pour chaque vol.

Informations a récupéré

Les informations à récupérer :

- La date de départ
- La date d'arrivée
- La durée totale du voyage
- Le nombre d'escales
- Le prix du vol
- La compagnie aérienne
- Code [IATA](#)
- Nom Aéroport
- Nombre de passagers

[\[Voir Annexe 3\]](#) – [\[Voir Annexe 4\]](#) - [\[Voir Annexe 5\]](#)

Processus de récupération des données

Avec Selenium, il est possible de simuler des interactions utilisateur telles que cliquer sur des boutons, récupérer du texte affiché sur une page et stocker ces informations dans des variables. En réalité, tout ce qui peut être fait avec une souris et un clavier peut être automatisé avec Selenium.

Dans notre cas, j'utilise Selenium pour extraire les informations affichées sur les pages de réservation et les organiser sous forme d'objets. Ces données structurées seront ensuite comparées aux résultats renvoyés par l'[API](#) afin de vérifier leur exactitude.

Voici un exemple de code permettant de récupérer des données en utilisant des [Xpath](#) :

```

public String[] getSegmentTooltipInfo(String tooltipXPath) { 4 usages  ▲ Raghunath Rajavelu +2
    String dateXPath = "@ //div[@class='dDayZone']";
    String departureCityXPath = "@ //div[@class='cpt-travel-resume departure']//span[@class='city']";
    String departureAirportXPath = "@ //div[@class='cpt-travel-resume departure']//span[@class='airport']";
    String departureTerminalXPath = "@ //div[@class='cpt-travel-resume departure']//span[@class='terminal']";
    String departureTimeXPath = "//div[@class='cpt-travel-resume departure']//div[@class='travel-hours']";
    String arrivalAirportXPath = "@ //div[@class='cpt-travel-resume arrival']//span[@class='airport']";
    String arrivalCityXPath = "@ //div[@class='cpt-travel-resume arrival']//span[@class='city']";
    String arrivalTerminalXPath = "@ //div[@class='cpt-travel-resume arrival']//span[@class='terminal']";
    String arrivalTimeXPath = "//div[@class='cpt-travel-resume arrival']//div[@class='travel-hours']";
    String segmentRoot = "@ //div[@class='infos-card-elements']";
    String carrierCodeAndFlightNumberXPath = "@ //div[@class='content']//div[@class='travelBlock']//div[@class='cpt-travel-resume departure']";
    String baggageXPath = "@ //div[@class='content']//div[@class='travelBlock']//div[@class='cpt-travel-resume arrival']";
    String baggageXPath2 = "@ //div[@class='content']//div[@class='travelBlock']//div[@class='cpt-travel-resume arrival']";
    String basePath = findElements(By.xpath(xpathExpression: currentTooltipXPath + segmentRoot)).size() > 1 ? tooltipXPath : null;

    //set transport type
    String transportType = findElements(By.xpath(basePath)).size() > 0 ?
        findElement(By.xpath(basePath)).getAttribute(name: "data-transport-type") : null;

    // set class
    String cabinType = findElements(By.xpath(basePath)).size() > 0 ?
        findElement(By.xpath(basePath)).getAttribute(name: "data-transport-class") : null;
    String cabinCode = findElements(By.xpath(basePath)).size() > 0 ?
        findElement(By.xpath(basePath)).getAttribute(name: "data-booking-class") : null;
}

```

Figure 6 - Code de récupérer des données par [XPath](#)

Par exemple, pour récupérer les informations sur l'aéroport de départ du vol, affichées sous la forme "Paris - (Aéroport Ch. De Gaulle)" dans le code HTML de la page, on peut utiliser [XPath](#) suivant :

```

.///*[@class='modal ppn_fullScreen withHeader fade in']//div[@class='infos-card-elements']//div[@class='cpt-travel-resume departure']//span[@class='airport']

```

```

<!DOCTYPE html>
<html lang="fr">
<head>
</head>
<body class="modal-open" style="padding-right: 17px;">
  <input type="hidden" name="processType" value="TRANSPORT">
  <input type="hidden" name="currentStep" value="TRANSPORT_RESULTS">
  <input type="hidden" name="profile" value="MDFXC3zQobFV43wmgEgeSchgk16s/Nuqk11fMcyZh/6aUt6C21a5NoD5n5cs">
  <input type="hidden" name="sessionId" value="1F60rBy22EDUvdf41q128c495NDupb7e0kUpwiX">
  <input type="hidden" name="invocationEnvironment" value="">
  <input type="hidden" name="externalSessionKey" value="">
  <div class="srn_serpTransport">
  </div>
  <script nonce="2784114187627485414" type="text/javascript" src="/tsvweb/process_common/js/libs/jquery-3.6.0.min.js">
  </script>
  <script nonce="2784114187627485414" type="text/javascript" src="/tsvweb/process_common/js/libs/bootstrap-3.4.1.min.js">
  </script>
  <script nonce="2784114187627485414" type="text/javascript" src="/tsvweb/process_common/js/libs/models.js">
  </script>
  <!-- ORX booking process entry point -->
  <script nonce="2784114187627485414" type="module" src="/tsvweb/process_common/js/orchestra.bookingprocess.front/main.min.js?version=1743801828712">
  </script>
  <div id="modals">
    <div role="dialog" tabindex="-1" id="ppn_datePicker" class="modal ppn_fullScreen withHeader" data-backdrop="true" data-keyboard="true">
    </div>
    <div role="dialog" tabindex="-1" id="ppn_searchForm" class="modal ppn_fullScreen withHeader fade" data-backdrop="true" data-keyboard="true">
    </div>
    <div role="dialog" tabindex="-1" id="ppn_transportDetails_EI7" class="modal ppn_fullScreen withHeader fade in" data-backdrop="true" data-keyboard="true" style="z-index: 1040; display: block; padding-right: 17px;">
      <div class="modal-dialog">
        <div class="modal-content">
          <div class="modal-header">
          </div>
          <div class="modal-body">
            <div class="transportDetailsList">
              <div class="cpt-infos-card">
                <h2 class="title">
                </h2>
                <div class="infos-card-container company-head">
                  <div class="infos-card-elements" data-transport-type="flight" data-transport-class="ECONOMIC" data-booking-class="S">
                    <div class="top-banner-start">
                    </div>
                    <div class="dDayZone" data-dep-date="25-05-2025" data-arr-date="25-05-2025">
                    </div>
                    <div class="content">
                      <div class="cpt-travel-resume departure">
                        <div class="travel-hours">
                          <div class="travel-airport">
                            <span class="city" data-country-code data-city-code="PAR">Paris</span>
                            <span class="airport" data-airport-code="CDG">Paris - (Aéroport Ch. De Gaulle)</span>
                            <span class="terminal">
                            </span>
                          </div>
                        </div>
                      </div>
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>

```

Figure 7 - Code HTML où on utilise un [XPath](#)

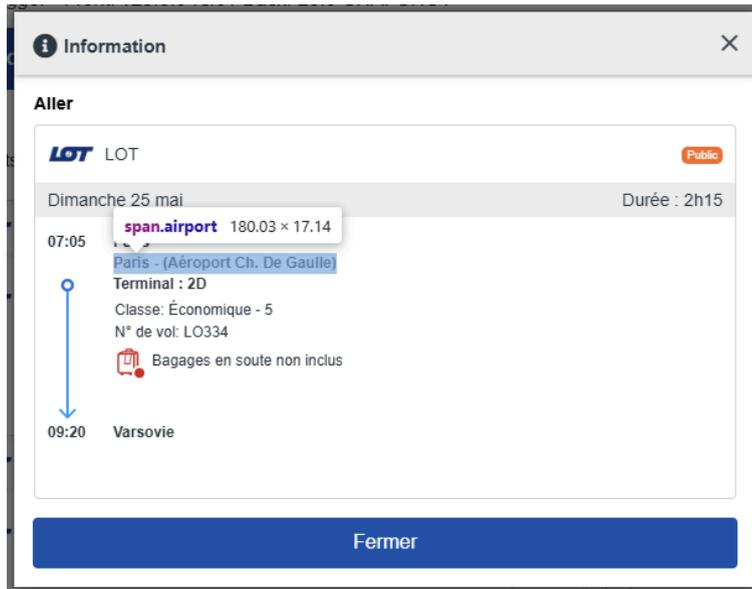


Figure 8 - Informations récupérées avec le [XPath](#)

Traitement des données

À l'issue du processus de récupération des données, on obtient une liste de *TransportSolution*, représentant un voyage complet. Chaque *TransportSolution* contient deux listes de *Segment* : l'une correspondant aux vols aller et l'autre aux vols retour. Chaque *Segment* contient les informations détaillées du vol. En outre, chaque *TransportSolution* contient un objet *Price*, qui présente les détails du tarif.

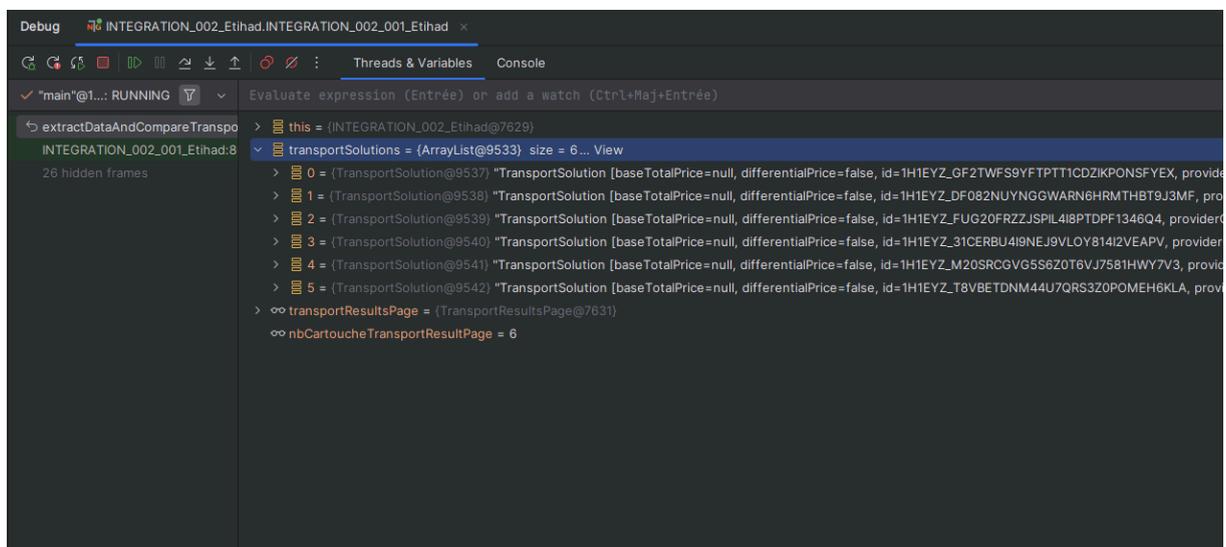


Figure 9 - Liste Objects TransportSolution en Debug

3.3 - Étape 3 : Récupération de la réponse API dans Tracer

Description

Après avoir récupéré les données affichées sur la page web, on doit maintenant obtenir la réponse [API](#) correspondante dans les logs via Tracer.

Pour cela, j'utilise un identifiant unique présent sur la page, qui me permet de retrouver dans les logs la réponse [API](#) associée à la requête effectuée.

La réponse [API](#) est retournée sous format XML, suivant la norme [NDC](#). Plus précisément, on récupère une réponse de type AirShoppingRS :

- AirShopping indique qu'il s'agit d'une liste de vols proposés,
- RS signifie "Response".

L'étape suivante consistera à mapper ces données XML afin de les comparer aux informations affichées sur la page.

Récupération du XML

Comme mentionné précédemment, j'utilise Tracer pour identifier les logs correspondant à notre recherche.

Ce processus est entièrement automatisé grâce à Sélénium, qui permet de naviguer dans l'interface et d'extraire les informations nécessaires via des [XPath](#).

The screenshot shows the Tracer application interface. At the top, there are search filters for 'Type de traces', 'Intervalle de recherche', 'Statut d'exécution', 'Session', 'Code externe produit', and 'Transaction ID de paiement'. Below these filters is a table of search results. The table has columns for DATE, DURATION, TYPE, SOURCE, STATUS, REQUEST, RESPONSE, STACK TRACE, ERROR CODE, ERROR EXTERNAL CODE, ERROR MESSAGE, ERROR CATEGORY, CHANNEL, DEPARTURE CITY, PRODUCT CODE, BEGIN DATE, DURATION, FAX, and CORR. The table contains several rows of data, including entries for 'Amadeus', 'Athlone', and 'Latan'.

DATE	DURATION	TYPE	SOURCE	STATUS	REQUEST	RESPONSE	STACK TRACE	ERROR CODE	ERROR EXTERNAL CODE	ERROR MESSAGE	ERROR CATEGORY	CHANNEL	DEPARTURE CITY	PRODUCT CODE	BEGIN DATE	DURATION	FAX	CORR
17/02/2025 14:02:05	1125	OnAPI	Amadeus	OK								RESPONSE+	PAR	AMADEUS-nul	2025-03-17	0:0	2:10:0	88495
17/02/2025 14:01:54	9188	Source	Amadeus	OK								RESPONSE+	PAR	AMADEUS-nul	2025-03-17	0:0	2:10:0	88495
17/02/2025 14:10:11	8645	OnAPI	Athlone	OK								RESPONSE+	PAR	AIRFRANCE-nul	2025-03-17	0:0	2:10:0	25726
17/02/2025 14:10:02	5538	Source	Athlone	OK								RESPONSE+	PAR	AIRFRANCE-nul	2025-03-17	0:0	2:10:0	25726
17/02/2025 14:13:58	20297	OnAPI	Latan	OK								RESPONSE+	B0B	LATAM-nul	2025-03-17	0:0	2:10:0	84534
17/02/2025 14:12:48	7368	Source	Latan	OK								RESPONSE+	B0B	LATAM-nul	2025-03-17	0:0	2:10:0	84534

Figure 10 - Tracer, recherche de la réponse de la source via ID

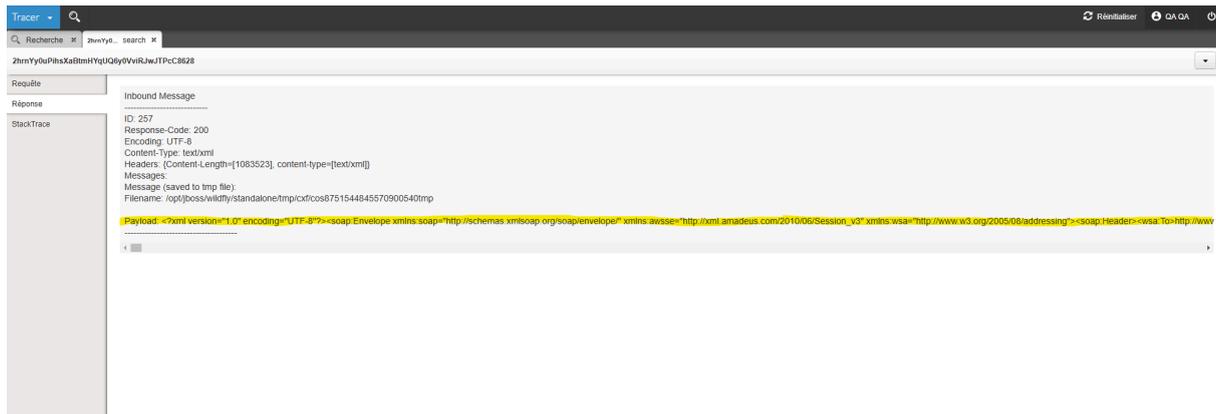


Figure 11 - Tracer, réponse de la source (en jaune le xml a récupérer)

Processus de récupération des données

Pour exploiter les données reçues, j'effectue un mapping du XML. Ce fichier est structuré en deux parties principales :

1. **OffersGroup** → Contient la liste des offres disponibles.
2. **DataLists** → Contient les informations détaillées sous forme de listes séparées :
 - **PaxSegmentList** : Liste des vols proposés,
 - **DatedMarketingSegmentList** : Dates de départ et d'arrivée,
 - **PriceClassList** : Tarification des offres.

Cette structure permet d'optimiser la gestion des offres : plusieurs offres peuvent partager le même vol ou d'autres éléments, sans répétition inutile des données. Chaque élément possède un ID unique, ce qui facilite leur liaison et permet un mapping efficace pour la suite du traitement.



Figure 12 - Exemple de Xml récupéré dans Tracer

```

public class PaxJourney_21_3 { 21 usages  ▲ Cyprien Bons
    // Info durée vol
    private javax.xml.datatype.Duration duration; 7 usages
    private String journeyID; 2 usages

    private List<PaxSegment_21_3> paxSegmentList = new ArrayList<>(); 2 usages

    public static List<PaxJourney_21_3> paxJourney_21_3_List(List<PaxJourneyType> paxJourneyList, List<PaxSegment_21_3> paxSegmentListAll) { 2 usages  ▲ Cyprien Bons
        List<PaxJourney_21_3> list = new ArrayList<>();
        for (PaxJourneyType pj : paxJourneyList) {
            PaxJourney_21_3 paxJourney = new PaxJourney_21_3();

            paxJourney.setDuration(pj.getDuration());
            paxJourney.setJourneyID(pj.getPaxJourneyID());

            for (String ps : pj.getPaxSegmentRefIDs()) {
                paxJourney.getPaxSegmentList().add(PaxSegment_21_3.getid(ps, paxSegmentListAll));
            }

            list.add(paxJourney);
        }
        return list;
    }
}

```

Figure 13 - Code de récupération du PaxJourney dans le xml

Apprentissages critiques

Pour récupérer les données, j'ai dû créer une fonction permettant de convertir un XML en un objet Java correspondant à une version spécifique de [NDC](#), comme AirShoppingRS. Cette méthode est générique et repose sur une classe abstraite utilisant un type paramétré <U>.

J'ai appliqué les connaissances que j'ai appris en cours pour développer cette solution, notamment en travaillant avec JAXB pour le marshalling des données XML.

Cela correspond à la Compétence 1, AC 1 : Élaborer et implémenter les spécifications.

```

public <U> U getXML(String data, Class<U> clazz) { 10 usages  ▲ Cyprien Bons +1
    try {
        Assert.assertNotNull(data, s: "Data Tracer is null");

        JAXBContext jaxbContext = JAXBContext.newInstance(clazz);
        Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();

        StringReader reader = new StringReader(data);
        U obj = (U) unmarshaller.unmarshal(reader);

        Assert.assertNotNull(obj, s: "Unmarshalling failed");

        return obj;
    } catch (JAXBException e) {
        Assert.fail(s: "Error while unmarshalling XML response");
        return null;
    }
}

```

Figure 14 - Code de marshalling du XML en objet <U>

Résultat

À l'issue de ce processus, j'obtiens un objet `AirShopping` contenant toutes les informations extraites de la réponse [API](#), prêtes à être comparées avec les données affichées sur la page web.

```

dataResponse = {IATAAirShoppingRS_21_3@10894}
  offerList = {ArrayList@10897} size = 200... View
    0 = {Offers_21_3@10904}
      id = "1H1EYZ_GF2TWFS9YFPTT1CDZIKPONSFYEX"
      paxJourneyArrival = {PaxJourney_21_3@11005}
        duration = {DurationImpl@11014} "PT14H20M"
        journeyID = "FLT1"
      paxSegmentList = {ArrayList@11016} size = 1... View
        0 = {PaxSegment_21_3@11019}
          datedMarketingSegment = {DatedMarketingSegment_21_3@11020}
            arrivalDateTime = {XMLGregorianCalendarImpl@11022} "2025-05-23T09:15:00.000"
            arrivalATALocationCode = "JFK"
            arrivalTerminalName = "4"
            datedMarketingSegmentID = "DMS14"
            duration = {DurationImpl@11026} "PT14H20M"
            departureDateTime = {XMLGregorianCalendarImpl@11027} "2025-05-23T02:55:00.000"
            departureATALocationCode = "AUH"
            departureTerminalName = "A"
            carrierCode = "EY"
            flightNumber = "1"
            paxSegmentID = "SEG4"
          paxJourneyDeparture = {PaxJourney_21_3@11006}
          prices = {Price_21_3[1]@11007} ... View
          arrivalBaggageAllowance = {ArrayList@11008} size = 1... View
          departureBaggageAllowance = {ArrayList@11009} size = 1... View
        1 = {Offers_21_3@10905}
        2 = {Offers_21_3@10906}
        3 = {Offers_21_3@10907}
        4 = {Offers_21_3@10908}
        5 = {Offers_21_3@10909}
  
```

Figure 15 - DebugMode affichant l'objet `AirShopping` reconstitué

3.4 - Étape 4 : Comparaison des données

Description

Maintenant que j'ai récupéré les informations de chaque côté, je peux les comparer entre elles.

Processus de comparaison

Tout d'abord, j'effectue un matching, c'est-à-dire que j'associe le bon élément de chaque côté avant de les comparer, car parfois les éléments ne sont pas dans le même ordre. Je fais cela à l'aide d'un [ID](#) disponible sur la page web.

Ensuite, je compare simplement les données avec des Assert, ce qui permet d'afficher les résultats dans les logs avec un message clair.

```
@Step("Compare XML data with data on transport results page")  Cyprien Bons
public static void compareTo(ResaFlightRequest flightRequest, TransportSolution transportSolution, Offers_21_3 offer) {
    // Info durée vol
    Assert.assertEquals(transportSolution.getTripsList()[0].getItineraryList()[0].getDuration().getValue(), offer.getPax
    if (offer.getPaxJourneyDeparture() != null) {
        Assert.assertEquals(transportSolution.getTripsList()[1].getItineraryList()[0].getDuration().getValue(), offer.ge
    }
    // List vol Arrival
    for (int i = 0; i < offer.getPaxJourneyArrival().getPaxSegmentList().size(); i++) {
        DatedMarketingSegment_21_3 dms = offer.getPaxJourneyArrival().getPaxSegmentList().get(i).getDatedMarketingSegmen
        Segment segment = transportSolution.getTripsList()[0].getItineraryList()[0].getSegmentList()[i];

        Assert.assertEquals(dms.getArrivalDateTimeDate(), segment.getArrivalDate());
        Assert.assertEquals(dms.getArrivalIATALocationCode(), segment.getArrivalLocation().getInnerLocation().getInnerLo
        Assert.assertEquals(dms.getArrivalTerminalName(), segment.getArrivalLocation().getInnerLocation().getInnerLocati

        Assert.assertEquals(dms.getDepartureDateTimeDate(), segment.getDepartureDate());
        Assert.assertEquals(dms.getDepartureIATALocationCode(), segment.getDepartureLocation().getInnerLocation().getInn
        Assert.assertEquals(dms.getDepartureTerminalName(), segment.getDepartureLocation().getInnerLocation().getInnerLo
```

Figure 16 - Code de comparaison des deux objets

3.5 - Étape 5 : Vérification sur Allure-Sandbox puis merge sur Master

Description

Pour que les tests soient visibles sur Allure, je dois merger notre branche sur Master. Cependant, avant d'être validé, je dois effectuer une merge request.

Cette merge request est soumise à validation par nos pairs, c'est-à-dire tous les autres développeurs de l'entreprise, y compris ceux qui ne sont pas directement dans notre équipe. Ils peuvent ainsi examiner le code, donner leur avis, suggérer des améliorations et détecter d'éventuelles erreurs.

Parallèlement à la merge request, je lance une simulation sur Allure-Sandbox. Cela me permet de vérifier quels tests passent ou échouent sur notre branche, de détecter d'éventuelles régressions et de faire les corrections nécessaires avant la fusion finale sur Master.

Création de la merge request et lancement d'Allure-Sandbox

- La merge request est créée sur GitLab, où les autres développeurs peuvent laisser des commentaires et approuver le code.

The screenshot shows a GitLab Merge Request (MR) page for a project named "New AutoTest By Sources". The MR is titled "feature/Transport" and is being merged into the "master" branch. It was requested by "Cyprien Bons" 2 weeks ago. The page includes an overview section with 12 comments, 162 commits, 37 pipelines, and 43 changes. A test description states: "Test consistant à récupérer la réponse XML de la source afin de la comparer aux données affichées sur la page web." There are 1 thumbs up and 0 thumbs down. The MR is approved by a user with a green checkmark. It shows that the MR was merged by "Cyprien Bons" 6 days ago, with options to "Revert" or "Cherry-pick". Merge details include: "Changes merged into master with 335d1214 (commits were squashed)." and "Did not delete the source branch." Below this, it shows "Pipeline #2460271 passed" for commit 335d1214 on the master branch 5 days ago. The "Activity" section shows: "Cyprien Bons requested review from @daria.brou, @mejdi.limem, @thanh-van.laronce, and @valentin.claudet 2 weeks ago", "Cyprien Bons assigned to @cyprien.bons and @rhys.trouve 2 weeks ago", and "Cyprien Bons added 5 commits 2 weeks ago" with a list of commit hashes and their descriptions, including "Merge branch 'master' into 'feature/Transport'".

Figure 17 - Merge request sur GitLab

- Dans la section Pipeline Scheduler, je pouvons configurer et lancer Allure-Sandbox pour tester notre branche `feature/Transport`, en jaune ci-dessous.

platform / quality / TestAuto / Schedules

All 10		Active	Inactive				New schedule
Description	Interval	Target	Last Pipeline	Next Run	Owner		
PRERECETTE - Internal product creation - Paris (PAR)	38 2 *** Europe/Paris	internal-products	Passed	Inactive			
TCD - Internal product creation - Paris	12 12 *** Europe/Paris	internal-products	Passed	Inactive			
TCD - Internal product creation - Séville (SVQ)	36 19 *** Europe/Paris	internal-products	Passed	Inactive			
QUALITE - Internal product creation - Paris (PAR)	45 20 *** Europe/Paris	internal-products	Passed	Inactive			
All Résa tests - Qualité - 1 am All days	00 01 *** Europe/Paris	MIGRATION_22_8	Passed	Inactive			
Tests - Sandbox - Qualité - On demand	0 0 *** Europe/Paris	feature/transport	Passed	Inactive			
Tests - Sandbox - Orchestra360 rec - On demand	0 19 *** Etc/UTC	test_perf	Passed	Inactive			
All test - Orchestra360 - 2.10 a.m all days	10 2 *** Etc/UTC	version_23_7	Passed	in 15 hours			
All test - Qualité - 6.15 a.m all days	15 6 *** Etc/UTC	master	Passed	in 19 hours			
All tests - Qualité Old -STOPPED (4:15 a.m all days)	15 4 *** Etc/UTC	master_old	Passed	Inactive			

Figure 18 - Pipeline Scheduler sur GitLab

Analyse des résultats et corrections

- Une fois le pipeline lancé, il faut attendre environ 3 heures pour obtenir les résultats.
- Si des tests échouent, cela signifie qu'une régression a été introduite. Je dois alors identifier le problème et apporter les corrections nécessaires.
- En dessous sont en jaune les tests qui sont anormalement en echec qu'il faut corriger.

Allure Behaviors

order name duration status Status: 4 11 313 0 10 Marks: [Icons]

> Administration				2	1	107
> B2B				17	2	
> Back office				14		
> One module				8		
▼ Reservation				2	10	144
> Activity				16		
> Payment				72		
> Prepackage				7		
> Product				7		
▼ Transport				2	10	42
> Additional bags				4		
> Airline integration mapping				1	18	6
> Book or Issue choice				3		
> Edit Search				3		
> Engine return				1	1	5
> Filtering and sorting				8		
> Group Branded Fares				1		
> Navigation				2	2	
> Reservation				1	3	
> Tarifs				22	2	

Figure 19 - Allure-Sandbox avec des tests en échec

Merge sur Master et affichage sur Allure

- Après validation et correction des bugs, je peux merger la branche sur Master.
- Le graphe du repository ci-dessous montre que notre branche `feature/Transport` a été fusionnée sur Master.
- Il y a qu'un seul commit par ce que lors du merge request je avons squashed les commits ce qui permet d'avoir qu'un seul commit pour plus de lisibilité.

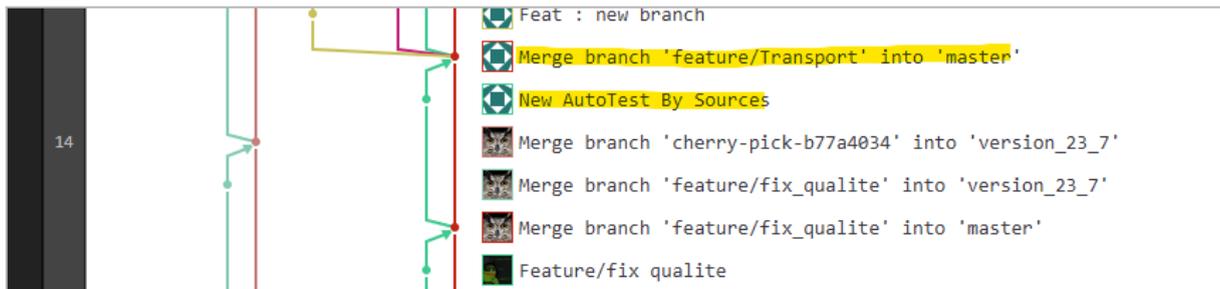


Figure 20 - Repository graph GitLab montrant la fusion des commits

- Enfin, une fois la merge request acceptée, les tests sont automatiquement exécutés le lendemain, ou peuvent être lancés manuellement pour afficher les résultats finaux sur Allure.
- On peut voir en jaune que les tests qui ne passer pas avant, sont maintenant en succès car je les avons corrigé.

order	name	duration	status	Status	Marks
>	Administration			2 1 125	
>	B2B			17 2	
>	Back office			1 5 8	
>	One module			9	
▼	Reservation			22 148 14	
>	Activity			16	
>	Payment			72	
>	Prepackage			7	
>	Product			7	
▼	Transport			22 46 14	
>	Additional bags			4	
>	Airline integration mapping			20 5	
>	Book or issue choice			3	
>	Edit Search			3	
>	Engine return			7	
>	Filtering and sorting			9	
>	Group Branded Fares			1	
>	Navigation			4	
>	Reservation			4	
>	Seat map			2 11 9	
>	Tarifs			22 2	

Figure 21 - Allure, branche Master

IV - Méthodologie et organisation du projet

4.1 - Organisation du travail dans l'entreprise

Organisation du projet

Orchestra utilise Jira pour organiser ses projets, ce qui a également été le cas pour notre travail. Cependant, comme le projet était expérimental, les fiches issues des tests manuels étaient souvent incohérentes ou incomplètes. Je les ai donc peu suivies et elles me servaient principalement à identifier les informations à tester. C'est mon maître de stage qui définissait les directives à suivre et qui modifiait les fiches en conséquence.

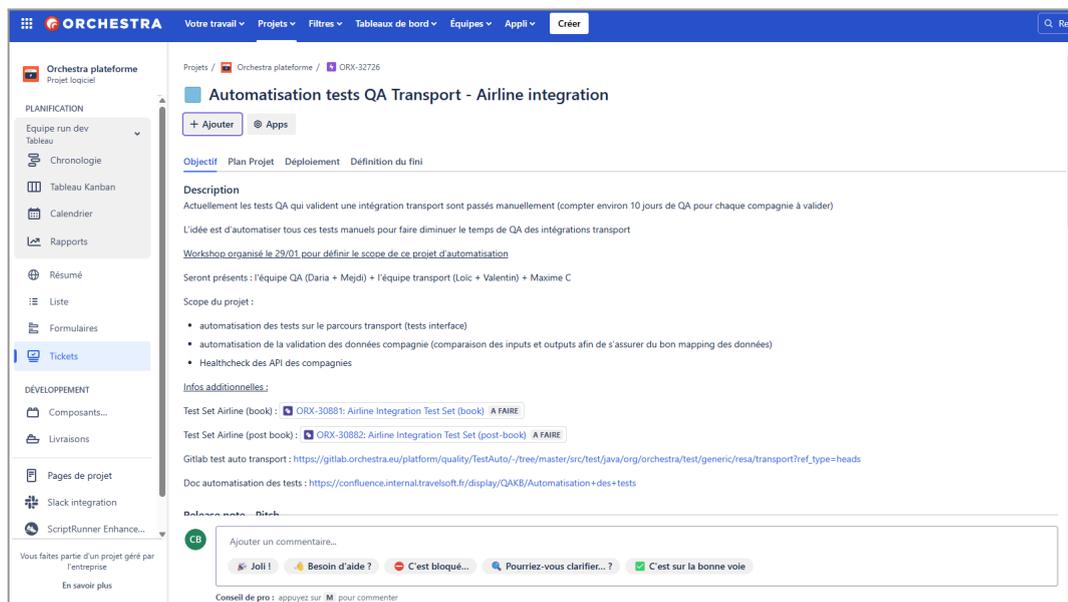


Figure 22 - Jira, fiche projet test auto airline intégration

À mi-parcours du projet, j'ai créé un tableau Excel pour faciliter le suivi des tests effectués sur la page, planifier les prochaines modifications et suivre l'avancement des tests pour chaque compagnie aérienne.

Airline information								
Airline	N° Test	NDC Version	Status	Referent	A / D Durée Vol Journey	A / D Durée Vol Segment	Escale Segment	A / D Date Segment
Air France	001	18.2	IN PROGRESS	Cyprien / Rhys	✓	✗	✗	✓
Etihad	002	21.3	LIVE	Cyprien / Rhys	✓	✗	✗	✓
Xiamen	003	21.3	IN PROGRESS	Cyprien / Rhys	✓	✗	✗	✓
Lot	004	21.3	IN PROGRESS	Cyprien / Rhys	✓	✗	✗	✓

Figure 23 - Tableau de suivi Excel

Enfin, pour être informés des actualités de l'entreprise, j'utilise Slack. Cela me permet de suivre les modifications en cours sur un service, les redémarrages ou autres événements techniques, et aussi de communiquer avec les membres de l'entreprise.

alert-test-auto

Messages Ajouter un canevas Fichiers Marque-pages

allure_notifications_app (APPLI) 11 h 26

Results:

- Environment: Developpement
- Comment: Qualite test auto
- Duration: 04:02:54.155
- Total scenarios: 395
- Total passed: 328 (83 %)
- Total failed: 20 (5,1 %)
- Total broken: 28
- Total unknown: 19

Report available at the link: <https://allure-qualite.internal.travelsoft>

Qualite

395

- 328 passed (83%)
- 20 failed (5.1%)
- 28 broken
- 19 unknown

Envoyer un message #alert-test-auto

Figure 24 - Interface Slack

Organisation du versionning du projet

Pour la gestion du versionning, j'utilise GitLab. j'ai travaillé sur la branche feature/Transport, sur laquelle j'ai développé notre projet. A chaque étape majeure, j'effectue un processus de merge request. Avant toute validation, j'exécute les tests sur Allure-SandBox pour détecter d'éventuelles régressions. Si aucun problème n'est détecté, Daria ou Valentin, nos responsables, valident la merge request et fusionnent notre travail sur la branche Master.

Voir [\[Rapport Technique - 3.5\]](#) pour plus de détails.

Organisation de l'équipe

Chaque matin, avec Rhys, le deuxième stagiaire, je fais une réunion où on discute des problèmes rencontrés et des tâches à accomplir durant la journée. Ensuite, Valentin, après avoir terminé son daily meeting, vient me voir pour suivre notre avancement, préciser les directives et s'assurer que je vais dans la bonne direction. Cela me permet d'avoir un cadre de travail clair et structuré, tout en bénéficiant de l'encadrement de nos responsables.

Apprentissages critiques

Je pense que tout ce que j'ai produit, mis en place et réalisé s'inscrit dans la Compétence 5. Cela inclut AC 2, où j'ai dû formaliser les besoins en structurant les tests et créant une documentation claire pour faciliter la compréhension et l'intégration des nouvelles compagnies. De plus, cela correspond à AC 4, car j'ai mis en place un suivi de projet efficace, en utilisant des outils comme Jira pour le suivi des tâches et Allure-SandBox pour assurer la qualité des tests avant chaque fusion.

4.2 - Méthodes de développement

Durant le projet, j'étais autonome et je devais identifier les différentes solutions possibles avant de choisir la plus adaptée à mettre en place. Une fois notre choix fait, je propose la solution à notre maître de stage, Valentin, avant de procéder à son implémentation.

À chaque difficulté rencontrée, je demandais souvent de l'aide aux membres de l'équipe [QA](#), qui étaient assis à côté de je. Lorsque cela était nécessaire, je préférais aller directement voir la personne concernée plutôt que d'envoyer un message, afin de favoriser la discussion et mieux expliquer mon problème. Bien sûr, lorsque la personne était en télétravail, j'utilisais la messagerie interne.

J'ai démarré ce projet sans aucune base et j'ai dû expérimenter pour trouver les solutions permettant d'automatiser les tests. Par exemple, l'accès à Tracer a été entièrement mis en place par eux-mêmes. j'ai conçu l'architecture des fichiers et réalisé leur implémentation.

je dispose d'une grande liberté dans notre travail. Souvent, je prenais l'initiative d'effectuer du refactoring afin d'optimiser l'ensemble des tests, même lorsque cela impliquait de modifier des fichiers extérieurs au projet. La seule exigence de nos responsables était de ne pas introduire de régressions, ce qui me laissait une marge de manœuvre importante pour améliorer le code et assurer une meilleure maintenance des tests.

Apprentissages critiques

Tout au long du projet, j'ai su m'intégrer dans l'équipe informatique en comprenant pleinement mon rôle et mes missions, tout en collaborant étroitement avec mes collègues. J'ai régulièrement échangé sur l'avancement de mes tâches, en rendant compte de mes actions et en m'assurant que mes contributions étaient alignées avec les objectifs de l'équipe.

Cela correspond à la Compétence 6, AC 2 Intégrer une équipe informatique et AC 3 Mobiliser les compétences interpersonnelles.

4.3 - Travail effectué en plus du projet principal

Création documentation sur confluence

Durant notre projet, j'ai créé une documentation sur Confluence afin de faciliter la compréhension du projet pour toute personne amenée à travailler dessus à l'avenir. J'ai également rédigé une notice détaillant le processus d'intégration d'une nouvelle compagnie. Cette documentation permet à n'importe quel développeur d'ajouter une compagnie sans risque de régression sur les autres tests, garantissant ainsi la stabilité et l'évolutivité du projet.

Conclusion

Le projet que j'ai développé est totalement repris et est d'ailleurs déjà utilisé par l'entreprise. Récemment, un développeur de l'équipe transport a ajouté des tests pour une compagnie qui renvoie du JSON, ce qui montre que notre travail s'intègre bien dans les processus existants. Il est probable que notre projet soit encore amélioré par les développeurs et utilisé quotidiennement par l'équipe [QA](#) pour vérifier la non-régression des fonctionnalités.

Notre projet a principalement apporté une amélioration au suivi des compagnies aériennes en fournissant des informations plus précises sur chacune d'elles et en permettant de détecter plus facilement les régressions. Grâce à l'automatisation des tests et à l'intégration avec Allure, l'équipe dispose désormais d'une meilleure visibilité sur les résultats des tests et l'état du système.

Ce projet m'a permis d'apprendre plusieurs choses essentielles en lien avec le référentiel de compétences du [BUT](#) informatique. Tout d'abord, j'ai appris à utiliser Selenium, notamment la manipulation des WebElements à travers les [XPath](#) pour interagir avec l'interface utilisateur. Cet apprentissage a été particulièrement intéressant et m'a donné envie de réaliser des projets personnels en automatisation de tests.

Ensuite, j'ai découvert la dynamique de travail au sein d'une moyenne entreprise, bien différente des projets de groupe réalisés en cours. J'ai appris à mieux communiquer, à gérer mes disponibilités ainsi que celles de mes collègues, et à identifier la bonne personne à qui poser mes questions en fonction des responsabilités de chacun.

J'ai également approfondi mes compétences en tests logiciels, notamment en réalisant des tests complexes avec une liaison à Allure. Cela m'a permis de mieux comprendre comment structurer des tests efficaces et obtenir des rapports clairs et exploitables.

Enfin, j'ai acquis une meilleure organisation et structuration d'un projet informatique. J'ai appris à respecter des conventions de nommage pour les répertoires et les classes, à écrire des méthodes et des variables en minuscule, et à rédiger les commentaires en anglais. De plus, j'ai intégré l'importance de réduire la duplication de code et de favoriser la généricité pour rendre le projet plus maintenable et évolutif.

Vu le 27/03/2025

Valentin Claudel

Bibliographie

[1] ChatGpt, chatgpt.com. Consulté le: 21 mars 2025. [En ligne].
Disponible sur: <https://chatgpt.com>

[2] Orchestra, orchestra.eu. Consulté le: 1 février 2025 [En ligne].
Disponible sur: <https://orchestra.eu/>

Annexes

Table des annexes

ANNEXE n°1 - Page de résultat des vols, Carrefour voyages.....	32
ANNEXE n°2 - Page de résultat des vols, Leclerc voyages.....	33
ANNEXE n°3 - Page Transport Result, liste des vols disponible.....	33
ANNEXE n°4 - Page Transport Result, information sur le vole.....	34
ANNEXE n°5 - Page Transport Result, information sur les prix.....	35
ANNEXE n°6 - Vidéo d'exemple d'un test complet.....	35
ANNEXE n°7 - Aide à la rédaction.....	35

ANNEXE n°1 - Page de résultat des vols, Carrefour voyages


0 805 41 41 21
Appel gratuit

RETOUR AU PRODUIT
DEVIS
PAIEMENT
CONFIRMATION

1 Répartition des voyageurs

Chambre 2 adultes inclus

2 x Chambre 1 adulte supplément de 318,70 €

2 Catégorie d'hébergement

Chambre Double Vue Jardin inclus

Chambre Double Vue Piscine supplément de 79,68 €

Chambre Double Vue Mer supplément de 111,55 €

3 Type de pension

Formule tout inclus inclus

4 Transport Voir plus de choix

Transport inclus inclus

ITA Airways

Vendredi 17 octobre Economique

Aller

CDG	1 escale	TUN
Paris	4h25	Tunis
06:10		09:35

Retour

TUN	Direct	ORY
Tunis	2h25	Paris
06:20		09:45

sans supplément

ITA Airways

Vendredi 17 octobre Economique

Aller

CDG	1 escale	TUN
Paris	4h25	Tunis
06:10		09:35

Retour

TUN	Direct	ORY
Tunis	2h25	Paris
07:50		11:15



Hôtel Lella Baya 4*
Tunis, Tunisie

Référence interne	237003
Date de départ	Vendredi 17 octobre 2025
Date de retour	Jeu di 23 octobre 2025
Durée	7 jours / 6 nuits
Ville de départ	Paris
Voyageurs	2 adultes

Devis

Séjour et prestations incluses	838,92 €
Activités	inclus
Frais de dossier	20,00 €
Taxes	307,28 €
TOTAL	1 166,20 €

Facilités de paiement

En plusieurs fois avec **Sofinco**

3X Apport 407,39 € + 2 x **388,73 €** i
 Cout du financement 18,65 €

4X Apport 319,53 € + 3 x **291,55 €** i
 Cout du financement 27,98 €

32

ANNEXE n°2 - Page de résultat des vols, Leclerc voyages


0825 884 620*

RETOUR À L'OFFRE
VOTRE CHOIX
PARTICIPANTS
RÉCAPITULATIF
PAIEMENT
CONFIRMATION

Votre choix de voyage

- ✓ Retrouvez ci-dessous les éventuelles options de voyage proposées: repas, transferts, bagages, location de voiture...
- ✓ Sélectionnez l'assurance adaptée à vos besoins.
- ✓ Passez à l'étape suivante pour saisir vos coordonnées et l'ensemble des voyageurs.

1 Répartition des voyageurs

Chambre 2 adultes

2 x Chambre 1 adulte supplément de 210,24 €

2 Catégorie d'hébergement

Bungalow Pionnier RC5 i

3 Prestations incluses

Entrées aux 2 Parcs Disney pour la durée du séjour
 Pour 2 adultes

Etape suivante >



Disney Davy Crockett Ranch : Séjournes au coeur de la forêt !

Marne la Vallée Paris, France

Référence interne	782353
Date de départ	Samedi 29 mars 2025
Date de retour	Dimanche 30 mars 2025
Durée	2 jours / 1 nuit
Sans transport	
Voyageurs	2 adultes

Devis

Séjour et prestations incluses	567,36 €
Taxes	15,00 €
TOTAL	572,36 €

ANNEXE n°3 - Page Transport Result, liste des vols disponible

Etihad Airways
Dimanche 27 avril
Econ. 15

Aller



10:20

CDG
Paris

1 escale
21h25

LHR
Londres

06:45

Ce transport arrive 1 jour après le départ

Etihad Airways
Dimanche 4 mai
Econ. 15

Retour



14:50

LHR
Londres

1 escale
15h55

CDG
Paris

07:45

Ce transport arrive 1 jour après le départ

7 206,40 €

Prix total i

Sélectionner →

Economy Deluxe

Voir plus ▾

ANNEXE n°4 - Page Transport Result, information sur le vole

i Information
×

Aller

Ce transport arrive 1 jour après le départ

Billet émis par Etihad Airways

Trajet avec un changement. La durée totale du trajet est de 21h25

Etihad Airways Public

Dimanche 27 avril Durée : 6h40

10:20 **Paris**

Paris - (Aéroport Ch. De Gaulle)

Terminal : 2E

Classe: Économique - 5

N° de vol: EY32

Bagages en soute inclus

19:00 **Abu Dhabi**

Terminal : A

Escale de 7h05

Etihad Airways Public

Lundi 28 avril Durée : 7h40

02:05 **Abu Dhabi**

Terminal : A

Classe: Économique - 5

N° de vol: EY61

Bagages en soute inclus

06:45 **Londres**

London (Heathrow Airport)

Terminal : 4

Fermer

ANNEXE n°5 - Page Transport Result, information sur les prix

i Détails des prix		X
Passagers	Adultes	
Tarifs	3 274,00 €	
Taxes	329,20 €	
Frais	0,00 €	
Prix	3 603,20 €	
Nb	x2	
Total	7 206,40 €	

Prix du billet aérien par passager, taxes incluses.

ANNEXE n°6 - Vidéo d'exemple d'un test complet

[\[Lien de la vidéo sur Youtube\]](#)

ANNEXE n°7 - Aide à la rédaction

Ce rapport a été rédigé avec l'aide d'un outil d'intelligence artificielle pour la reformulation et la correction orthographique. Les contenus techniques et les analyses proviennent exclusivement de mon travail réalisé durant le stage. [1]