

NOM	EXACBBDAPDSDP62A_313892_20240530235430
Prénom	EXACBBDAPDSDP62A_313892_20240530235430
Date de naissance	

Copie à rendre

Bloc 3 – Développement d’une solution digitale avec Python

Documents à compléter et à rendre

Lien Github : <https://github.com/SARHIRI83/AppJoDepot.git>

Dossier 1 : Spécifier une solution digitale

1.1 Énumérez toutes les fonctionnalités attendues par le client sous forme d’User Story (agilité)

Vous trouverez ci-dessous les fonctionnalités attendues par le client sous forme d’User Story. Le lien Figma suivant présente les wireframes de ces fonctionnalités pour faciliter le développement du site.

<https://www.figma.com/file/s92JVBP4oGD4JyaSk6LQWc/Untitled?type=design&node-id=0%3A1&mode=dev&t=4lgGEL9gOwVbKrn7-1>

1. User Story: Consulter le site des Jeux Olympiques

En tant que visiteur, je souhaite avoir accès au site « Jeux Olympiques France » pour m’informer des épreuves proposés.

Utilisateur : Visiteur

Cas d'Usage : Découvrir les différentes épreuves du site « Jeux Olympiques »

Contexte : Les visiteurs peuvent chercher les détails de l’évènement (épreuves, prix, date).

Critères de Validation : Les visiteurs doivent facilement naviguer sur le site pour trouver des informations sur les épreuves, les prix, les dates, les lieux, etc.

Prérequis : Il n’y a pas de prérequis spécifique, tous les visiteurs peuvent accéder au site sans besoin de créer un compte.

Actions Possibles :

- Consulter les différentes épreuves des Jeux Olympiques.
- Naviguer à travers les pages du site pour découvrir les détails de chaque épreuve.

Fonctionnalités Déclenchées : Cette action ne nécessite pas de fonctionnalité spécifique déclenchée.

2. User Story: Visualiser des Offres de Tickets

En tant que visiteur, je souhaite visualiser les offres de tickets disponibles solo, duo et familiales.

Utilisateur : Visiteur

Cas d'Usage : Voir les différentes offres de tickets disponibles.

Contexte : Les visiteurs désirent sélectionner le ou les types de billet qui correspond à leurs besoins.

Critères de Validation : La disponibilité et le détail des offres sont clairement visibles par les visiteurs.

Prérequis : Il n’y a pas de prérequis spécifique, tous les visiteurs peuvent accéder aux offres sans connexion.

Référence : EXACBBDAPSDP62A_313892_20240530235430

Actions Possibles :

- Visualiser la disponibilité des différentes offres ainsi que les détails et les tarifs.
- Sélectionner une ou plusieurs offres et l'ajouter au panier.

Fonctionnalités Déclenchées : L'ajout d'un type d'offre au panier déclenche la fonctionnalité de la gestion de la commande.

3. User Story : Sélectionner et ajouter au panier

En tant que visiteur, je désire sélectionner une ou plusieurs offres dont j'ai besoin et l'ajouter à mon panier pour passer une réservation.

Utilisateur : Visiteur

Cas d'Usage : Sélectionner une ou plusieurs offres de ticket et l'ajouter au panier.

Contexte : Les visiteurs souhaitent réserver leurs billets.

Critères de Validation : L'offre et le détail de la sélection sont visibles dans le panier.

Prérequis : Il n'y a pas de prérequis spécifique, tous les visiteurs peuvent accéder aux offres sans connexion.

Actions Possibles :

- Sélectionner une ou plusieurs offres (solo, duo, familiale) pour l'ajouter au panier.
- Visualiser le détail des offres choisies dans le panier.

Fonctionnalités Déclenchées : La gestion du panier déclenche la fonctionnalité de la demande de connexion de l'utilisateur (ou création de son compte) pour poursuivre le processus de réservation.

4. User Story : Gérer des offres (Administrateur)

En tant qu'administrateur, je souhaite pouvoir paramétrer les offres de tickets sur le site pour ajouter, modifier, visualiser ou créer de nouvelles offres.

Utilisateur : Administrateur

Cas d'Usage : Afficher, ajouter, modifier ou créer de nouvelles offres de tickets.

Contexte : Les développeurs du site « Jeux Olympiques » souhaitent paramétrer les offres pour qu'elles soient à la disposition des visiteurs et des utilisateurs.

Critères de Validation : L'administrateur a la possibilité d'effectuer l'ensemble des actions nécessaires liées à la gestion des offres.

Prérequis : Il est nécessaire d'être connecté en tant qu'administrateur.

Référence : EXACBBDAPDSDP62A_313892_20240530235430

Actions Possibles :

- Visualiser les offres existantes.
- Ajouter de nouvelles offres.
- Modifier les détails ou les tarifs des offres existantes.
- Supprimer des offres.

Fonctionnalités Déclenchées : Le paramétrage des offres gère la disponibilité des offres et leur affichage sur le site.

5. User Story : Réserver des billets

En tant que visiteur, je souhaite sélectionner un ou plusieurs types de billet (solo, duo, familial) et transmettre mes informations personnelles pour finaliser ma réservation.

Utilisateur : Visiteur

Cas d'Usage : Sélectionner un billet, transmettre des données personnelles et terminer la réservation.

Contexte : Les visiteurs désirent acheter leurs billets.

Critères de Validation : Le processus de réservation peut être complété par le visiteur sans difficulté.

Prérequis : Les visiteurs doivent s'authentifier et fournir des informations personnelles pour finaliser la réservation.

Actions Possibles :

- Sélectionner le type de billet (solo, duo, familial).
- Renseigner les informations personnelles lors de la réservation.
- Finaliser le processus de réservation en effectuant le paiement.

Fonctionnalités Déclenchées : La réservation des tickets déclenche la procédure de paiement sécurisé.

6. User Story : Gérer les comptes utilisateurs (Administrateur)

En tant qu'administrateur, je souhaite gérer et consulter les données utilisateur pour assurer la sécurité et la conformité.

Utilisateur : Administrateur

Cas d'Usage : Gérer les données utilisateur.

Contexte : L'administrateur souhaite assurer la conformité et la sécurité des données personnelles via un compte.

Critères de Validation : L'administrateur peut accéder aux informations utilisateur de manière sécurisée.

Prérequis : Connexion en tant qu'administrateur nécessaire.

Référence : EXACBBDAPDSDP62A_313892_20240530235430

Actions Possibles :

- Consulter les informations utilisateur.
- Gérer les comptes utilisateur (création, modification, suppression).

Fonctionnalités Déclenchées : Les actions effectuées par l'administrateur instaure un compte utilisateur sécurisé exploitable par le visiteur.

7. User Story : Créer un compte utilisateur

En tant que visiteur, je souhaite créer un compte en fournissant mon nom, prénom, adresse e-mail et mot de passe sécurisé pour accéder à mes réservations.

Utilisateur : Visiteur

Cas d'Usage : Créer un compte en indiquant des données personnelles.

Contexte : Les visiteurs souhaitent continuer leur processus de réservations et sécuriser leur compte.

Critères de Validation : Les visiteurs peuvent s'inscrire et accéder à leur compte.

Prérequis : Récupérer les coordonnées des visiteurs. Les utilisateurs doivent avoir une adresse email valide.

Actions Possibles :

- Remplir le formulaire d'inscription avec le nom, le prénom, l'adresse e-mail et le mot de passe de l'utilisateur.
- Créer un compte utilisateur.

Fonctionnalités Déclenchées : Un e-mail de confirmation est envoyé à l'utilisateur après l'inscription.

8. User Story: Générer automatiquement une Clé au moment de la création du compte (Administrateur)

En tant qu'administrateur, je souhaite qu'une clé soit générée automatiquement au moment de la création du compte utilisateur, et qu'il soit uniquement visible par l'organisation des Jeux Olympiques.

Utilisateur : Administrateur

Cas d'Usage : Générer une clé lors de la création d'un compte utilisateur de manière automatique.

Contexte : L'administrateur souhaite garantir la sécurité des comptes utilisateurs.

Critères de Validation : La génération de la clé se fait de manière transparente et visible uniquement pour l'administrateur.

Prérequis : La clé est générée après que l'utilisateur a créé son compte.

Actions Possibles :

- Générer automatiquement une clé unique au moment de la création du compte utilisateur.

Référence : EXACBBDAPSDP62A_313892_20240530235430

Fonctionnalités Déclenchées : La clé générée sera utilisée pour des procédures de sécurité, et sera accessible uniquement par l'administrateur.

9. User Story : Authentifier l'utilisateur pour sa sécurité

En tant qu'utilisateur, je souhaite être authentifié de manière sécurisée pour finaliser ma réservation et effectuer le paiement.

Utilisateur : Visiteur

Cas d'Usage : Authentifier le visiteur de manière sécurisée pour finaliser la réservation sur son compte.

Contexte : L'administrateur souhaite assurer l'intégrité des transactions et des données utilisateur.

Critères de Validation : Le processus d'authentification évite l'usurpation d'identité.

Prérequis : Connaître son identifiant et son mot de passe. Posséder une adresse email valide.

Actions Possibles :

- S'authentifier en tant qu'utilisateur avec les identifiants créés lors de l'inscription.

Fonctionnalités Déclenchées : Après l'authentification, l'utilisateur peut accéder à ses fonctionnalités spécifiques (réservation, gestion de compte, etc.).

10. User Story: Suivre le processus de paiement sécurisé

En tant qu'utilisateur, je souhaite effectuer un paiement sécurisé en ligne pour confirmer ma réservation de billet.

Utilisateur : Visiteur

Cas d'Usage : Effectuer un paiement sécurisé en ligne pour confirmer la réservation.

Contexte : Le visiteur souhaite une garantie de la confidentialité des transactions financières.

Critères de Validation : Les paiements sont traités de manière fiable et sécurisée.

Prérequis : Avoir sélectionné des billets à réserver.

Actions Possibles :

- Payer avec sa carte bancaire en ligne pour confirmer la réservation des billets.

Fonctionnalités Déclenchées : Après le paiement, les billets réservés sont confirmés et sécurisés pour l'utilisateur.

11. User Story : Générer une clé de sécurité après le paiement

En tant qu'administrateur, je souhaite générer une clé de sécurité supplémentaire au moment du paiement pour sécuriser le billet.

Référence : EXACBBDAPSDP62A_313892_20240530235430

Utilisateur : Administrateur

Cas d'Usage : Générer une clé de sécurité supplémentaire lors du paiement.

Contexte : Les Jeux Olympiques souhaitent renforcer la sécurité des billets achetés pour éviter les usurpateurs.

Critères de Validation : La clé est générée de manière automatique et fiable.

Prérequis : Avoir finalisé le processus de paiement.

Actions Possibles :

- Générer automatiquement une clé supplémentaire pour sécuriser les billets achetés.

Fonctionnalités Déclenchées : La clé générée est utilisée pour vérifier l'authenticité des billets et des visiteurs lors de l'événement.

12. User Story : Vérifier les billets

En tant qu'organisateur, je souhaite vérifier les billets le jour de l'événement en utilisant la combinaison des clés (heure d'achat + nom de l'utilisateur) pour garantir leur authenticité.

Utilisateur : Organisateur

Cas d'Usage : Vérifier les billets le jour de l'événement en utilisant les clés générées.

Contexte : L'organisateur souhaite assurer l'authenticité des billets présentés.

Critères de Validation : La vérification des billets par QR code doit permettre un contrôle efficace et rapide.

Prérequis : L'utilisateur doit présenter son billet pour être scanné par le contrôleur.

Actions Possibles :

- Vérifier les billets présentés lors de l'événement en utilisant les clés générées.

Fonctionnalités Déclenchées : Les billets sont validés pour permettre l'utilisateur d'accéder aux épreuves.

Les User Stories ci-dessus respectent les besoins du clients et permettent donc de couvrir les fonctionnalités clés nécessaires au bon fonctionnement du site web.

1.2 Quels sont les éléments que vous allez sécuriser ? Comment allez-vous procéder ?

La sécurité constitue une priorité essentielle pour le développement de l'application.

L'application peut être vulnérable : des individus malveillants peuvent compromettre un serveur web, accéder à des données sensibles ou causer d'autres types de préjudices. Ces failles proviennent souvent au niveau des entrées utilisateurs, l'attaquant exploite les erreurs de codage pour récupérer des informations sensibles. Nous veillerons à nous défendre des types d'attaques suivants : attaques par injection de code, XSS, CSRF et Hijacking.

Les éléments à sécuriser sont les suivantes:

1-Authentification et autorisation des comptes :

Élément à sécuriser : Les mécanismes d'authentification pour les comptes administrateurs et utilisateurs.

Pour renforcer la sécurité contre les attaques par injection de commande, il est crucial de mettre en place des mesures de sécurité pour traiter correctement les entrées utilisateur.

Méthode : On utilisera un système d'authentification robuste comme celui intégré dans Flask. On veillera à ce que seulement les utilisateurs authentifiés soient autorisés à accéder aux fonctionnalités administrateur ou utilisateur.

2-Communication avec la base de données :

Élément à sécuriser : Les interactions avec la base de données.

L'attaquant peut exécuter du code SQL malveillant sur le serveur de base de données en exploitant une faille de sécurité au niveau des entrées utilisateur dans le but d'extraire des informations sensibles ou même prendre le contrôle du serveur.

Méthode : On utilise les fonctionnalités intégrées de Flask pour interagir avec la base de données de manière sécurisée. Afin de valider et filtrer les entrées utilisateurs, nous utiliserons les requêtes préparées. Ce sont des marqueurs de paramètres qui sont remplacés par les valeurs appropriées au moment de l'exécution de la requête, de cette manière l'attaquant ne peut pas accéder à la base de données avec ses entrées malveillantes.

3- Authentification de la session et autorisation du paiement:

Élément à Sécuriser : L'authentification et la confidentialité du paiement

On peut être victime d'une attaque Hijacking qui a lieu après que l'utilisateur se connecte à sa session et valide les cookies du site légitime. L'attaquant essaye ensuite de détourner l'attention de l'utilisateur pour l'emmener sur un site malveillant et soutirer ses informations dans un formulaire malveillant. Vu que la méthode POST est authentifiée par le serveur, cela permet à l'attaquant d'avoir accès à la base de données si l'utilisateur fournit ses informations sur un formulaire malveillant.

Référence : EXACBBDAPDSDP62A_313892_20240530235430

Méthode : La méthode pour éviter cette attaque est de générer des jetons CSRF au moment de la connexion (jeton de la session) et au moment de la requête du formulaire (jeton envoyé par le formulaire). L'authenticité de l'utilisateur sera vérifiée par la conformité des 2 jetons.

6-Sécurité des URL et des fichiers téléchargés :

Élément à Sécuriser : Les fichiers envoyés par les utilisateurs.

Les attaques d'inclusion de fichiers sont le fait d'inclure le contenu d'un fichier dans un autre fichier. Les attaquants localisent une variable non filtrée du code pour injecter un fichier local.

Méthode : Pour éviter cela, il est important de valider et filtrer toutes les entrées utilisateur (champs formulaire ou recherche), comme les paramètres d'URL, les formulaires et les cookies, pour s'assurer de minimiser les risques de piratage. Par ailleurs, on pourrait limiter les types de fichiers autorisés et stocker les fichiers en dehors du répertoire web pour éviter les attaques basées sur les fichiers.

6-Gestion des Erreurs :

Élément à Sécuriser : Les messages d'erreur.

Méthode : On personnalisera les messages d'erreur pour ne pas révéler d'informations sensibles tels que les identifiants, les mots de passe ou les coordonnées bancaires. Le log d'erreurs sera stocké côté serveur.

7-Déploiement Sécurisé :

Élément à Sécuriser : Le processus de déploiement.

Méthode : On utilisera des connexions sécurisées (HTTPS) pour le déploiement. On s'assurera que les dépendances et les serveurs sont à jour pour éviter les vulnérabilités connues.

1.3 Évoquez les choix techniques que vous avez choisis concernant votre application et justifiez-les (tout en faisant référence au besoin client)

Les choix techniques pour le développement de l'application sont essentiels pour répondre aux besoins du client. Voici les choix techniques que j'envisagerais, en justifiant chacun d'eux en référence aux besoins spécifiques du client :

1. Front-end : HTML/Javascript/Jinja2

Justification : HyperText Markup Language nous servira de balise de langage pour créer et structurer le contenu de nos pages web afin que l'utilisateur puisse visionner le détail des informations du site. HTML utilise des éléments encadrés par des chevrons <> qui indique au navigateur comment afficher le contenu (texte, image, vidéo et audio). La technologie HTML sera associée à la technologie CSS (Cascading Style Sheets) pour le stylage et la mise en forme des pages, et avec la technologie JavaScript pour la création de pages web interactives et dynamiques. Javascript est un langage de script côté client qui permet d'exécuter les actions de l'utilisateur tels que les clics de souris et les envois de formulaires, par exemple pour la connexion du compte utilisateur, la sélection des offres ou le processus de paiement.

Référence : EXACBBDAPDSDP62A_313892_20240530235430

Le choix de HTML avec le moteur de modèle Jinja2 permet de développer le site beaucoup plus facilement tout étant modulable et flexible, c'est-à-dire de répondre au besoin du client plus facilement et rapidement. Jinja2 est un moteur de template pour Python qui permet de générer des pages web dynamiques en utilisant des modèles HTML tout en s'intégrant avec le framework Flask.

2. Communication Front-end/Back-end : API REST

Justification : Nous utiliserons un style d'architecture logicielle pour les systèmes distribués API REST (Representational State Transfer) pour relier le front end et le back end de notre application.

L'API REST est une interface de programmation qui permet de concevoir, de développer des services web interconnectés et qui permet aux utilisateurs par la suite d'accéder et d'exploiter des données et fonctionnalités identifiées par une URI (Uniform Resource Identifier). En effet, les ressources sont définies par des URI et sont manipulées à l'aide d'opérations standardisées telles que GET, POST, PUT et DELETE. L'API REST se base sur le protocole HTTP pour communiquer avec les différents systèmes distribués. Cela permet aux clients de soumettre des requêtes HTTP aux différentes ressources liées à l'API REST pour créer (saisie de l'adresse email, nom et prénom pour la création de compte utilisateur), obtenir des données (authentification du compte et des billets) ou les modifier (changement d'offres).

3. Back-end : Flask (Python)

Nous utiliserons le framework Flask qui fonctionne avec le langage Python pour créer notre API REST. Flask fournit une structure de base pour développer notre application et ce framework permet d'implémenter notre architecture logicielle API REST en définissant des routes et des fonctions de vue qui structureront les fonctionnalités de notre application. Flask offre plusieurs fonctionnalités telles qu'un serveur de développement, un débogueur intégrés pour l'exécution et le débogage de notre application, un système de routage qui associe des URL à des fonctions Python (vues) pour gérer les requêtes. L'architecture modulaire et enfichable de l'API REST permet de prendre en charge les extensions pour ajouter des caractéristiques et des fonctionnalités supplémentaires tels que l'intégration d'une base de données, l'authentification et les formulaires.

Nous utiliserons Flask-RESTful qui est une extension de Flask pour simplifier la gestion des routes de Flask en utilisant sa classe ressources et en fournissant un décorateur afin d'associer les méthodes HTTP aux ressources. Il intègre la validation des données entrantes pour être conforme aux attentes, la gestion des erreurs spécifique à l'API, et la sérialisation des données pour formater les réponses http (conversion objet Python en JSON).

4. Base de données : PostgreSQL

Justification : Nous utiliserons le Système de Gestion de Bases de Données (SGBD) PostgreSQL afin de manipuler les données et les stocker. Il permet de stocker les données dans des tables, de définir des relations entre les tables, d'exécuter des requêtes SQL pour récupérer les données, gérer les utilisateurs et les autorisations.

PostgreSQL est un choix judicieux en raison de sa fiabilité, de ses performances et de ses fonctionnalités avancées. Il est bien pris en charge par le micro framework Flask et offre des fonctionnalités de sécurité robustes. En outre, il est conforme aux exigences du client.

5. Sécurité : Utilisation des extensions de Flask

Justification : Bien que Flask ne propose pas certaines fonctionnalités de sécurité directement intégrées dans son noyau, c'est un micro framework qui intègre des extensions pour rajouter des fonctionnalités de sécurité. L'extension SQLAlchemy permet d'éviter, via des requêtes préparées, les attaques par injection de commande et par injection SQL.

Référence : EXACBBDAPSDP62A_313892_20240530235430

L'extension Jinja2 permet d'éviter les attaques XSS (Cross-Site Scripting).

Les extensions Flask-SeaSurf (pour la protection CSRF) et Flask-Session permettent d'éviter les attaques Hijacking. Cela répond au besoin du client en matière de sécurité des données.

6. Déploiement : Digital Ocean

Justification : Nous utiliserons le fournisseur de service d'hébergement Cloud « Digital Ocean » pour nous concentrer sur notre code, éviter la gestion de l'infrastructure de développement et diminuer nos coûts. C'est un service de déploiement moderne qui offre la scalabilité lorsque l'augmentation du trafic nécessitera d'augmenter les ressources. Cela assure une gestion efficace des ressources tout en répondant aux besoins du client en matière de déploiement.

Dossier 2 : Développement de la solution

2.1 Développez la solution demandée par le client, pour ce faire vous devrez également :

- Produire le MCD (Modèle conceptuel de données)

Voir fichier Draw.io « MCD » dans le Git.

-Effectuez une gestion de projet

o Fournissez l'outil de gestion de projet que vous avez utilisé suivant la méthode **KANBAN**

Pour la gestion de projet, j'ai utilisé un tableau Kanban pour suivre les tâches et leur progression. J'ai choisi Trello, un outil de gestion de projet en ligne, qui offre une interface intuitive et flexible pour organiser les tâches. Le tableau est visible en suivant le lien suivant :

<https://trello.com/invite/b/yhxQkOvs/ATTId027c7062f549874fb4f003c0084c55aA23B3E20/projet-jeux-olympiques>

Voici une représentation simplifiée du tableau Kanban utilisé :

Tableau Kanban sur Trello :

- Colonne du Tableau :
 1. À faire : Liste des tâches à accomplir.
 2. En cours : Tâches en cours de développement ou de test.
 3. À Réviser : Tâches terminées, en attente de révision.
 4. Terminé : Tâches terminées et révisées avec succès.
- Cartes de Tâches : Chaque tâche est représentée par une carte sur laquelle sont indiqués le titre de la tâche, une description détaillée, les membres de l'équipe responsables, et toute date limite associée.
 - Membres de l'Équipe : Chaque membre de l'équipe est représenté par une icône sur les cartes correspondant aux tâches dont il est responsable.
 - Labels : Des labels sont utilisés pour marquer certaines tâches en fonction de leur nature (ex: "Fonctionnalité", "Sécurité", "Documentation", "Fonctionnalité").

Référence : EXACBBDAPDSDP62A_313892_20240530235430

Cette méthode Kanban offre une visibilité en temps réel sur l'avancement du projet, permettant à l'équipe de rester organisée et réactive aux changements.

-Produire une documentation technique

o Cette documentation devra aborder la sécurité, mais également, des évolutions futures de votre application.

1. Architecture et conception :

L'architecture de l'application "AppJO" est conçue en systèmes distribués, elle utilise l'API REST. Cette architecture sépare le front end du back end. Le front end est composé du HTML, CSS, JavaScript et Jinja2 pour les templates, alors que le back end est développé avec Flask, ses extensions et une base de données relationnelle PostgreSQL. La modularité, la scalabilité et la maintenance de l'application sont ainsi assurées.

2. Installation et configuration :

L'installation et la configuration de l'application "AppJO" suivent les étapes suivantes :

- Installation de Python sur votre système.
- Clonage du dépôt Git de l'application.
- Installation des dépendances Python à l'aide de pip .
- Configuration de la base de données PostgreSQL.
- Exécution du script SQL pour créer les tables nécessaires dans la base de données.
- Lancement de l'application en exécutant python app.py depuis le répertoire racine de l'application.

3. API et services :

L'application "AppJO" expose plusieurs services via son API REST, notamment :

- Service d'inscription et de connexion des utilisateurs.
- Service de gestion des billets (affichage, ajout au panier).
- Service de gestion du panier (affichage, mise à jour, paiement).
- Service d'administration des offres (ajout, suppression, mise à jour).

4. Bases de données et modèle de données :

La base de données PostgreSQL est utilisée pour stocker les données de l'application. Le modèle de données comprend 3 tables :

- Table des utilisateurs pour stocker les informations d'identification et les détails des utilisateurs.
- Table des offres pour stocker les informations sur les billets disponibles.
- Table des commandes pour stocker les détails des commandes passées par les utilisateurs.

5. Fonctionnalité et composants :

Les principales fonctionnalités de l'application "AppJO" comprennent :

- Inscription et connexion des utilisateurs avec gestion des sessions.
- Affichage et réservation de billets pour les Jeux Olympiques.
- Gestion du panier d'achat avec fonctionnalités de mise à jour et de paiement.
- Interface d'administration pour gérer les offres de billets.

Référence : EXACBBDAPDSDP62A_313892_20240530235430

6. Convention de codage et bonnes pratiques :

Le code de l'application "AppJO" suit les conventions de codage Python PEP 8 pour garantir une cohérence et une lisibilité du code. De plus, des bonnes pratiques de développement web sont suivies, telles que la protection contre les attaques par injection de code, XSS, CSRF et Hijacking.

7. Tests et débogages :

L'application "AppJO" est accompagnée de tests unitaires du paiement pour assurer le bon fonctionnement de cette fonctionnalité. Des outils de débogage sont utilisés pour détecter et corriger les erreurs rapidement.

8. Maintenance et évolution :

Pour assurer la maintenance et l'évolution de l'application "AppJO", il est recommandé de :

- Effectuer régulièrement des mises à jour de sécurité pour les dépendances Python.
- Surveiller les journaux d'erreurs pour détecter les problèmes potentiels.
- Collecter les retours des utilisateurs pour identifier les améliorations à apporter.
- Planifier des mises à jour fonctionnelles en fonction des besoins des utilisateurs et des évolutions technologiques.

Cette documentation technique fournit un aperçu de l'architecture, de la conception, de l'installation, des fonctionnalités et des bonnes pratiques de l'application "AppJO". En plus du fichier « readme », elle doit être utilisée comme guide pour le déploiement, la maintenance et l'évolution de l'application.

-Produire un manuel d'utilisation

o Conception d'une documentation permettant a l'utilisateur d'utiliser votre application, on peut voir cela comme un « tuto »

Ce manuel d'utilisation, considéré comme un tuto, fournit une introduction rapide aux fonctionnalités principales de l'application.

Manuel d'Utilisation (Tutoriel)

Accéder à l'Application :

1. Rendez-vous sur le site web.
2. Visitez les différents évènements disponibles.

Sélectionner une offre :

1. Cliquez sur « Billet » pour choisir l'offre désirée.
2. Cliquez sur "ajouter"
3. Cliquez sur « Panier » pour visualiser la commande désirée.
4. Connectez-vous ou créer un compte en indiquant votre nom, prénom et email.

Procéder au paiement :

1. Validez votre commande.
2. Renseignez vos coordonnées bancaires.
3. Suivez le processus de paiement.
4. Visualiser le récapitulatif de la commande de votre billet

Référence : EXACBBDAPDSDP62A_313892_20240530235430

BACK-END :

Le client vous impose :

- o D'effectuer un back-end
- o D'effectuer une application dynamique avec du JavaScript (le JavaScript permet de contacter votre back-end sans rechargement de page)
- o Mettre en place des tests et produire un rapport détaillant le pourcentage de code total couvert par les tests
- o Déploiement en ligne

Voir le répertoire « AppJoDepot » sur Git.

<https://github.com/SARHIRI83/AppJoDepot.git>

Référence : EXACBBDAPDSDP62A_313892_20240530235430

README

Installations et configurations :

Cloner le dépôt Github sur votre PC avec le lien suivant :

« <https://github.com/SARHIRI83/AppJoDepot.git> »

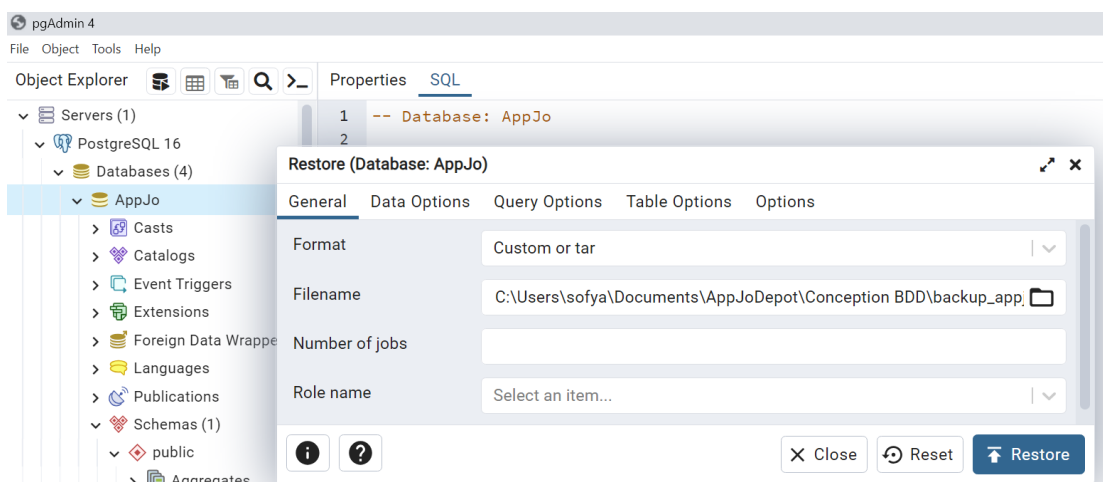
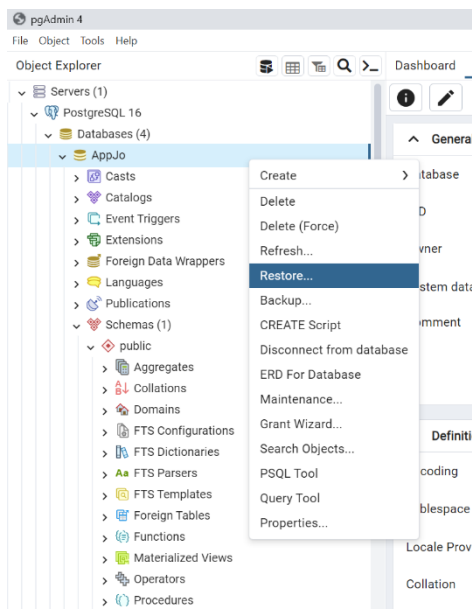
Un répertoire se nommant « AppJoDepot » va alors être créé à l'endroit où vous clonez le dépôt.

Installer Python

Installer PgAdmin, l'interface graphique de PostgreSQL.

Dans PgAdmin, créer une base de données nommée « AppJo ».

Importer le fichier sql qui se trouve dans AppJoDepot/Conception BDD/backup_appjo.sql en faisant un clic droit sur la base de données « AppJo » et en faisant un clic gauche sur « Restore » .



Référence : EXACBBDAPDSDP62A_313892_20240530235430

Une fenêtre s'ouvre, dans le champ « Filename » choisir le fichier «backup_appjo.sql » puis cliquez sur « Restore ».

Ouvrir un terminal ou un IDE.

Installer les modules suivants en ouvrant un terminal en mode administrateur et en entrant la commande pip install pour les modules suivants :

- psycopg2, psycopg2.extras
- flask
- session
- loggingManager, userMixin
- os
- hashlib
- random
- logging
- string
- json
- datetime

```
app.py > ...
1  import datetime
2  from flask import Flask, current_app, render_template, request, jsonify, make_response, redirect, url_for
3  from flask_session import Session
4  from flask_login import LoginManager, UserMixin, login_user, logout_user, login_required, current_user
5  import psycopg2
6  import psycopg2.extras
7  import os
8  import hashlib
9  import random
10 import string
11 import json
12 import logging
```

Ouvrir le projet dans un IDE tel que VSCode

Dans le fichier App.py :

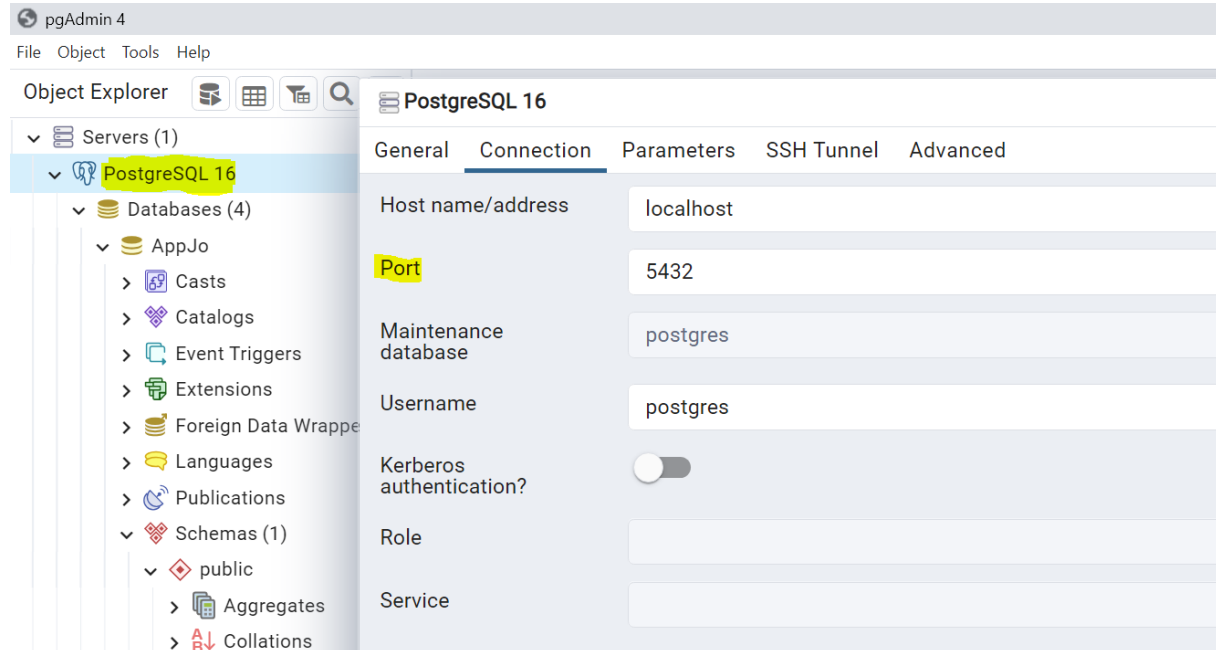
Par défaut, le port d'écoute du serveur PostgreSQL est 5432. Vérifier donc que la valeur du port d'écoute de votre serveur soit identique à la valeur du port d'écoute spécifier dans le fichier app.py. Si le port par défaut (5432) est utilisé, cette ligne peut ne pas être explicite.

Si vous n'utilisez pas l'utilisateur postgres par défaut pour vous connecter à la base de données, changez l'utilisateur dans le champ « user » et mettre le mot de passe associé à l'utilisateur que vous avez choisi dans le champ « password ».

Dans le champ « dbname » il faut qu'il y ait le nom de la base de donnée que vous avez crée plus tôt. Si vous l'avez nommé « AppJo » comme il était indiqué, alors ne changez rien.

Référence : EXACBBDAPDSDP62A_313892_20240530235430

```
# Connexion à la base de données
def get_db_connection():
    conn = psycopg2.connect(
        host='localhost',
        dbname='AppJo',
        user='postgres',
        password='sofyane',
        port=5432 #modifiez le port pour qu'il soit le même que le port d'écoute de votre serveur postgre sql local
    )
    return conn
```



pgAdmin 4

File Object Tools Help

Object Explorer PostgreSQL 16

Servers (1)

PostgreSQL 16

Databases (4)

AppJo

Casts

Catalogs

Event Triggers

Extensions

Foreign Data Wrappers

Languages

Publications

Schemas (1)

public

Aggregates

Collations

General Connection Parameters SSH Tunnel Advanced

Host name/address localhost

Port 5432

Maintenance database postgres

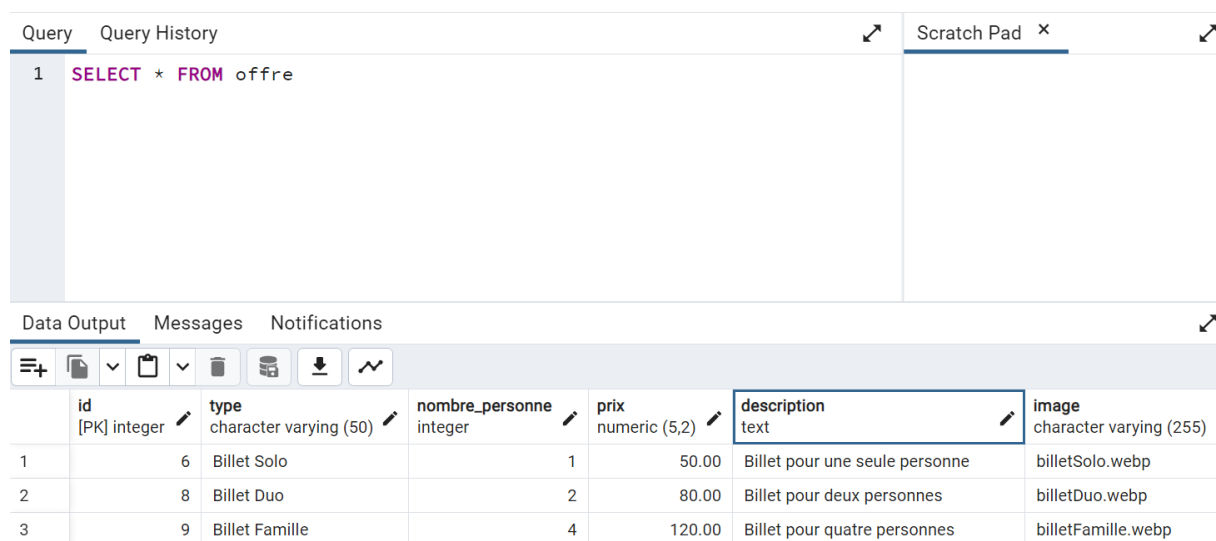
Username postgres

Kerberos authentication? ☐

Role

Service

En vous aidant du modèle conceptuel de données, intégrer la table « offres » dans la base de données :



Query Query History Scratch Pad

1 SELECT * FROM offres

Data Output Messages Notifications

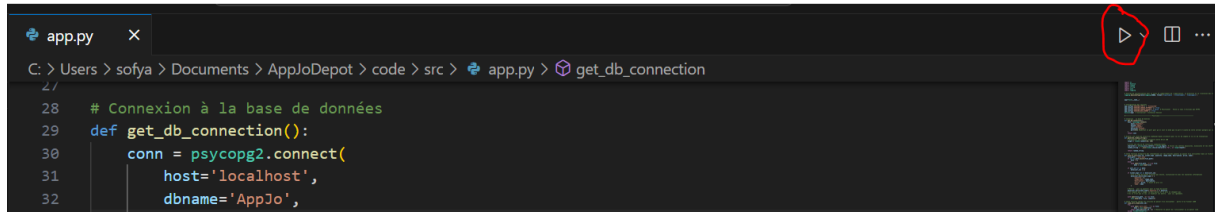
	id [PK] integer	type character varying (50)	nombre_personne integer	prix numeric (5,2)	description text	image character varying (255)
1	6	Billet Solo	1	50.00	Billet pour une seule personne	billetSolo.webp
2	8	Billet Duo	2	80.00	Billet pour deux personnes	billetDuo.webp
3	9	Billet Famille	4	120.00	Billet pour quatre personnes	billetFamille.webp

Faire de même pour la table « utilisateurs » et « commande ».

Référence : EXACBBDAPDSDP62A_313892_20240530235430

Lancement de l'application app.py :

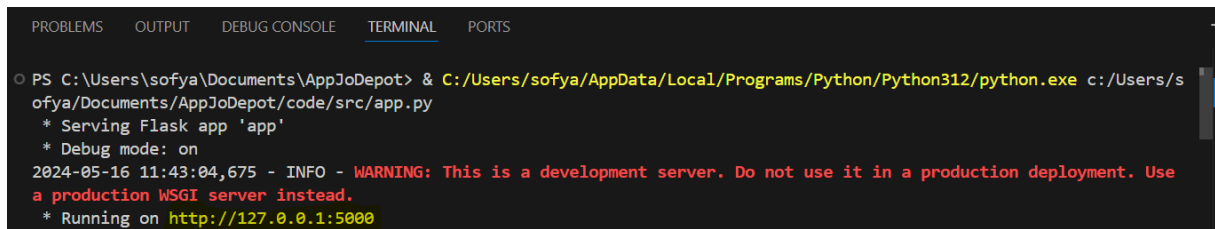
Ouvrez dans VSCode le fichier « app.py » puis cliquer sur le bouton en haut à droite pour exécuter le programme qui lancera le serveur Flask.



```
28 # Connexion à la base de données
29 def get_db_connection():
30     conn = psycopg2.connect(
31         host='localhost',
32         dbname='AppJo',
```

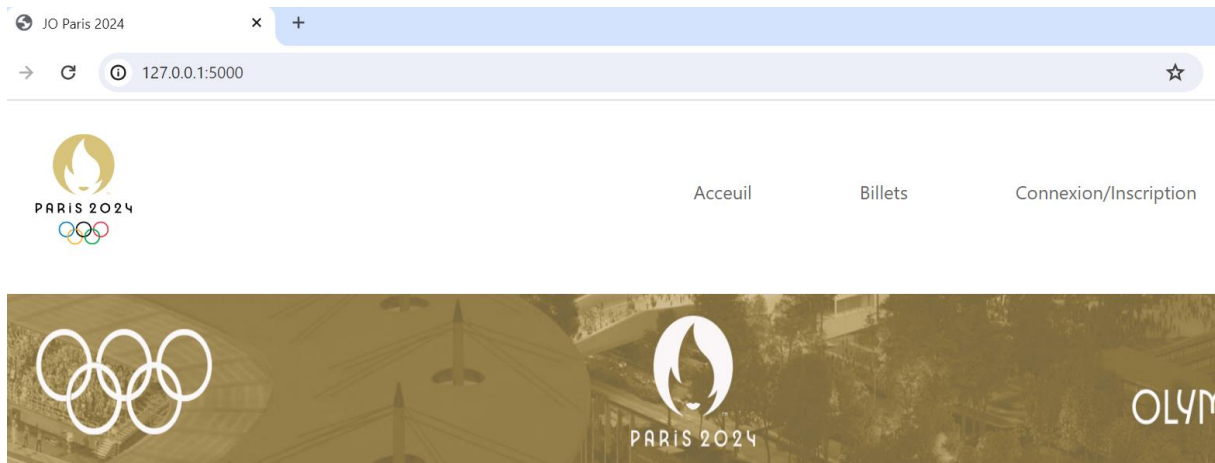
Sinon dans un terminal, accéder au répertoire « AppJoDepot/code/src/ » et taper la commande « python app.py »

Dans le terminal des logs, il apparaît l'URL <http://127.0.0.1:5000/> qui sera entré dans le navigateur.



```
PS C:\Users\sofya\Documents\AppJoDepot> & C:/Users/sofya/AppData/Local/Programs/Python/Python312/python.exe c:/Users/s
ofya/Documents/AppJoDepot/code/src/app.py
* Serving Flask app 'app'
* Debug mode: on
2024-05-16 11:43:04,675 - INFO - WARNING: This is a development server. Do not use it in a production deployment. Use
a production WSGI server instead.
* Running on http://127.0.0.1:5000
```

Entrer cet URL sur votre navigateur pour voir afficher la page d'accueil du site :



Épreuves à la Une




Référence : EXACBBDAPDSDP62A_313892_20240530235430



Parcours client :


Vous pouvez désormais naviguer sur le site pour sélectionner les offres désirées dans l'onglet « Billets », pour vous connecter avec votre email dans l'onglet « Connexion/Inscription, et pour acheter vos billets dans l'onglet « Panier ».

Cliquer sur « Billet » puis dans la page « Billet » cliquer « ajouter au panier » sur l'offre de votre choix, un message vous indique que l'article a été ajouté avec succès, confirmez le message en cliquant OK :



[Accueil](#) [Billets](#) [Connexion/Inscription](#) [Panier](#)

JEUX OLYMPIQUES






Billet Solo
Billet pour une seule personne

Prix : €50.00


[Ajouter au panier](#)

Ensuite, sur cette même page, cliquer sur « Panier » pour que le site vous dirige sur la page « Panier » :



JEUX OLYMPIQUES

Votre Panier



Billet Solo
Billet pour une seule personne
Prix unitaire: €50.00
Quantité: 1

Sous-total: €50.00

[Payer](#)

Référence : EXACBBDAPDSDP62A_313892_20240530235430

Cliquer sur « Payer » pour rentrer dans le processus de paiement. Une page vous demandera de vous connecter pour payer le billet.

Pour le paiement, il faut d'abord que vous créez votre compte puis que vous vous connectez.



Connexion

Email

Mot de passe

[Se connecter](#)

[Pas encore membre ?](#)

Après connexion, suivez le parcours client jusqu' à la page suivante puis rentrez vos coordonnées. Nous utiliserons des fonctions mockées pour simuler le paiement.



Paiement

Nom sur la carte:

Numéro de carte:

Date d'expiration (MM/AA):

CVV:


[Payer](#)



Renseignez les informations de paiement et cliquer sur « payer » pour valider le paiement.

Référence : EXACBBDAPDSDP62A_313892_20240530235430


Compte administrateur :

Lorsque vous vous connectez en administrateur, vous devez saisir l'URL <http://127.0.0.1:5000/offers> pour gérer les offres. Vous remarquerez l'ajout d'onglets comme « ajouter une offre », « modifier » et « supprimer ».

AccueilBilletsConnexion/InscriptionPanier

JEUX OLYMPIQUES

[Ajouter une offre](#)



Prix : €50.00

1

[Ajouter au panier](#)

[Modifier](#)

[Supprimer](#)

Billet Solo
Billet pour une seule personne

Bouton « modifier » :

Lorsque vous cliquez sur « modifier », vous remarquerez en bas de page des champs pour modifier les paramètres de l'offre. Par exemple, pour l'offre solo, si vous modifier le prix à 60€ et que vous rajoutez « super ! » dans les commentaires comme ci-dessous.



Prix : €120.00

1

[Ajouter au panier](#)

[Modifier](#)

[Supprimer](#)

Billet Famille
Billet pour quatre personnes

Modifier Offre

Billet Solo	1	60.00	Billet pour une seule personne, Super !	billetSolo.webp	Modifier
-------------	---	-------	--	-----------------	--------------------------

Référence : EXACBBDAPDSDP62A_313892_20240530235430

Le billet solo se met à jour dès que vous cliquez sur le bouton « modifier » comme ci-dessous :



Billet Solo
Billet pour une seule personne, **Super !**

Prix : **€60,00**
1
Ajouter au panier
Modifier
Supprimer

Bouton « Ajouter » :

Lorsque vous cliquez sur le bouton « ajouter », en bas de page vous trouverez des champs pour ajouter votre offre.



Billet Famille
Billet pour quatre personnes

Prix : **€100,00**
1
Ajouter au panier
Modifier
Supprimer

Nouvelle Offre

Trio
3
100

Billet pour 3 personnes

Image URL

Ajouter

Contactez-nous
+33 1 23 45 67 89

Liens rapides
À propos des Jeux

Suivez-nous
Restez connectés pour les mises à jour et les nouvelles

En prenant l'exemple ci-dessus, vous ajouterez l'offre trio, la page affichera donc l'offre en respectant vos choix comme ci-dessous :



Trio
Billet pour 3 personnes

Prix : **€100,00**
1
Ajouter au panier
Modifier
Supprimer

Contactez-nous


Liens rapides

Suivez-nous

Référence : EXACBBDAPSDP62A_313892_20240530235430

Bouton « supprimer » :

Pour supprimer une offre, il suffit de reprendre l'offre trio que vous avez créer à l'instant et de cliquer sur « supprimer ».



127.0.0.1:5000 indique


Voulez-vous vraiment supprimer cette offre ?

OK

Annuler

Modifier

Supprimer

Trio

Trio

Billet pour 3 personnes

Prix : €100.00

1

Ajouter au panier

Modifier

Supprimer

L'offre trio disparaîtra après confirmation sur le bouton OK.