Du 08 Janvier au 23 Février 2024

# RAPPORT DE STAGE


PAUL-ANTOINE CARREAU
BTS SIO 2EME ANNEE OPTION SLAM

# Sommaire

## Remerciements

Tout d’abord, je tiens à remercier Vincent HEMERY, mon tuteur de stage qui m’a permis d’effectuer mon stage à Odysséo.

Je souhaite aussi remercier toute l’équipe de HTTP qui m’a chaleureusement accueilli dans ses bureaux.

## Introduction

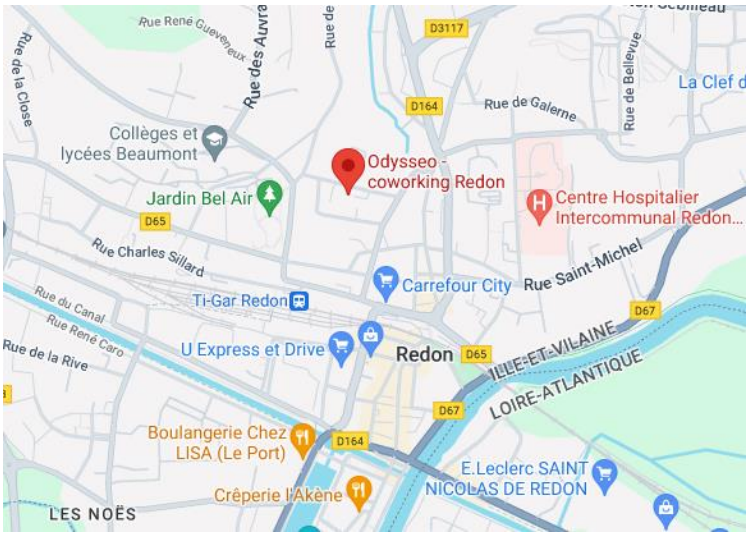
Ce rapport présente mon expérience de stage au sein de l’entreprise Odysséo. Le stage s’est déroulé du 8 janvier au 23 février 2024 dans le cadre de ma formation de BTS SIO option SLAM.

## Présentation de l’entreprise

Odysséo est un espace professionnel offrant une gamme de services de domiciliation, de location de bureaux fermés, de salles de réunion et d'espace de coworking. **L’entreprise a été fondé en novembre 2019.**

Pour la gestion des réservations, Odysséo utilise la plateforme **Archie App**, permettant une gestion efficace des réservations de bureaux, salles de réunion et espaces de coworking. Cette solution facilite la réservation et le paiement des services proposés, offrant ainsi une expérience client fluide et transparente.

Par ailleurs, Odysséo intègre également la technologie **WelcomR** pour la gestion des accès. Cette solution innovante permet aux clients d'accéder facilement aux locaux de l'entreprise à l'aide de leur smartphone, garantissant ainsi une sécurité et une praticité lors de leurs visites.




L'espace se trouve au 7 rue Saint-Conwoïon à Redon

## Cadre de travail

Horaires : Du lundi au vendredi de 8 h 30 à 12 h et de 13 h 30 à 17 h

J'étais dans les locaux de HTTP, une entreprise voisine qui appartient à Vincent HEMERY.

Je travaillais seul car il n'y a pas d'équipe informatique à Odysséo.

## Objectifs du stage

L'objectif du stage était de créer 2 affichages dynamiques pour l'espace de coworking.

Le premier affichage, qui se trouve à l'intérieur de l'espace, permet d'afficher le plan de l'espace avec des indications pour les différentes salles de réunion, un affichage des réservations en cours et à venir et un message de bienvenue pour les nouveaux coworkers.

Le deuxième affichage se trouve devant l'entrée de l'espace. Il affiche un message concernant l'ouverture de la porte. En dessous de ce message, se trouve la liste des coworkers et des membres de l'espace ainsi que leur numéro afin que les clients puissent les contacter pour ouvrir la porte s'ils ont un rendez-vous.

# Mission

## Environnement

Pour mener à bien mon projet, j'ai d'abord rédigé les différentes étapes que je devais suivre pour développer les affichages.

Dans un premier temps, j'ai commencé par m'immerger dans l'environnement de travail pour comprendre le fonctionnement du système de réservation. J'ai découvert que l'entreprise utilise une application appelée Archie App pour gérer les réservations dans l'espace de coworking. Cette application permet à l'administrateur de saisir les détails des salles, des bureaux et des chaises disponibles, ce qui facilite ensuite les réservations et les paiements pour les clients. Les données relatives aux réservations sont stockées sous forme de fichiers CSV, accessibles via l'interface de l'application.

## API

Ensuite, j'ai exploré l'utilisation de l'API fournie par Archie App pour récupérer les données de réservation. Pour cela, il faut créer un compte développeur sur l'espace Odysseo du logiciel Archie App. Ensuite, une clé d'identification et une clé secrète nous sont données, elles permettent de récupérer un TOKEN, qui lui, permet d'obtenir les données que l'on souhaite. Ce token à une durée de validité, il doit donc être renouvelé. Cependant, je me suis rapidement heurté à des difficultés, car l'API n'était pas correctement configurée pour récupérer les données dans un intervalle de dates spécifique. J'ai donc contacté le support technique d'Archie App pour résoudre ces problèmes d'API et obtenir les autorisations nécessaires pour accéder aux données de manière plus flexible.

Après avoir résolu les problèmes d'API, j'ai identifié les fichiers "utilisateur" et de "réservation" comme les principales sources de données nécessaires à mon projet. Le fichier utilisateur contenait des informations sur les clients, tandis que le fichier de réservation contenait des détails sur les réservations de salles et de bureaux.

Dans le cadre de cette exploration, j'ai testé l'API en utilisant un script Python fourni dans la documentation de l'API. Ce script m'a permis de comprendre les requêtes nécessaires pour récupérer les données et de vérifier la cohérence des informations retournées par l'API.

## Outils pour la création de l'affichage

J'ai entrepris des recherches approfondies pour explorer les différentes options disponibles pour développer un affichage dynamique. J'ai examiné divers logiciels et plates-formes disponibles sur Internet, en tenant compte de leur maintenabilité, de leur flexibilité et de leur adaptabilité à mes besoins spécifiques.

Après avoir évalué plusieurs options, j'ai conclu que le meilleur moyen de créer l'affichage dynamique était d'utiliser les langages de programmation Web tels que HTML, CSS et JavaScript. Cette décision a été prise en tenant compte des exigences particulières de mon projet, notamment la récupération des données de l'API d'Archie App. Ces langages offraient la flexibilité nécessaire pour interagir avec les données de manière dynamique et les afficher de manière convaincante sur l'interface utilisateur. De plus, cette approche est plus enrichissante d'un point de vue de ma formation, car elle me permettait de maîtriser pleinement le processus de développement et d'améliorer mes compétences dans ces langages de programmation.

En mettant en pratique cette décision, j'ai créé un projet HTML, CSS et JavaScript que j'ai hébergé sur un serveur local à l'aide du logiciel WAMP. Cela m'a permis de développer et de tester l'affichage dynamique.

## Récupération des données et tri

J'ai d'abord tenté de récupérer les données en utilisant JavaScript, mais j'ai rapidement rencontré des erreurs en raison des restrictions CORS imposées par l'API. J'ai essayé de contourner les erreurs avec des outils JavaScript, mais cela ne fonctionnait pas. Pour éviter ce problème, j'ai choisi d'utiliser Python pour récupérer les données à partir des fichiers CSV fournis par l'API.

J'ai fait les requêtes à l'API pour récupérer le fichier des réservations. La requête se fait en utilisant l'Endpoint : c'est l'adresse de base pour requêter l'API.

Celle de mon application était : https://api.archieapp.co/v1/spaces/odysseo-1/

Pour récupérer le fichier réservations, il faut utiliser l'Endpoint en mentionnant la date de début et date de fin de la période pour les réservations :

```
https://api.archieapp.co/v1/spaces/odysseo-1/bookings/exportCSV?startDate={start_date}&endDate={end_date}
```

« start_date » et « end_date » sont des variables qui seront initialisées sur « datetime.now().date() ». Cela permet de récupérer toutes les réservations du jour même uniquement.

```
client_id = "8e5dba0e-59c1-4ff5-bf2b-335a722aa858"
client_secret = "[redacted]"

today = datetime.now().date()

start_date = today.strftime("%Y-%m-%d")
end_date = today.strftime("%Y-%m-%d")
```

Voici les variables initialisées pour définir la clé ID et la clé secret pour l'API. Il y a aussi l'initialisation pour l'intervalle de la requête « réservation ».

Une fois les données récupérées, j'ai développé des fonctions en Python pour trier et nettoyer les données. Cela incluait la suppression des virgules superflues dans les adresses postales, ainsi que la suppression de la première ligne qui contenait uniquement le séparateur de colonnes. Cette étape de nettoyage des données était essentielle pour garantir l'intégrité et la qualité des données utilisées dans mon application.

```
import csv
def remplacer_valeurs_csv(chemin_fichier):
    # Ouvrir le fichier CSV en mode lecture
    with open(chemin_fichier, 'r', encoding='utf-8') as fichier:
        # Utiliser le module csv pour lire le fichier
        lecteur_csv = csv.reader(fichier, delimiter=',')

        # Convertir le lecteur en liste pour manipuler les lignes
        lignes = list(lecteur_csv)

    # Remplacer la valeur spécifique dans chaque ligne
    for i in range(len(lignes)):
        lignes[i] = [v.replace('7, rue st conwoion', '7 rue saint conwoion') for v in lignes[i]]

    # Réécrire le contenu modifié dans le même fichier
    with open(chemin_fichier, 'w', encoding='utf-8', newline='') as fichier:
        # Utiliser le module csv pour écrire dans le fichier avec une virgule comme séparateur
        ecrivain_csv = csv.writer(fichier, delimiter=',')
        ecrivain_csv.writerows(lignes)
```

Voici un exemple de fonction. Celle-ci permet de retirer la virgule pour l'adresse de l'espace de coworking.

## Automatisation du script python

Maintenant, que le script python récupère les données et les nettoie, il faut que le script soit exécuté plusieurs fois par jours pour que les données soit à jour. J'ai donc créé un script JS qui s'appelle *server.js* qui permet d'exécuter le script python toutes les heures. Ce script JS utilise « child_process » qui permet d'exécuter des processus pendant le script JS. En l'occurrence, lancer le script python en mettant un intervalle de temps avec `setInterval(() =>` en mettant le temps en millisecondes, donc 3 600 000 pour mettre 1 h.

## Implémentation des données

Après avoir automatisé la récupération et le tri des données, il fallait implémenter les données dans une page HTML CSS. Pour cela, j'ai fait un script JS qui récupère les données dans les fichiers CSV. Pour que les données soient mises à jour sans l'actualisation du site, j'ai utilisé des requêtes AJAX.

```
function loadData2() {
    // Charger les données des utilisateurs
    $.ajax({
        url: '/odysseo-coworking/data/data_users.csv',
        dataType: 'text',
        success: function (userData) {
            userCsvData = userData; // Assigner les données des utilisateurs à la variable globale
            // Charger les données de réservation pour chaque salle
            $.ajax({
                url: '/odysseo-coworking/data/data_bookings.csv',
                dataType: 'text',
                success: function (reservationData) {
                    // Afficher les nouveaux coworkers du jour avec réservations
                    displayNewUsersWithReservations(reservationData);
                }
            });
        }
    });
}
```

Voici un exemple d'une fonction JavaScript pour charger les données. Le « $.ajax » permet d'appeler l'outil AJAX .

Pour tester l'implémentation de données, j'ai créé un tableau ou j'ai rangé toutes mes réservations pour voir si les données étaient parfaitement importées et triées. Le test était concluant.

## Création des affichages

Dans le cadre de la création d'affichages, j'ai créé un projet utilisant HTML, CSS et JavaScript pour chaque affichage nécessaire. Cela m'a permis de construire des interfaces utilisateur dynamiques et interactives, adaptées aux besoins spécifiques de l'espace de coworking.

### Affichage des réservations

J'ai développé des fonctions en JavaScript dans le but d'afficher les réservations actuelles et futures des salles de réunion. Cette phase de développement s'est déroulée en plusieurs étapes. Tout d'abord, j'ai créé une fonction qui récupérait les données des réservations des salles de réunion, les triait par nom de salle, et vérifiait leur validité. Ensuite, j'ai élaboré un mécanisme pour générer du code HTML correspondant à chaque réservation. Pour ce faire, j'ai mis en place des conditions : si la réservation était à venir, le code HTML affichait le nom de l'entreprise, l'heure de la réservation et un message indiquant qu'elle était à venir. Si la réservation était en cours, le code HTML affichait également le nom de l'entreprise et l'heure de la réservation, tout en calculant le temps restant de la réunion.

Pour cet affichage, j'ai aussi créé une fonction qui permettait de vérifier si des nouveaux coworkers avait une réservation pour afficher un message de bienvenue. Cette fonctionnalité était plus dure à développer, car je devais récupérer les informations de 2 fichiers CSV différents.


Affichage de la réservation et décompte du temps restant et affichage du message de bienvenu

### Affichage de la liste des membres

Pour l'affichage vertical, du message informatif et de la liste des membres de l'espace de coworking, j'ai mis en place une fonction spécifique dans mon projet. Cette fonction était chargée de récupérer les données pertinentes depuis le fichier "user". Cependant, la récupération des membres n'était pas aussi simple qu'il n'y paraissait. En effet, dans le fichier CSV, les critères pour déterminer si un client était membre de l'espace de coworking étaient basés sur deux combinaisons de paramètres.

La première combinaison de paramètres était la suivante : si le "segment1" était égal à "Membre", si le rôle n'était pas défini (null), et si le statut "archived" était à "false".

La deuxième combinaison de paramètres était : si le rôle était égal à "manager" et si le statut "archived" était à "false".

Ces conditions ont rendu la vérification un peu complexe, mais une fois que j'ai compris les critères nécessaires, j'ai pu développer la fonction appropriée pour récupérer avec succès la liste des membres de l'espace de coworking.

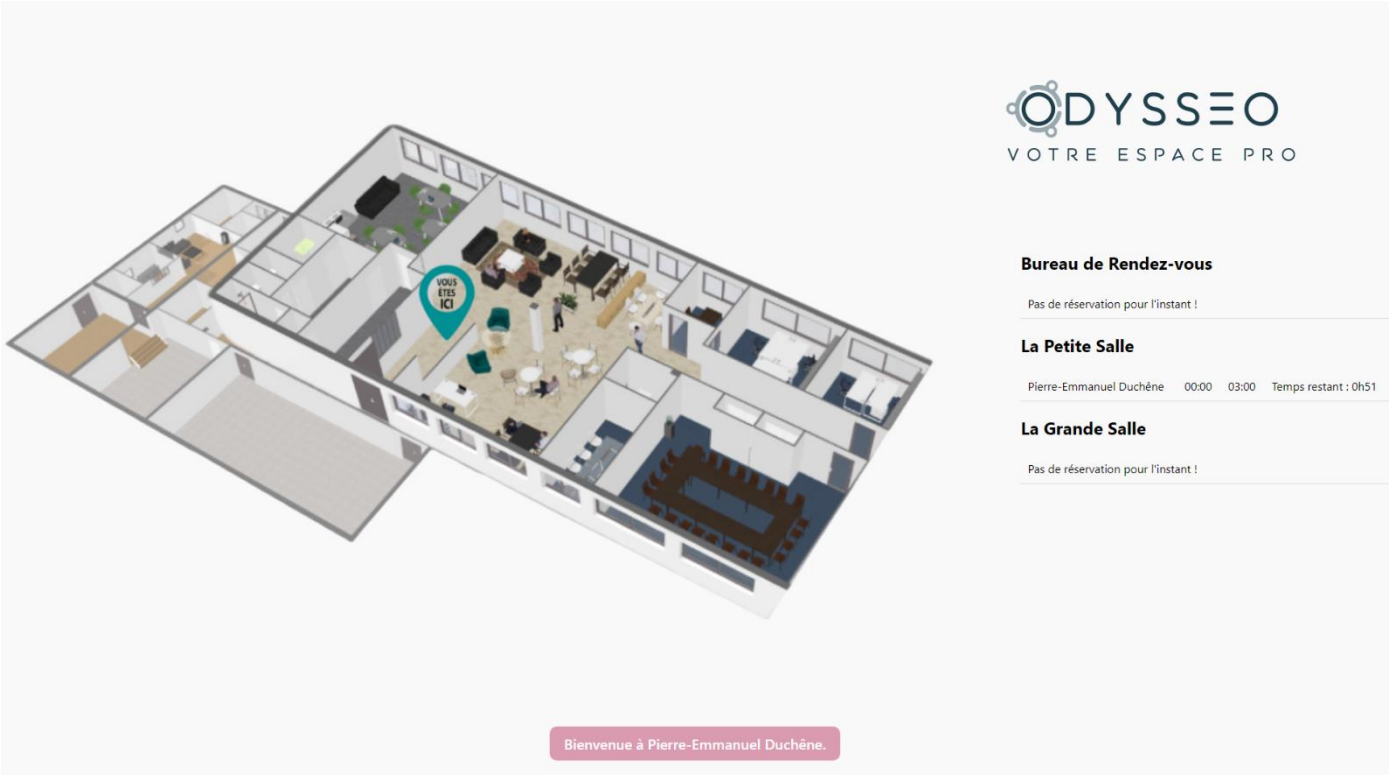
Affichage sur le site du tableau des membres et de leur numéro

| Nom de l'entreprise | Numéro de téléphone |
|---|---|
| High Tech for Telecontrol Project | |
| CHRISTOPHER GUILLOU | 06 72 89 05 91 |
| HTTP | 06 86 71 53 67 |
| Alter ecoh | 06 82 43 09 01 |
| Fanny Perron | 06 88 48 25 64 |
| EO | 06 95 70 50 78 |
| Wimoov Grand Ouest | 06 74 92 88 04 |
| WALL STRUCTURE | 06 58 88 56 17 |
| PLANOE | 06 46 29 04 80 |

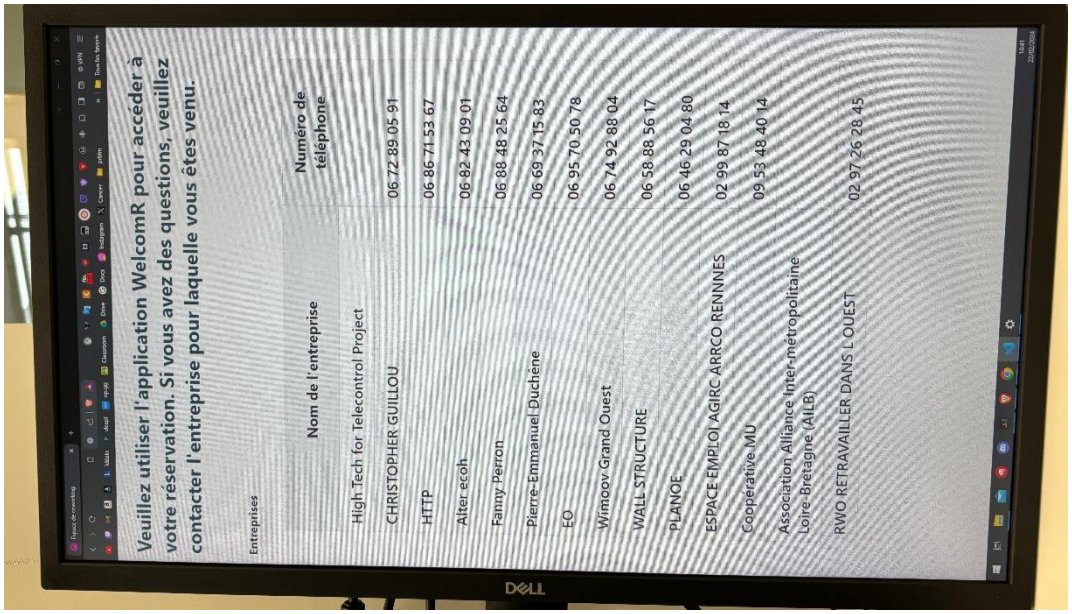
## Mise en forme des sites et solution de maintenabilité

Une fois les fonctionnalités principales développées, j'ai consacré du temps à la mise en forme des deux affichages, en ajustant les images, les différents éléments et en veillant à ce que l'interface utilisateur soit conviviale et attrayante. Parallèlement, j'ai rédigé un guide détaillé expliquant à mon client comment installer l'application, sa structure et les étapes à suivre pour assurer sa maintenance à long terme. Ce guide comprenait des instructions claires sur les prérequis techniques, les commandes à exécuter, ainsi que des recommandations pour résoudre les problèmes courants et effectuer des mises à jour. Cette approche proactive visait à garantir que l'application reste fonctionnelle et facilement maintenable même après la fin de mon stage.

# Rendu final




Affichage des réservations




Affichage vertical pour l’entrée

## Conclusion

Ce stage m'a offert une expérience enrichissante dans le domaine du développement web et de la gestion de projet. En travaillant sur la création d'un affichage dynamique pour un espace de coworking, j'ai pu mettre en pratique mes compétences en programmation et en analyse de données.

La résolution des problèmes techniques liés à l'API et à la récupération des données a été une étape clé de ce projet. En utilisant une approche itérative, j'ai surmonté les obstacles et trouvé des solutions efficaces pour récupérer, nettoyer et afficher les données de manière dynamique.

La création des affichages en HTML, CSS et JavaScript m'a permis d'améliorer mes compétences en développement front-end. J'ai également acquis une meilleure compréhension des bonnes pratiques de codage et de la gestion de projet.

En rédigeant un guide de déploiement et de maintenance, j'ai renforcé ma capacité à communiquer des informations techniques de manière claire et accessible, tout en me sensibilisant à l'importance de la documentation et de la planification pour assurer la pérennité des projets informatiques.

Ce stage a été une expérience stimulante qui m'a permis de développer mes compétences techniques et de renforcer mes capacités de résolution de problèmes. Je suis reconnaissant pour cette opportunité et confiant dans ma capacité à appliquer les connaissances et les compétences acquises dans mes futurs projets professionnels.

# Annexes

**Plan de développement du projet :**

1. Compréhension des données ArchieApp :

- Familiarisez-vous avec la structure des données exportées par ArchieApp au format CSV.
- Identifiez les informations nécessaires telles que l'état d'occupation des salles, les noms des clients, les identifiants des salles, etc.

2. Choix de la technologie :

- Sélectionnez une technologie pour créer l'affichage dynamique. Les langages comme HTML, CSS, JavaScript avec une bibliothèque comme React ou Vue.js pourraient être appropriés.

3. Analyse des besoins en affichage :

- Conception de l'interface utilisateur (UI) en fonction des besoins spécifiques de votre espace de coworking.
- Choisissez si vous souhaitez afficher un plan de l'espace avec une vue en temps réel ou une mise à jour périodique.

4. Mise en place de l'interface utilisateur :

- Créez une interface utilisateur interactive avec le plan de l'espace de coworking.
- Utilisez des bibliothèques de cartographie ou des outils de création d'interfaces pour représenter les salles et les zones communes.

5. Intégration des données ArchieApp :

- Chargez les données CSV d'ArchieApp dans votre application.
- Utilisez des requêtes AJAX ou une méthode similaire pour obtenir les dernières données d'occupation des salles.

6. Affichage dynamique :

- En fonction des données, mettez à jour l'affichage pour indiquer quelles salles sont occupées par quels clients.
- Utilisez des couleurs, des icônes ou d'autres indicateurs visuels pour rendre l'affichage facilement compréhensible.

7. Gestion des mises à jour en temps réel :

- Mettez en place un mécanisme pour recevoir des mises à jour en temps réel des données ArchieApp (si possible) pour assurer l'actualisation constante de l'affichage.

8. Tests et Débogage :

- Testez l'application avec différentes données d'ArchieApp pour vous assurer que l'affichage fonctionne correctement.
- Assurez-vous que l'interface utilisateur est conviviale et intuitive.

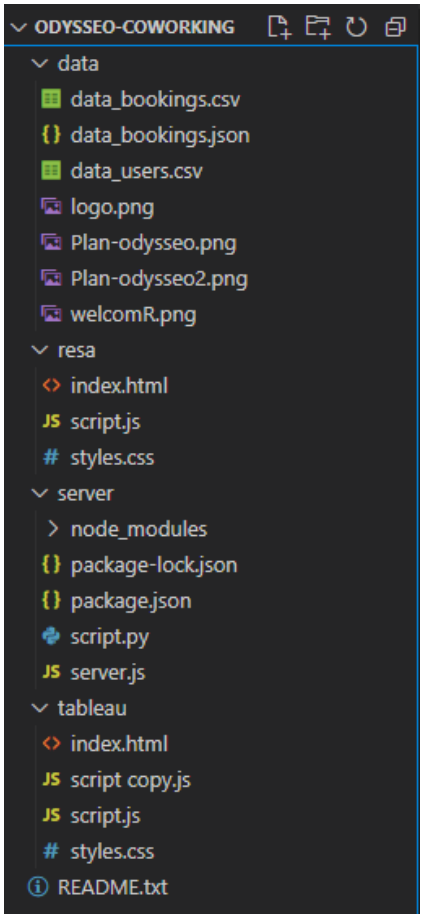
9. Déploiement :

- Déployez l'application dans l'environnement souhaité, que ce soit sur un serveur local, dans le cloud ou sur un dispositif matériel dédié dans l'espace de coworking.

10. Maintenance :

- Assurez-vous de maintenir l'application en fonctionnement en cas de mises à jour d'ArchieApp ou de changements dans les besoins de l'espace de coworking.

**Structure du projet VS Code :**



**La double vérification pour les membres :**

```
if (role === 'manager' && archived.toLowerCase() === 'false') {
    const accountId = userColumns[14].replace(/"/g, ''); // Account ID
    const phoneNumber = userColumns[6]; // Numéro de téléphone

    //console.log(accountId);
    //console.log(phoneNumber);
    // Vérifier si l'Account ID de l'utilisateur a une réservation
    if (reservationAccountIds.includes(accountId)) {
        //console.log('oui');
        // Extraire le companyName à partir du fichier de réservation
        const companyName = getCompanyNameFromAccountId(accountId, reservationCsvData);
        //console.log(companyName);
        activeUsers.push({ companyName, phoneNumber });
    }
}

if (segment1 === 'Membre' && !role && archived.toLowerCase() === 'false') {
    const accountId = userColumns[0].replace(/"/g, ''); // Account ID
    const phoneNumber = userColumns[6]; // Numéro de téléphone

    if (reservationAccountIds.includes(accountId)) {
        //console.log('oui');
        // Extraire le companyName à partir du fichier de réservation
        const companyName = getCompanyNameFromAccountId(accountId, reservationCsvData);
        //console.log(companyName);
        activeUsers.push({ companyName, phoneNumber });
    }
}
```

**Deux fonctions pour charger les deux fichiers CSV :**

```
// Fonction pour charger les données des utilisateurs
function loadUserData(callback) {
    $.ajax({
        url: '/odysseo-coworking/data/data_users.csv',
        dataType: 'text',
        success: function (userData) {
            callback(userData);
        }
    });
}

// Fonction pour charger les données de réservation
function loadReservationData(callback) {
    $.ajax({
        url: '/odysseo-coworking/data/data_bookings.csv',
        dataType: 'text',
        success: function (reservationData) {
            callback(reservationData);
        }
    });
}
```

**Fonction pour ajouter le tableau et les contacts :**

```
function displayEntranceMembers() {
    const membersTable = $('#entranceTable');
    let tableHTML = '<thead><tr><th>Nom de l\'entreprise</th><th>Numéro de téléphone</th></tr></thead><tbody>';

    // Obtenez la liste des utilisateurs actifs
    const activeUsers = getActiveUsers(userCsvData, reservationCsvData);
    // Parcourez la liste des utilisateurs actifs pour afficher leurs informations
    activeUsers.forEach(user => {
        // Vérifiez si le numéro de téléphone est vide ou commence déjà par un zéro
        let phoneNumber = user.phoneNumber;
        if (phoneNumber && !phoneNumber.startsWith('0')) {
            // Ajoutez un zéro au début du numéro de téléphone
            phoneNumber = '0' + phoneNumber;
        }

        // Insérer un espace tous les deux chiffres dans le numéro de téléphone
        let formattedPhoneNumber = '';
        for (let i = 0; i < phoneNumber.length; i++) {
            formattedPhoneNumber += phoneNumber[i];
            if ((i + 1) % 2 === 0 && i !== phoneNumber.length - 1) {
                formattedPhoneNumber += ' ';
            }
        }

        tableHTML += `
            <tr>
                <td>${user.companyName}</td>
                <td>${formattedPhoneNumber}</td>
            </tr>
        `;
    });

    tableHTML += '</tbody>';

    // Afficher le tableau des membres dans la div correspondante
    membersTable.html(tableHTML);
}
```

**Plan de l'espace :**