Smartfuture
Certifications | Training | Technology solutions

Formation & Certification • ITS OD-305

IT Specialist – Software Development

Formation préparant à la certification IT Specialist Software Development, couvrant la programmation, l'orienté objet, les applications web et les bases de données pour former les futurs développeurs.

Durée	Examen
25,5 h	ITS OD-305
Modalité	**Niveau**
Distanciel	Fondamental

Distributeur officiel Certiport

Centre d'examen Certiport

Learn • Practice • Certify

INSCRIPTION / RÉSERVATION

Je m'inscris maintenant

- **Learn** : parcours structuré couvrant l'ensemble des objectifs officiels de l'examen ITS Software Development.
- **Practice** : accès à un simulateur d'examen interactif et à des exercices pratiques pour s'entraîner en conditions réelles.
- **Certify** : voucher d'examen ITS OD-305 Software Development inclus, à passer dans notre centre Certiport.

OBJECTIFS PÉDAGOGIQUES

- Comprendre la structure des programmes, des données et de la mémoire.
- Analyser des problèmes, concevoir des algorithmes et les coder.
- Appliquer la programmation orientée objet avec classes, héritage et polymorphisme.
- Découvrir les bases des applications web et des bases de données tout en se préparant à l'examen ITS OD-305.

PUBLIC CIBLE

- Pour élèves, étudiants et débutants souhaitant démarrer en développement logiciel.
- Pour personnes en reconversion vers les métiers du développement et du génie logiciel.
- Pour profils techniques juniors voulant renforcer leurs bases en programmation.

PRÉREQUIS

- Niveau collège/lycée requis en lecture et mathématiques.
- Premiers pas recommandés en HTML, CSS, JavaScript et un langage orienté objet.
- Des notions de bases de données et de SQL sont un plus.

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Concepts fondamentaux de programmation

Stockage informatique et types de données

- Comprendre comment les programmes et instructions sont stockés en mémoire.
- Différencier piles et tas mémoire et leurs usages pour les variables et objets.
- Identifier les besoins en mémoire des différents types (entiers, flottants, chaînes de caractères, etc.).
- Faire la différence entre données numériques et textuelles.
- Expliquer le principe du ramasse-miettes (garbage collection) dans les langages modernes.

Algorithmes et organigrammes

- Décrire le rôle des structures de décision dans tout langage de programmation.
- Utiliser les tests simples (if) et multiples (if...else, switch).
- Lire, interpréter et construire des organigrammes (flowcharts) pour représenter un algorithme.
- Élaborer des tables de décision pour clarifier les différents cas possibles.
- Évaluer des expressions logiques et arithmétiques dans les structures de contrôle.
- Utiliser les boucles for, while et do...while pour les traitements répétitifs.
- Comprendre la récursivité et identifier les cas où elle est pertinente.

Gestion des erreurs

- Mettre en place une gestion structurée des exceptions (try-catch-finally).
- Réaliser des tests unitaires simples pour valider des fonctions ou modules.
- Lancer et propager des exceptions adaptées aux situations d'erreur.
- Lire et interpréter la pile d'exécution (stack) lors d'un plantage.
- Adopter une approche de "code défensif" pour anticiper les erreurs.
- Comprendre la portée (scope) des variables dans le cadre du traitement des exceptions.

Programmation fonctionnelle et asynchrone

- Utiliser les événements et délégués pour découpler les composants.
- Comprendre le rôle des promesses pour gérer le traitement asynchrone.
- Faire la différence entre exécution synchrone et asynchrone (AJAX, XHR, appels réseau).
- Appréhender le concept d'immutabilité et son intérêt en programmation.

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Principes de développement logiciel

Cycle de vie du développement logiciel (SDLC)

- Identifier les étapes clés : analyse des besoins, conception, implémentation, tests, déploiement et maintenance.
- Comprendre les grandes notions des approches Agiles.
- Différencier les types de tests : unitaires, d'intégration, système, recette utilisateur, fumée (smoke), performance et charge.

Spécifications d'application

- Lire et interpréter des spécifications fonctionnelles et techniques.
- Traduire des besoins métier en prototypes, maquettes ou code.
- Choisir le type d'application et les composants techniques adaptés.

Algorithmes et structures de données

- Utiliser des tableaux, piles, files, listes chaînées et dictionnaires (paires clé–valeur).
- Connaître les principaux algorithmes de tri : sélection, bulle, rapide (quick sort), fusion (merge sort).
- Connaître les algorithmes de recherche : linéaire et dichotomique (binary search).
- Évaluer les impacts de performance liés au choix d'une structure de données.
- Différencier les principes FIFO (file) et LIFO (pile).

Gestion de versions

- Comprendre le rôle d'un système de contrôle de version (ex. Git, GitHub).
- Effectuer les opérations courantes : check-in, check-out, commit, push/pull.
- Créer et gérer des branches, fusionner (merge) et revenir en arrière (rollback).
- Cloner un dépôt et résoudre les conflits de fusion.
- Mettre en œuvre des revues de code (code review) pour améliorer la qualité.

Concepts de codage sécurisé

- Différencier chiffrement, hachage et signatures numériques.
- Comprendre la notion de clés publiques, privées et partagées.
- Identifier les risques liés aux attaques (CSRF, injection SQL, etc.).
- Reconnaître les risques liés à l'utilisation d'iframes et d'éléments externes.

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Programmation orientée objet (POO)

Classes et objets

- Créer et utiliser des classes avec propriétés, méthodes, événements, champs et constructeurs.
- Comprendre la différence entre membres statiques et d'instance.
- Maîtriser l'encapsulation pour protéger les données.
- Composer des objets complexes à partir d'objets plus simples (composition).
- Utiliser les modificateurs d'accès pour contrôler la visibilité (public, privé, etc.).

Héritage

- Étendre les fonctionnalités d'une classe de base dans une classe dérivée.
- Utiliser les classes génériques pour factoriser du code réutilisable.
- Comprendre l'intérêt des classes abstraites pour définir des contrats partiels.

Polymorphisme

- Redéfinir (override) des méthodes d'une classe de base dans une classe dérivée.
- Utiliser les interfaces pour définir des comportements à implémenter.
- Mettre en œuvre la surcharge (overload) de méthodes pour gérer plusieurs signatures.

Applications web

Fondamentaux des applications web

- Utiliser HTML5, CSS3 et JavaScript (ES6) pour construire des interfaces modernes.
- Exploiter les outils de développement des navigateurs pour analyser le DOM, le réseau et les performances.
- Comprendre le fonctionnement des requêtes et réponses HTTP.
- Gérer l'état côté client : cookies, stockage local (localStorage) et de session (sessionStorage).
- Comprendre le cycle de vie d'une page web et le modèle d'événements.
- Différencier programmation côté client et côté serveur.

Hébergement web

- Créer des répertoires virtuels et des sites sur un serveur web.
- Publier et déployer une application web.
- Comprendre le rôle du serveur web dans la chaîne de traitement.

Services web

- Consommer des services web depuis une application cliente.
- Utiliser JSON et XML pour échanger des données.
- Comprendre les API REST, les appels HTTP et l'authentification oAuth.

Modèles d'architecture

- Identifier les modèles MVC (Model-View-Controller) et MVVM (Model-View-ViewModel).
- Comprendre le principe des applications monopage (SPA – Single Page Application).

Bases de données

Conception et normalisation

- Comparer les principales familles de systèmes de gestion de base de données.
- Concevoir un schéma relationnel cohérent à partir des besoins métier.
- Construire et lire des diagrammes entité–relation (ERD).
- Appliquer les principes de normalisation jusqu'à la troisième forme normale (3NF).
- Utiliser index, contraintes, clés primaires et clés étrangères.
- Modéliser les relations et la cardinalité entre tables.

Requêtes ANSI SQL

- Écrire des requêtes de sélection et de mise à jour (SELECT, INSERT, UPDATE, DELETE).
- Différencier les commandes de manipulation (DML) et de définition de données (DDL).
- Créer et utiliser des procédures stockées simples.
- Mettre en œuvre des fonctions, déclencheurs (triggers) et curseurs.
- Réaliser des jointures et optimiser les requêtes avec des index.

PROGRAMME DE LA FORMATION – DÉTAILLÉ

Transactions et concurrence

- Utiliser les commandes de validation (commit) et d'annulation (rollback).
- Comprendre les points de sauvegarde (savepoint) et la gestion de la concurrence.
- Identifier les niveaux d'isolation et les verrous (locks) de base.

Accès aux données

- Comprendre les approches d'accès aux données type Entity Framework (Code First, Database First).
- Expliquer la notion de pool de connexions pour optimiser les accès.
- Utiliser LINQ pour interroger des collections d'objets ou des bases de données.

Types de bases NoSQL

- Différencier bases de documents et bases clé–valeur.
- Identifier les cas d'usage typiques des bases NoSQL.