RÉFÉRENTIEL DE DIPLÔME FEDE

FEDEration for European Education
FÉDÉration Européenne des Ecoles

Bachelor européen

Développement d'applications et solutions connectées

www.fede.education
version 04/2026

OING dotée du statut participatif auprès du Conseil de l'Europe - OING dotée du statut consultatif auprès de la Francophonie
OING dotée du statut de partenaire officiel de l'UNESCO et du CESNU
Registre de transparence de l'Union européenne - 313869925841-90
FEDE - La Voie Creuse 16 - 1202 - Genève - SUISSE - RC Genève : CHE-109.997.364

FEDEration for European Education
FÉDÉration Européenne des Ecoles

Fédération Européenne Des Écoles

Federation for European Education

La FEDE est une Organisation Internationale Non Gouvernementale (OING), institution supranationale, créée à Barcelone en 1963. Elle est dotée du statut participatif auprès du Conseil de l'Europe, du statut consultatif auprès de l'Organisation internationale de la Francophonie (OIF), de l'UNESCO et du Conseil économique et social des Nations unies (CESNU). La FEDE est également membre de la Fédération européenne des employeurs de l'éducation (EFEE), de l'Association internationale des Universités (AIU) et de l'Association européenne des établissements de l'enseignement supérieur (EURASHE).

Elle fédère un réseau international de près de 500 établissements d'enseignement supérieur et professionnel, dans 40 pays et sur 4 continents, qui partagent un projet commun d'excellence académique, d'innovation pédagogique, de recherche scientifique et d'ouverture au monde. La FEDE propose plus de 170 référentiels accessibles en français et en anglais, et pour certains en plusieurs langues européennes (espagnol, allemand, italien, roumain, etc.), préparant les apprenants du Foundation Degree, Bachelor européen, Mastère européen, MBA européen, au DBA.

La FEDE rassemble un réseau international de plus de 300 000 personnes.

SOMMAIRE

Développement d'applications et solutions connectées

LEXIQUE

UC : Unité Capitalisable

UE : Unité d'Enseignement

ECTS : Le terme ECTS signifie *European Credits Transfer System* en anglais, soit système européen de transfert et d'accumulation de crédits

CECRL : Cadre Européen Commun de Référence pour les Langues

LV : Langue Vivante

PRESENTATION

Contexte

Dans un monde en pleine transformation numérique, les entreprises recherchent des développeurs capables de concevoir, sécuriser et déployer des solutions logicielles adaptées aux nouveaux défis technologiques. L'essor de l'intelligence artificielle générative, des microservices, du cloud serverless et des approches Green IT impacte fortement le développement des applications modernes.

Le **Bachelor européen en Développement d'applications et solutions connectées** forme des professionnels polyvalents, maîtrisant à la fois les technologies front-end et back-end, la gestion des infrastructures cloud, la cybersécurité et l'Internet des Objets (IoT). Il répond aux attentes des entreprises en formant des développeurs autonomes, capables de travailler en mode agile et DevOps tout en intégrant les principes d'éco-conception logicielle. Cette formation couvre les langages et outils les plus demandés :

- Les langages de programmation : JavaScript, Python, TypeScript, Kotlin, Swift
- Les Frameworks et technologies : React, Angular, Vue.js, Node.js, Laravel
- Les outils DevOps : Docker, Kubernetes, CI/CD, GitHub Actions
- Les environnements cloud : AWS, Microsoft Azure, Google Cloud

Grâce à une approche pratique et immersive, les étudiants seront formés aux bonnes pratiques de développement, aux architectures modernes (microservices, serverless) et aux normes de sécurité et conformité RGPD.

Ce programme prépare aux métiers en forte demande tels que développeur full stack, ingénieur DevOps, développeur IoT, architecte logiciel ou administrateur de bases de données etc. et s'aligne également sur les certifications professionnelles les plus recherchées : AWS Certified Developer, Microsoft Azure Developer et Google Associate Android Developer.

Avec une pédagogie axée sur des projets concrets et des cas réels d'entreprise, les futurs diplômés seront prêts à relever les défis du développement d'applications dans un environnement technologique en constante évolution.

Objectifs et compétences

- Concevoir et développer des applications web et mobiles en utilisant les technologies front-end (React, Angular, Vue.js) et back-end (Node.js, Express, PHP, Laravel)
- Déployer et sécuriser des infrastructures cloud (AWS, Azure, GCP) en optimisant les performances et la scalabilité des applications
- Maîtriser la gestion des bases de données relationnelles et NoSQL (MySQL, PostgreSQL, MongoDB) en appliquant les principes de normalisation et d'optimisation des requêtes
- Développer des API RESTful et GraphQL sécurisées, en intégrant les bonnes pratiques de gestion des accès et des authentifications (OAuth, JWT, API Gateway)
- Intégrer et programmer des objets connectés (IoT) en assurant la communication entre hardware et software via des protocoles sécurisés (MQTT, HTTP, WebSockets)
- Appliquer les méthodologies Agiles et DevOps (Scrum, Kanban, CI/CD) pour garantir une gestion de projet efficace et une intégration continue
- Implémenter des architectures modernes (microservices, serverless) pour améliorer la modularité et la résilience des solutions logicielles
- Mettre en œuvre des tests automatisés (unitaires, d'intégration et end-to-end) pour garantir la qualité et la fiabilité des applications

Développement d'applications et solutions connectées

- Sécuriser les applications web et mobiles en appliquant les normes OWASP, chiffrement SSL/TLS, gestion des identités (OAuth, JWT)
- Gérer des projets techniques en équipe en utilisant des outils collaboratifs (Jira, Trello, GitLab, Slack) et en rédigeant des documentations techniques complètes

Perspectives d'emploi

Détenir un Bachelor européen de la FEDE, c'est bénéficier de nouvelles opportunités et d'un réseau professionnel international.
Le Bachelor européen développement d'applications et solutions connectées prépare les futurs professionnels des écoles FEDE aux fonctions de :

Liste des métiers accessibles après l'obtention du Bachelor (0-2 ans d'expérience)

✓ Développeur full stack (web et mobile)
✓ Développeur back-end / front-end
✓ Développeur d'applications IoT
✓ Intégrateur d'applications
✓ Administrateur de bases de données
✓ Ingénieur DevOps junior

Liste des métiers accessibles avec 3 à 5 ans d'expérience après la diplomation

✓ Architecte logiciel
✓ Lead développeur / Tech Lead
✓ Consultant en développement d'applications
✓ Développeur cloud/serverless
✓ Développeur d'API et microservices
✓ Développeur en éco-conception logicielle (Green IT, low-code/no-code)
✓ Développeur en intelligence artificielle (IA, machine learning appliqué au développement d'applications)

Grâce à sa pluridisciplinarité et son alignement avec les technologies les plus demandées, le Bachelor européen développement d'applications et solutions connectées permet aux futurs diplômés de postuler dans des entreprises technologiques et startups en forte croissance, en Europe et dans le monde, telles que :

❖ France : Dassault, Ubisoft, Orange Digital etc.
❖ Allemagne : SAP, Siemens, Zalando etc.
❖ Pays-Bas : Booking.com, Adyen, Mollie etc.
❖ Estonie : Hub européen du numérique (ex : Tallinn est un pôle d'innovation)

Développement d'applications et solutions connectées

Correspondances avec les certifications professionnelles

Ce référentiel vise à préparer les étudiants aux compétences exigées par les métiers du développement d'applications et solutions connectées. Afin de renforcer leur employabilité, les modules d'enseignement abordés ont été alignés avec des certifications professionnelles reconnues.
Le tableau, ci-dessous, présente les correspondances entre les modules du référentiel et les certifications professionnelles qu'un apprenant pourra envisager de valider en complément de sa formation :

Module	Certifications associées	Compétences professionnelles couvertes
Développement Frontend (HTML, CSS, JavaScript, Frameworks modernes)	*Meta Front-End Developer Certificate* *Google Mobile Web Specialist*	Développement d'interfaces web interactives, accessibilité, performance web
Développement Backend (Node.js, Express, PHP, SQL/NoSQL)	*AWS Certified Developer – Associate* *Microsoft Certified : Azure Developer Associate*	Création d'API REST, gestion des bases de données, sécurité backend
Développement d'applications mobiles (natives & hybrides)	*Google Associate Android Developer* *Apple Certified iOS App Developer*	Développement d'applications mobiles sur Android et iOS, gestion des performances mobiles
Architecture logicielle et méthodologies Agile	*Scrum Master Certified (SMC)* *Professional Scrum Developer (PSD)*	Conception d'architectures robustes, gestion des projets agiles (Scrum, Kanban)
DevOps et déploiement Cloud (CI/CD, Docker, Kubernetes, AWS, Azure, GCP)	*AWS Certified DevOps Engineer – Professional* *Docker Certified Associate (DCA)* *Microsoft Azure DevOps Engineer Expert*	Automatisation du déploiement, conteneurisation, CI/CD, orchestration
Sécurité des applications et conformité RGPD	*Certified Secure Software Lifecycle Professional (CSSLP)* *OWASP Software Security Practitioner*	Sécurisation des applications web et mobiles, protection des données utilisateurs, conformité réglementai
Développement IoT et systèmes embarqués	*Cisco IoT Fundamentals Certification* *AWS Certified IoT Specialty*	Intégration des objets connectés, protocoles IoT (MQTT, HTTP, WebSockets), cybersécurité IoT
Tests et assurance qualité logicielle	*ISTQB Certified Tester – Foundation Level* *Certified Agile Tester (CAT)*	Tests unitaires, tests d'intégration, automatisation des tests
Green IT et éco-conception logicielle	*Certified Green Software Engineer (CGSE)*	Optimisation des performances énergétiques des applications, développement durable
Gestion de projet informatique (Agile, DevOps, Jira, Trello, GitLab)	*PMI Agile Certified Practitioner (PMI-ACP)* *Professional Scrum Master (PSM I)*	Gestion d'équipes techniques, outils de gestion de projet et collaboration

Développement d'applications et solutions connectées

Prérequis d'entrée en formation

Le candidat doit être titulaire d'un diplôme ayant validé l'acquisition de 120 ECTS ou être titulaire d'une certification professionnelle de niveau 5 du CEC (Cadre Européen des Certifications) dans la spécialité.

Règlements des diplômes

- Règlement général des diplômes FEDE
- Règlement du Bachelor européen de la FEDE

Candidat en situation de handicap

Pour garantir l'égalité des chances entre les candidats, les modalités d'évaluation sont, dans la mesure du possible, inclusives. Des aménagements aux conditions de passation des épreuves orales, écrites et pratiques des examens de CDE FEDE FRANCE, rendus nécessaires en raison d'un handicap ou d'un trouble de la santé invalidant, sont prévus.

En outre, si ces aménagements n'étaient pas suffisants, tout candidat peut saisir le référent handicap du certificateur pour aménager, dans le respect du règlement des examens et des spécifications du référentiel, les modalités d'évaluation.

Formation et Intelligence artificielle (IA)

Consciente des enjeux et des transformations profondes induites par l'IA sur les méthodes pédagogiques, les compétences professionnelles et les métiers de demain, la FEDE encourage ses établissements membres à sensibiliser ses formateurs et apprenants à un usage raisonné et réfléchi de l'IA dans leurs pratiques éducatives et professionnelles.

La FEDE préconise que l'intégration de l'IA dans les parcours de formation s'inscrive dans une approche éthique et soit alignée avec les principes des grandes institutions internationales citées ci-dessous. La FEDE recommande notamment de s'appuyer sur les publications et cadres de référence suivants :

- UNESCO, *Référentiel de compétences en IA pour les apprenants*, 2025
- UNESCO, *Référentiel de compétences en IA pour les enseignants*, 2025
- UNESCO, *Orientations pour l'intelligence artificielle générative dans l'éducation et la recherche*, 2024
- Commission européenne, *Lignes directrices éthiques sur l'utilisation de l'intelligence artificielle (IA) et des données dans l'enseignement et l'apprentissage à l'intention des éducateurs*, 2022
- Council of Europe, *Artificial intelligence and education - A critical view through the lens of human rights, democracy and the rule of law*, November 2022

Les établissements membres sont invités à intégrer ces principes dans leur approche pédagogique et à favoriser une réflexion critique et constructive sur l'impact de l'IA à chaque étape du parcours d'apprentissage.

Formation aux compétences linguistiques

Depuis sa création, en 1963, la FEDE promeut une éducation qui associe développement des compétences professionnelles et citoyennes. Dans un monde caractérisé par la diversité des environnements de travail, la mobilité des parcours et l'intensification des échanges, la maîtrise des langues vivantes constitue un levier important d'insertion professionnelle et d'employabilité. Elle contribue également au développement de compétences interculturelles et à la capacité des diplômés à

s'adapter à des contextes variés et à collaborer avec des interlocuteurs d'horizons culturels et linguistiques différents, y compris dans leur propre pays.

Conformément aux principes portés par le Conseil de l'Europe en faveur du multilinguisme et du dialogue interculturel, l'enseignement et l'évaluation d'une langue vivante européenne sont obligatoires dans les diplômes FEDE. Ces enseignements s'appuient sur le Cadre européen commun de référence pour les langues (CECR) et visent à développer des compétences de communication mobilisables dans des situations concrètes d'étude et de travail.

La FEDE inscrit par ailleurs ses dispositifs d'évaluation linguistique dans une démarche de qualité et d'amélioration continue en tant que membre associé de l'Association Language Testers in Europe (ALTE).

Formation aux compétences citoyennes

Les modules de Culture et citoyenneté européennes occupent une place centrale dans les diplômes européens FEDE. Ils traduisent la volonté d'associer l'acquisition de compétences professionnelles au développement de compétences citoyennes, afin de préparer les apprenants à exercer leurs responsabilités dans des sociétés démocratiques, interculturelles et confrontées à des défis économiques, sociaux et environnementaux majeurs.

Cette approche s'inscrit dans les travaux menés au niveau européen en matière de compétences citoyennes, auxquels la FEDE contribue notamment dans le cadre de son statut participatif auprès du Conseil de l'Europe, et s'aligne en particulier sur le Cadre de référence des compétences pour une culture de la démocratie. Elle s'inscrit également dans les initiatives internationales liées à l'éducation au développement durable, en lien avec l'UNESCO, auprès de laquelle la FEDE dispose d'un statut consultatif et participe notamment au Green Education Partnership.

Dans ce cadre, certaines unités d'enseignement sont obligatoires selon le niveau du diplôme : les programmes de Foundation Degree et de Bachelor intègrent des modules consacrés à la compréhension du projet européen, des institutions démocratiques et du dialogue interculturel, tandis que les Mastères incluent un enseignement dédié aux enjeux de la transition écologique et à la responsabilité des organisations face aux défis du développement durable.

Ces enseignements contribuent ainsi à former des apprenants capables d'exercer leurs responsabilités professionnelles avec discernement et d'agir comme citoyens dans des sociétés démocratiques, interculturelles et engagées dans la transition écologique.

UNITES CAPITALISABLES ET HORAIRES INDICATIFS

	Liste des unités capitalisables	Contenu	Horaires indicatifs en face à face pédagogique
Épreuves obligatoires	**UE D** UC D31	Développement d'applications et systèmes connectés	500 à 640 h
	UC D32	Mission professionnelle	12 semaines
	UC D33	Projet Intégratif de développement d'une application de gestion d'une communauté avec IoT intégrée	55 h
	UE B UC B31*	Langue vivante européenne 1 *Utilisateur indépendant*	60 à 80 h
	UE A UC A2	Le projet européen : culture et démocratie pour une citoyenneté en action	20 à 25 h
	UC A3	Le management interculturel et les ressources humaines en Europe	20 à 25h
Épreuves facultatives	UC B32*	Langue vivante 2 *Utilisateur indépendant*	
	UC B33*	Langue vivante 3 *Utilisateur indépendant*	

* Le référentiel d'examens est commun pour toutes les langues vivantes européennes.
Les apprenants ont la possibilité de choisir parmi les langues vivantes suivantes :

- Langue vivante 1 : Allemand, Anglais, Espagnol, Français, Italien, Portugais ;
- Langues vivantes 2 et 3 : Allemand, Anglais, Arabe, Chinois, Espagnol, Français, Italien, Portugais.

NB.1 Attention : les langues vivantes choisies par l'apprenant doivent être différentes de celle dans laquelle il passe les épreuves du domaine européen et du domaine professionnel. Exemple : si les épreuves européennes et professionnelles sont passées en français, les langues vivantes choisies ne peuvent pas comprendre le français.

NB.2 Attention : les horaires ci-dessus représentent les heures de face à face pédagogique préconisées (en présentiel ou en distanciel) et doivent être complétées par les heures de travail personnel de l'apprenant.

ARCHITECTURE DU DIPLOME FEDE

Bachelor européen Développement d'applications et solutions connectées				Evaluations	
Épreuves	**U.C.**	**ECTS**	**Coeff.**	**Modalités**	**Durée**
D3 Expertise Professionnelle	**D31**	22	7	Épreuve professionnelle écrite + étude de cas	6h00
	D32	12	4	Soutenance professionnelle	0h30
	D33	8	2	Contrôle continu	
B31 Langue Vivante Européenne 1	**B31.1**	6	2	Ecrit	1 h
	B31.2	6	2	Oral	40min
A2 Le projet européen : culture et démocratie pour une citoyenneté en action	**A2**	3	1	QCM en ligne	40 min
A3 Le management interculturel et les ressources humaines en Europe	**A3**	3	2	QCM en ligne	40 min
Total		**60**	**20**		
Epreuves facultatives — **B32** Langue Vivante 2	**B32**	6		Écrit + Oral	105 min
Epreuves facultatives — **B33** Langue Vivante 3	**B33**	6		Écrit + Oral	105 min

Pour les épreuves facultatives, seuls les points au-dessus de 10/20 sont comptabilisés et comptent double.

UE D | Expertise Professionnelle

UC D31

Développement d'applications et systèmes connectés

A. Formation

Le volume horaire recommandé de formation en face à face pédagogique est de 500 à 640 heures.

L'UC D31 est composé des 13 modules suivants :

1. Gestion de projet Agile et DevOps (30 à 35 h)
2. Architecture logicielle, conception et gestion Agile (40 à 50 h)
3. Développement web moderne ; front-end et PWA (100 à 110 h)
4. Développement Backend et API : Node.js, PHP, SQL/NoSQL (90 à 100 h)
5. Développement avancé d'API et microservices : architectures API-first, GraphQL et serverless (40 à 50 h)
6. Développement mobile natif et hybride : iOS, Android, Flutter (60 à 70 h)
7. Tests logiciels et automatisation QA (25 à 30 h)
8. Sécurisation des applications web et mobiles : OWASP, RGPD, API Security (40 à 50 h)
9. Cloud computing et architectures serverless (40 à 50 h)
10. Développement IoT et communication entre objets connectés (30 à 40 h)
11. Éthique, réglementation et éco-conception logicielle (20 à 25 h)
12. Développement d'applications IA et Machine Learning appliqué (20 à 25 h)
13. Veille technologique et évolutions des pratiques IT (5 à 6 h)

Contenu	Capacités attendues
Gestion de projet Agile et DevOps (30 à 35 h)	
Objectifs pédagogiques globaux : - **Assimiler les principes de base de la gestion de projet appliqués aux environnements de développement logiciel et d'applications connectées** - **Appréhender les méthodologies Agiles (Scrum, Kanban) et leur application concrète dans les projets de développement** - **Appliquer des méthodes de planification, suivi et reporting adaptées aux équipes techniques** - **Utiliser des outils de gestion de projet en fonction des besoins spécifiques des développeurs** - **Mettre en œuvre des stratégies de gestion des risques propres aux projets IoT et solutions connectées**	
1. Introduction à la gestion de projet informatique • Les principes de bases de la gestion de projet technique ▪ Définition et cycle de vie d'un projet ○ Définition des objectifs ○ Planification : création d'un diagramme de Gantt pour la gestion des livrables ○ Exécution : suivi des tâches via des outils comme Jira ou GitLab par exemple ○ Clôture du projet : analyse « post-mortem » ▪ Rôles et responsabilités des membres d'une équipe projet : focus sur le rôle du développeur • Les environnements spécifiques aux développeurs IoT ▪ Les défis de gestion d'un projet IoT ○ L'intégration matérielle et logicielle ○ La scalabilité et la performance	*Analyser les principes de base d'un projet technique, en décrivant son cycle de vie, afin de structurer le développement d'une application connectée* *Planifier les étapes d'un projet, en utilisant des outils comme le diagramme de Gantt, pour gérer efficacement les livrables* *Suivre l'avancée des tâches, en utilisant des plateformes comme Jira ou GitLab, afin de garantir la réalisation des objectifs du projet* *Évaluer les résultats d'un projet, en menant une analyse post-mortem, pour identifier les axes d'amélioration pour les futurs projets*

○ La sécurité des données ▪ La planification des tâches : Jira ou Trello, par exemple, pour l'organisation des étapes spécifiques aux projets IoT (ex : ajout d'une tâche dans un tableau Kanban avec Jira, user stories avec Trello) ▪ La collaboration technique : utilisation de GitLab, par exemple, pour suivre les versions du code et travailler en équipe	*Définir les rôles et responsabilités dans une équipe projet, en mettant l'accent sur le rôle du développeur, afin de renforcer la collaboration* *Identifier les défis spécifiques aux projets IoT, en analysant les contraintes d'intégration matérielle et logicielle, pour anticiper les risques et les gérer efficacement* *Organiser les tâches d'un projet IoT, en utilisant des outils comme Jira ou Trello, afin de structurer les workflows et les user stories* *Collaborer au sein d'une équipe, en utilisant des outils comme GitLab, pour assurer un suivi des versions et une gestion collective du code*
2. Les méthodologies agiles appliquées au développement d'applications • La méthodologie Scrum ▪ Les différents rôles clés (ex : Scrum Master) ▪ Sprint backlog et user stories ▪ Daily Stand-ups : présentation d'un Daily Scrum pour le partage des progrès et des points de blocages dans l'avancée du projet • Kanban et ses fonctions ▪ La gestion des flux de tâches avec Kanban ▪ La visualisation des workflows	*Appliquer les principes de la méthodologie Scrum, en définissant les rôles clés et en organisant des sprints, pour structurer efficacement les projets* *Créer un backlog de projet, en rédigeant et priorisant des user stories, pour organiser le travail de développement.* *Participer aux réunions Daily Scrum, en partageant les avancées et les blocages, pour favoriser une progression collective harmonieuse* *Gérer les flux de tâches et visualiser les workflows, en utilisant un tableau Kanban, afin d'optimiser la répartition du travail dans une équipe technique*
3. La gestion des risques et des imprévus • L'identification des risques ▪ Les divers types de risques dans un projet technique (ex : latence dans les communications entre capteurs et API etc.) ▪ La matrice de risque : le classement des risques selon leur impact et probabilité • La planification de la résilience ▪ La création de plans de contingence (ex : changement de protocole en cas d'échec technique)	*Identifier les risques d'un projet technique, en analysant leur impact et leur probabilité, pour anticiper les obstacles potentiels* *Classer les risques, en utilisant une matrice d'évaluation, afin de prioriser les actions correctives* *Concevoir des plans de contingence, en élaborant des stratégies de résilience, pour gérer les imprévus techniques comme un changement de protocole*
4. Collaboration et communication dans une équipe technique • La communication interdisciplinaire ▪ La rédaction technique de documentations claires (ex : instructions techniques pour l'installation d'un capteur connecté) ▪ Les outils digitaux collaboratifs : Slack ou Teams pour les échanges de travail d'une équipe	*Rédiger des documentations techniques claires, en détaillant les processus et instructions, pour faciliter l'installation et l'utilisation de dispositifs connectés* *Communiquer au sein d'une équipe, en utilisant des outils collaboratifs comme Slack ou Teams, afin de centraliser les échanges et améliorer la coordination*

• La collaboration technique via des outils numériques ▪ Les outils de gestion comme GitLab (ex : mise en place de tickets avec des tags spécifiques pour signaler des « bugs » etc.)	*Gérer les tickets de tâches, en utilisant des outils comme GitLab, pour suivre les bugs, signaler les priorités, et organiser le travail collectif*
Contenu	**Capacités attendues**
Architecture logicielle, conception et gestion Agile (40 à 50 h)	
Objectifs pédagogiques globaux : - **Appréhender et appliquer les principes des méthodologies Agiles, notamment Scrum, pour la gestion de projets logiciels et la collaboration efficace en équipe** - **Concevoir des architectures logicielles, en suivant le modèle MVC et en utilisant des frameworks courants, pour assurer une séparation claire des responsabilités dans une application** - **Mettre en œuvre une architecture basée sur des micro-services, en gérant la communication interservices et en optimisant les déploiements indépendants et l'évolutivité des services** - **Utiliser des outils DevOps, notamment Docker et Kubernetes, pour coordonner des conteneurs, gérer les déploiements, et automatiser les pipelines d'intégration continue (CI) et de déploiement continu (CD)** - **Assurer le suivi, le monitoring et la gestion des performances des services déployés, via des outils de surveillance et d'observabilité afin de garantir la fiabilité des systèmes**	
1. Les méthodologies agiles : Scrum • Introduction à la méthodologie de projet agile ▪ Les valeurs et les principes du manifeste Agile : collaboration, flexibilité, réactivité aux changements etc. ▪ Les méthodologies agiles vs les méthodologies classiques de projets (Waterfall) • Présentation du cadre Scrum ▪ Les différents rôles du cadre Scrum : Product Owner, Scrum Master et les équipes de développement ▪ Les artifacts Scrum : Product backlog, Sprint backlog, Increment ▪ Les cérémonies Scrum : Sprint planning, Daily Scrum, Sprint review, Sprint retrospective • Présentation des workflows Agile intégrant DevOps ▪ Automatisation des tests et validation continue dans les cycles Agile ▪ Introduction aux outils CI/CD (Jenkins, GitHub Actions, GitLab CI/CD) et à leur rôle dans l'intégration continue • La création et la gestion d'un backlog ▪ La rédaction et la priorisation des user stories pour l'évaluation de la vélocité de l'équipe ▪ Introduction à la gestion du backlog : les outils ClickUp, Jira, Trello, Azure DevOps etc. • La gestion de l'équipe et la communication dans un environnement Agile ▪ La collaboration, la gestion des conflits et le maintien de la motivation de l'équipe	*Appréhender les valeurs et principes du manifeste Agile, en les comparant aux méthodologies classiques, afin de promouvoir la collaboration, la flexibilité et la réactivité dans la gestion de projets* *Identifier les rôles clés du cadre Scrum, en détaillant les responsabilités de chacun, pour structurer efficacement une équipe projet dans un environnement agile* *Mettre en œuvre des workflows Agile intégrant DevOps, en automatisant les tests et la validation continue à l'aide d'outils CI/CD tels que Jenkins, GitHub Actions et GitLab CI/CD, afin d'optimiser la gestion et le déploiement des développements logiciels* *Créer un backlog de produit, en rédigeant et en priorisant les user stories, afin d'organiser et évaluer le travail de l'équipe* *Suivre la progression de l'équipe, en utilisant les burn-down charts et burn-up charts, afin de visualiser l'avancement des sprints et ajuster les efforts si nécessaire*

▪ Le suivi de la progression à l'aide des burn-down charts et burn-up charts • Les indicateurs de performance dans Scrum : l'utilisation de la vélocité, le cycle time, le lead time • Les outils de gestion de projets agiles ▪ Jira, Trello et Azure DevOps pour la gestion des projets et le suivi des user stories ▪ Slack, Microsoft Teams, ou Discord pour la communication et la collaboration au sein de l'équipe	*Analyser les performances de l'équipe, en mesurant la vélocité, le cycle time, et le lead time, afin d'optimiser l'efficacité et améliorer le déroulement des sprints dans une équipe Agile* *Utiliser des outils de gestion de projets Agiles, pour gérer les user stories, les sprints, et faciliter la collaboration entre les membres de l'équipe* *Faciliter la communication et la gestion des équipes, en utilisant des outils collaboratifs, pour maintenir la motivation et résoudre efficacement les conflits*
2. L'architecture MVC & les microservices • Introduction à l'architecture MVC (Modèle-Vue-Contrôleur) ▪ La séparation des préoccupations dans une application avec l'architecture MVC ▪ Le rôle de chaque couche : Modèle (gestion des données), Vue (affichage), et Contrôleur (logique métier) ▪ Les frameworks MVC courants (ex : Laravel pour PHP, Spring pour Java, Express.js pour Node.js etc.) • Les microservices ▪ Les architectures monolithiques vs les microservices : avantages et inconvénients ▪ Les concepts clés des microservices ○ La communication par API ○ La gestion des services ○ L'indépendance des déploiements ▪ Les patterns de microservices : Service Discovery, API Gateway, Circuit Breaker • La structuration d'une application en microservices ▪ La décomposition des fonctionnalités en services indépendants ▪ La mise en place de la communication interservices via des API RESTful, gRPC, ou messaging queues (RabbitMQ, Kafka etc.) ▪ La gestion des bases de données décentralisées pour chaque micro-service • La gestion avancée des micro-services : l'implémentation des patterns d'auto-scaling et la gestion des erreurs à l'aide de mécanismes spécifiques (ex : Hystrix) • Les défis et enjeux des microservices ▪ La gestion de la consistance des données dans un système distribué ▪ Le monitoring et la traçabilité des services via les outils Prometheus, Grafana et Jagger • Les outils pour la gestion des microservices	*Explorer l'architecture MVC, en identifiant les rôles du Modèle, de la Vue, et du Contrôleur, afin de structurer une application web selon la séparation des préoccupations* *Utiliser des frameworks MVC courants comme Laravel, Spring, ou Express.js, pour implémenter des applications web modulaires et bien structurées* *Comparer les architectures monolithiques et les micro-services, en évaluant leurs avantages et inconvénients, pour choisir la meilleure approche selon la nature du projet* *Implémenter des patterns de micro-services, en utilisant Service Discovery, API Gateway, et Circuit Breaker, afin d'assurer la résilience et l'efficacité des services dans un environnement distribué* *Structurer une application en micro-services, en décomposant les fonctionnalités en services indépendants, afin d'assurer l'indépendance des déploiements et une meilleure scalabilité* *Mettre en place la communication interservices, en utilisant des API RESTful, gRPC, ou des messaging queues, pour garantir des échanges fiables entre les micro-services* *Automatiser la gestion des micro-services, en mettant en œuvre des mécanismes d'auto-scaling et de gestion des erreurs, afin d'assurer la scalabilité et la solidité des services*

▪ Express.js, Spring Boot, ou Laravel pour implémenter des services basés sur l'architecture MVC ▪ Kong ou NGINX et API Gateway ▪ RabbitMQ ou Kafka pour la communication asynchrone entre services	*Surveiller et tracer les micro-services, en employant des outils dédiés, pour assurer un monitoring efficace et une traçabilité des services*
3. DevOps : Introduction à Docker, Kubernetes et la gestion des CI/CD	
• Introduction à Docker et aux conteneurs ▪ Le concept de conteneurs et leurs différences avec les machines virtuelles ▪ Présentation de Docker : installation, la création de dockerfiles, la gestion des images et des conteneurs ▪ Les différents avantages de Docker : la portabilité, l'isolation des environnements et la scalabilité	*Explorer le concept des conteneurs, en différenciant Docker des machines virtuelles, afin d'exploiter les avantages de la portabilité, de l'isolation des environnements, et de la scalabilité dans les systèmes modernes* *Installer et configurer Docker, en créant des dockerfiles et en gérant des images et des conteneurs, pour automatiser le déploiement des applications dans des environnements conteneurisés*
• Les conteneurs avec Kubernetes ▪ La gestion des conteneurs à grande échelle avec Kubernetes ▪ Les concepts clés de Kubernetes : pods, services, deployment, confingmaps et volumes ▪ La mise en place d'un cluster Kubernetes pour coordonner plusieurs services déployés via Docker	*Mettre en place un cluster Kubernetes, en combinant plusieurs services conteneurisés, afin de gérer les conteneurs à grande échelle et assurer la coordination des micro-services*
• L'automatisation des pipelines CI/CD ▪ Introduction à l'intégration continue (CI) et au déploiement continu (CD) ▪ Les outils de CI/CD : Jenkins, GitLab CI, GitHub Actions pour l'automatisation des tests, des builds et des déploiements ▪ L'utilisation de Docker dans les pipelines CI/CD pour garantir la cohérence des environnements de développement, de test, et de production	*Automatiser le cycle de vie des applications, en mettant en œuvre des pipelines CI/CD, pour automatiser les tests, les builds et les déploiements dans des environnements DevOps* *Intégrer Docker dans les pipelines CI/CD, en garantissant la cohérence entre les environnements de développement, de test et de production, afin de faciliter le déploiement des applications*
• Surveillance et monitoring des environnements DevOps ▪ Les outils de monitoring (Prometheus) et de visualisation (Grafana) pour suivre la performance des services ▪ Introduction à l'observabilité avec ELK Stack (Elasticsearch, Logstash, Kibana) pour l'analyse des logs et l'identification des anomalies dans les systèmes déployés	*Surveiller les services et les systèmes déployés, en utilisant des outils de monitoring, afin de suivre les performances des services et d'identifier les anomalies* *Analyser les logs des systèmes, en utilisant l'ELK Stack, pour améliorer l'observabilité des systèmes et résoudre les incidents en production*
• Les outils spécifiques au Devops ▪ Docker pour la création et la gestion des conteneurs. ▪ Kubernetes pour l'orchestration des conteneurs et la gestion des déploiements à grande échelle	*Optimiser la gestion des déploiements à grande échelle, en utilisant Kubernetes, afin d'adapter les ressources en fonction de la charge*

▪ Jenkins, GitLab CI, ou GitHub Actions pour la mise en place des pipelines CI/CD ▪ Prometheus et Grafana pour le monitoring et la visualisation des performances des services ▪ ELK Stack (Elasticsearch, Logstash, Kibana) pour l'analyse des logs et l'observabilité des systèmes en production	
Contenu	**Capacités attendues**
Développement web moderne : front-end et PWA (100 à 110 h)	
Objectifs pédagogiques globaux : - **Maîtriser la création de pages web en utilisant HTML5 et CSS3, en appliquant les principes de responsive design pour assurer une compatibilité multi-supports** - **Implémenter des fonctionnalités interactives en manipulant le DOM avec JavaScript et en intégrant des requêtes asynchrones pour améliorer l'expérience utilisateur** - **Développer des applications web dynamiques, en utilisant les Frameworks modernes et en respectant les bonnes pratiques de performance et de réactivité** - **Intégrer des techniques avancées de gestion de l'état, de routage et de communication avec des API externes dans des applications web complexes** - **Concevoir un site web responsive, en utilisant respectivement HTML5, CSS3, JavaScript, et les Frameworks modernes**	
1. Initiation à la conception graphique et UI/UX • Les principes de base du design graphique appliqué au développement web et mobile ▪ Les notions essentielles de composition visuelle, typographie, couleurs et contrastes ▪ Les règles du design responsive : mobile-first, grilles et flexibilité des interfaces ▪ L'accessibilité et les normes d'ergonomie web (WCAG, Material Design, Human Interface Guidelines) • Initiation aux outils de création graphique et de prototypage ▪ Figma, Adobe XD, Sketch pour la conception d'interfaces interactives ▪ La création et l'utilisation de composants réutilisables ▪ Animation et transitions d'interfaces avec Lottie, CSS animations, GSAP • La création d'une charte graphique et d'un prototype fonctionnel ▪ Définition des éléments visuels et graphiques d'une application web ou mobile ▪ Intégration des maquettes en HTML, CSS et JavaScript ▪ Utilisation d'animations et interactions pour dynamiser l'expérience utilisateur **2. Les langages HTML5 et CSS3**	*Structurer des interfaces utilisateur, en appliquant les principes fondamentaux du design graphique, afin d'améliorer l'ergonomie et l'esthétique des applications web et mobiles* *Adapter les mises en page, en mettant en œuvre les règles du design responsive, afin d'assurer une compatibilité optimale sur différents écrans et formats* *Concevoir des prototypes interactifs, en utilisant des outils de prototypage avancés, afin de tester et valider les parcours utilisateurs* *Animer les interfaces, en intégrant des transitions et micro-interactions avec Lottie, CSS animations et GSAP, afin d'améliorer l'expérience utilisateur et l'engagement* *Définir une charte graphique cohérente, en harmonisant couleurs, typographies et composants visuels, afin d'assurer une identité visuelle homogène sur l'ensemble du projet*

Développement d'applications et solutions connectées

• Introduction à HTML5 : la structuration des pages, le responsive design, SASS/LESS • Initiation au maquettage de sites internet sur Figma	*Maquetter un site web, en s'appuyant sur Figma, pour concevoir et planifier les interfaces utilisateurs avant l'implémentation en HTML5/CSS3*
• Introduction au SEO : ▪ Core Vitals Web ▪ Les normes Google ▪ L'algorithme de racking de Google ▪ Les 3 piliers du SEO : le contenu, la popularité et la performance (ex : qualité du code etc.) ▪ Introduction à Google Search Console : indexation et suivi des performances SEO	*Evaluer les performances SEO d'un site web, en analysant les données fournies par Google Search Console et en appliquant les principes du Core Vitals Web, afin d'optimiser le référencement et les performances des pages*
• Les éléments de base d'une page web HTML5 : les balises de base, les titres, les métadonnées et les sections ▪ Les balises principales ▪ Les balises sémantiques et leur importance pour l'accessibilité (WCAG 2.1) et l'optimisation du SEO ▪ La gestion des métadonnées : la balise *<meta>* et l'attribut *<lang>* pour l'internationalisation et le SEO ▪ Introduction à la compatibilité mobile avec les attributs de la balise *<meta viewport>*	*Structurer une page web, en employant les balises HTML5 sémantiques, pour assurer une meilleure accessibilité et compréhension par les moteurs de recherche* *Configurer les balises et métadonnées HTML5, en intégrant les attributs <meta>, pour optimiser la compatibilité mobile et améliorer le SEO*
• Les formulaires HTML5 ▪ Les nouvelles balises avancées pour l'amélioration des formulaires (ex *: e.g. <input type="date">, <input type="email">*) ▪ Les règles de validation et de contraintes (ex : required, pattern, min, max etc.) ▪ Les formulaires accessibles et ergonomiques pour les utilisateurs d'appareils mobiles	*Améliorer l'accessibilité des formulaires, en intégrant des balises HTML5 avancées, afin de garantir une expérience utilisateur ergonomique et conforme aux normes mobiles*
• Présentation de CSS3 ▪ La syntaxe de base de CSS : les sélecteurs, les classes et les identifiants ▪ Les sélecteurs avancés : les pseudo-classes et les pseudos-éléments ▪ Le modèle de boîte (Box model) : la gestion des marges (margin), des bordures, des padding et de la mise en page ▪ La compatibilité entre les navigateurs et les outils de comptabilité de style (ex : Auto prefixer)	*Vérifier la compatibilité des styles CSS3, en utilisant des outils spécifiques et en ajustant automatiquement les préfixes, afin de garantir une expérience utilisateur cohérente sur différents navigateurs*
• Les outils Flexbox et Grid Layout ▪ Les alignements dynamiques et flexibles avec Flexbox ▪ Les mises en page complexes et adaptables avec Grid Layout ▪ Le tableau de bord administrateur multi composants : sidebar, graphique et tableau de données	*Concevoir des mises en page flexibles et dynamiques, en exploitant les outils Flexbox et Grid Layout, pour permettre une structuration adaptable aux différentes résolutions d'écran*

• Le responsive design avec CSS3 ▪ Introduction aux médias queries et à l'adaptation des styles en fonction de la taille de l'écran (ex : smartphone, tablette, ordinateur etc.) ▪ Les unités relatives (em, rem, %, vw, vh) pour des designs fluides ▪ Le chargement optimisé et progressif sur les petits écrans : l'approche mobile-first et progressive enhancement	*Appliquer des principes de responsive design, en recourant à CSS3 et aux médias queries, afin d'adapter le site web à différents types d'écrans*
• Les préprocesseurs CSS (SASS et LESS) ▪ Les variables CSS pour l'amélioration de la réutilisation des styles ▪ L'imbrication (Nesting) pour l'écriture propre et structurée du CSS ▪ Mixins : les blocs de styles réutilisables pour un code plus D.R.Y (Don't Repeat Yourself) ▪ L'automatisation de la compilation de fichiers CSS à partir de SASS et LESS dans des projets complexes : Gulp et Webpack	*Personnaliser les styles CSS, en recourant aux préprocesseurs, pour améliorer la réutilisation des blocs de styles et rendre le code plus maintenable*
• Les outils de développement HTML5 ▪ Les éditeurs de code : Visual Studio Code, les extensions Live Server, Prettier etc. ▪ Les navigateurs internet : Google Chrome, Firefox avec l'outil Devtools pour les tests de responsive design et la compatibilité des styles ▪ Les outils de build : Gulp et Webpack pour la compilation SASS/LESS	*Optimiser les performances de chargement d'une page web, en mettant en œuvre des techniques de minification et d'automatisation, pour améliorer la rapidité et l'efficacité du site sur tous les appareils*
3. Le langage JavaScript : manipulation du DOM, Ajax, jQuery	
• Introduction à Javascript ▪ L'intégration de JavaScript dans une page HTML : utilisation de la fonction *<script>* et des ES modules pour la structuration modulaire du code Javascript	*Intégrer du code JavaScript, en utilisant la balise <script> et les ES modules, pour structurer le code de manière modulaire dans une page HTML*
▪ Les éléments de bases Javascript : les variables, les types de données, les fonctions, les boucles et conditions, la manipulation des objets	*Définir des variables et manipuler des types de données JavaScript, en utilisant des structures de contrôle, afin d'exécuter des opérations complexes dans une application web*
▪ Introduction à ES6+ et aux fonctions let, cont, arrow functions et template literals	*Mettre en œuvre des fonctionnalités modernes de JavaScript, en exploitant les syntaxes ES6+, pour écrire du code plus concis et performant*
• La manipulation du Document Object Model (DOM) ▪ La sélection d 'éléments avec getElementById(), querySelector() et la manipulation du DOM en temps réel	*Manipuler des éléments du DOM, en utilisant des méthodes de sélection, pour interagir dynamiquement avec les éléments d'une page web*
▪ La modification dynamique des éléments : l'ajout, la suppression et la modification des attributs et du contenu	
• La gestion des événements via la délégation d'évènements	*Modifier le contenu d'une page web, en ajoutant, supprimant ou modifiant des attributs et des éléments*

▪ L'ajout d'évènement : les clics, le survol, la soumission de formulaires ▪ Les formulaires dynamiques : la modification et la validation des entrées en temps réel (ex : champs obligatoires, format prédéfini des emails etc.)	*DOM, afin de créer des interactions utilisateur dynamiques* *Gérer les événements d'une page web, en utilisant la délégation d'événements, pour capturer les actions des utilisateurs comme les clics et la soumission de formulaires et les traiter efficacement*
4. Ajax et Fetch API • Les envois de requêtes http avec les outils AJAX et Fecht API pour la récupération des données externes (JSON) sans rechargement de la page • La manipulation des données au format JSON • La gestion des erreurs dans les requêtes asynchrones (timeout, erreurs http etc.) • Introduction à l'authentification via JWT	*Dynamiser les interactions d'une page web, en utilisant des événements, pour rendre l'interface utilisateur plus interactive et réactive* *Envoyer des requêtes HTTP asynchrones, en utilisant AJAX et Fetch API, afin de récupérer des données externes au format JSON sans recharger la page, pour améliorer la réactivité des applications web* *Manipuler des données JSON, en les intégrant dans une page web via JavaScript, pour afficher dynamiquement les informations récupérées à partir d'une API externe* *Gérer les erreurs des requêtes asynchrones, en mettant en place des mécanismes de gestion et la vérification des statuts HTTP, afin de garantir la fiabilité des interactions réseau*
5. JQuery • Introduction à la bibliothèque jQuery pour simplifier la manipulation du DOM et les interactions utilisateurs • Les sélecteurs jQuery pour les animations et la gestion des événements • Les animations simples avec JQuery (ex : effet de glissement sur un menu etc.) • Les plugins jQuery (ex : DataTables) pour l'ajout de fonctionnalités avancées	*Authentifier les utilisateurs, en utilisant des JSON Web Tokens, pour sécuriser les échanges entre le client et le serveur dans des applications web* *Simplifier la manipulation du DOM, en utilisant la bibliothèque jQuery, afin de faciliter les interactions utilisateur et les modifications dynamiques des éléments d'une page web* *Sélectionner des éléments du DOM, en appliquant les sélecteurs jQuery, pour gérer les événements et effectuer des animations interactives*
6. Frameworks frontend : React, Vue.js, Angular • Rôle et importance d'un framework : la gestion du DOM virtuel, la réactivité, les composants réutilisables ▪ Introduction à React : JSX pour intégrer HTML dans javascript, les composants fonctionnels avec Hooks, la gestion du State. ▪ La gestion des états complexes avec useReducer pour les applications aux multiples interactions (ex : panier d'achat)	*Ajouter des fonctionnalités avancées à une page web, en intégrant des plugins jQuery, pour enrichir l'interactivité et les capacités de gestion des données* *Analyser le rôle d'un framework frontend, en utilisant React, Vue.js, et Angular, afin de gérer efficacement le DOM virtuel, la réactivité, et les composants réutilisables dans des applications web modernes* *Créer des composants dynamiques, en utilisant JSX dans React et les directives dans Vue.js, pour intégrer*

▪ Introduction à Vue.js : les directives, les composants dynamiques, Vue Router ▪ La création d'un composant simple avec React (ex : formulaire dynamique) et Vue.js (ex : liste de tâches interactives) ▪ Introduction à Angular ○ Vue d'ensemble de Angular en tant que framework complet : l'architecture, les modules, les composants etc. ○ Data Binding et Directives : Le two-way data binding, ngIf, ngFor ○ Le cycle de vie des composants Angular et comparaison avec ceux de React/Vue.js. ○ Le routage avec Angular Router : gestion des applications à page unique (SPA) ○ Les services et Dependency Injection : Introduction à l'injection de dépendances et la création de services pour la communication entre composants	*du HTML dans JavaScript et gérer l'interactivité des interfaces utilisateur* *Gérer les états complexes d'une application, en utilisant useReducer dans React et Vue Router dans Vue.js, pour développer des applications interactives avec des multiples interactions utilisateur* *Implémenter des fonctionnalités Angular, en utilisant le data binding et les directives, afin de manipuler dynamiquement les données et les affichages dans les Single Page Applications* *Gérer l'injection de dépendances et la communication entre composants, en utilisant les services d'Angular, pour optimiser la structuration et le partage des données au sein des applications*
• La gestion d'état (State management) ▪ La gestion d'état local et la communication entre les composants : les fonctions State et Props dans React et Vue.js ▪ Introduction à Redux et Vuex pour la gestion d'état global pour les applications complexes ▪ Angular : introduction à NgRx et les services et gestion d'état avec RxJS pour les applications complexes ▪ L'interactivité basée sur l'utilisateur : Les évènements et les modifications d'état ▪ La SPA et ses composants multiples	*Intégrer des services et gérer l'état global d'une application complexe, en utilisant Redux dans React et Vuex dans Vue.js, pour gérer efficacement la communication et les interactions entre les composants*
• La communication avec une API externe ▪ L'utilisation de Fetch dans React/Vue.js pour appeler une API externe et afficher dynamiquement les résultats ▪ La gestion des états asynchrones et l'affichage conditionnel en fonction des réponses de l'API ▪ L'affichage des données d'une API externe avec Vue.js et React (ex : catalogue de livres etc.) ▪ Angular : utilisation de Http Client pour consommer une API externe et la gestion des requêtes et des erreurs asynchrones	*Communiquer avec une API externe, en utilisant Fetch dans React et Vue.js, afin d'afficher dynamiquement les données récupérées depuis une API et gérer les états asynchrones*
• Les outils de frameworks Frontend ▪ Les outils de développement : Create React App, Vue CLI, Angular CLI, RxJS ▪ Les librairies associées : Axios, Redux, Vuex	*Utiliser des outils de développement, en employant Create React App, Vue CLI, et Angular CLI, afin de configurer rapidement des environnements de développement pour des applications front-end performantes*

Développement d'applications et solutions connectées

Contenu	Capacités attendues
Développement Backend et API : Node.js, PHP, SQL/NoSQL (90 à 100 h)	
Objectifs pédagogiques globaux : - **Mettre en œuvre des applications backend fiables en utilisant Node.js et Express pour gérer les requêtes HTTP, les routes, et les sessions de manière sécurisée** - **Concevoir des bases de données relationnelles (SQL) et non relationnelles (NoSQL) optimisées pour la gestion et la manipulation efficace de données dans des environnements backend** - **Développer des APIs RESTful sécurisées avec Node.js, Express, et PHP, en intégrant les bonnes pratiques d'authentification, de gestion des erreurs et de performance** - **Intégrer des bases de données SQL et NoSQL à des applications backend, en utilisant des ORM/ODM pour assurer une interaction fluide avec les données** - **Optimiser la sécurité, les performances, et la maintenance des APIs backend, en mettant en place des techniques avancées et en utilisant des outils de tests CI/CD**	
1. Node.js & Express : Programmation serveur, gestion des requêtes http • Introduction à Node.js ▪ Les concepts de base de Node.js : les évènements, single-thread, non-blocking I/O ▪ Les modules intégrés de Node.js : http, fs, path, os, pour la gestion de fichiers, de routes, et d'interactions avec le système ▪ Le module EventEmitter pour la gestion des événements asynchrones dans une application Node.js. ▪ Installation et utilisation de npm pour la gestion des dépendances • Programmation serveur Express.js ▪ La gestion des routes avec Express : les fonctions get, post, put, delete ▪ L'utilisation de middleware pour la gestion des erreurs, des authentifications et des sessions utilisateurs ▪ La gestion centralisée des erreurs et l'intégration de Winston et Morgan pour la journalisation (logs) des requêtes et des erreurs ▪ Les concepts de « *req* » et « *res* » et leurs rôles dans la manipulation des données utilisateurs ▪ La gestion des sessions et des cookies pour l'authentification ▪ Les websockets avec Node.js et Express : la gestion des connexions en temps réel (ex : chat, e-jeux en ligne etc.) ▪ Intégration de tests automatisés avec Jest et Mocha pour valider les fonctionnalités backend ▪ La mise en place d'un pipeline CI/CD avec GitHub Actions ou GitLab CI/CD pour tester et déployer automatiquement les applications Express.js	*Assimiler les concepts de base de Node.js, en manipulant les événements, le single-thread, et les I/O non bloquantes, afin de développer des applications serveur performantes* *Utiliser Node.js, en employant ses modules intégrés, afin de structurer les applications serveur* *Gérer les événements asynchrones dans une application serveur, en utilisant le module EventEmitter, pour améliorer la réactivité des applications* *Programmer des serveurs web, en utilisant Express.js pour gérer les routes HTTP, afin de manipuler et traiter les données des utilisateurs efficacement* *Assurer la sécurité et la gestion des erreurs dans une application serveur, en intégrant des middlewares, afin d'améliorer la fiabilité des applications* *Tester les fonctionnalités serveur, en utilisant Mocha et Jest, pour réaliser des tests unitaires, afin de garantir la stabilité des applications*

• APIs RESTful avec Node.js et Express.js ▪ La création d'API RESTful : la structuration des routes, la validation des données, la gestion des statuts HTTP ▪ La sécurisation des APIs avec les JWT (JSON Web Tokens) et les middlewares d'authentification (Postman) ▪ Implémentation de CORS pour permettre les requêtes inter-domaines : Helmet pour sécuriser l'API contre les vulnérabilités courantes (XSS, CSRF) ▪ La gestion des permissions et authentification avec OAuth 2.0 et API Gateway ▪ Chiffrement des données sensibles en transit avec TLS/SSL et en stockage avec AES-256 et RSA ▪ Automatisation des tests de sécurité des APIs avec OWASP ZAP et Burp Suite dans un pipeline CI/CD • Introduction aux microservices et à l'architecture découplée • Sécurisation des communications entre microservices avec Mutual TLS (mTLS) et gestion des certificats SSL/TLS • Les outils de développement backend ▪ Node.js pour la gestion des paquets ▪ Postman pour les tests de requêtes http ▪ Express.js pour la gestion des routes et des middlewares ▪ OWASP ZAP et Burp Suite pour les tests de sécurité des API	*Créer des APIs RESTful, en utilisant Node.js et Express.js, pour structurer les routes, valider les données et gérer les statuts HTTP dans des applications serveur* *Sécuriser les APIs, en implémentant des JWT et des middlewares de gestion d'authentification, afin de protéger les échanges de données et les utilisateurs*
2. Les bases de données SQL & NoSQL : MySQL, PostgreSQL, MongoDB • Introduction à la conception des bases de données : les modèles entités-relation, les règles de normalisation • Les bases de données relationnelles SQL ▪ Les concepts de tables, colonnes, lignes et clés primaires/étrangères ▪ Le langage SQL : les fonctions select, insert, update, delete, join pour la gestion et l'interrogation des données ▪ Introduction à MySQL et PostgreSQL : la gestion des bases de données, la création de schémas, la gestion des utilisateurs • Introduction aux bases de données NoSQL ▪ Les concepts de documents, de collections et les bases de données NoSQL	*Concevoir une base de données relationnelles, en appliquant les modèles entités-relation et les règles de normalisation, afin de structurer efficacement les données et garantir leur cohérence* *Manipuler des données dans une base relationnelle, en utilisant le langage SQL et ses fonctions, pour interroger et gérer les données dans des bases MySQL ou PostgreSQL* *Gérer une base de données MySQL ou PostgreSQL, en créant des schémas et en administrant les utilisateurs, pour assurer le bon fonctionnement et la sécurité des bases relationnelles* *Examiner les concepts de bases de données NoSQL, en étudiant les notions de documents et de collections,*

▪ MongoDB pour la manipulation des documents avec Mongoose (ODM pour MongoDB). ▪ SQL vs NoSQL : les divers cas d'usage des bases NoSQL • L'intégration d'une base de données avec une API ▪ La connexion de Node.js à une base de données MySQL ou MongoDB ▪ L'utilisation de ORM (Object-Relational Mapping) avec Sequelize pour MySQL/PostgreSQL ou Mongoose pour MongoDB ▪ La sécurisation des bases de données contre les injections SQL en utilisant des requêtes paramétrées et ORM sécurisé ▪ La gestion des erreurs liées à la connexion et la sécurité des données (ex : injections SQL, etc.) ▪ Le chiffrement des données stockées avec AES-256 et gestion des clés avec AWS KMS ▪ Implémentation des rôles utilisateurs avec RBAC (Role-Based Access Control) et gestion des accès via OAuth 2.0 ▪ Surveillance et audit des accès avec logs automatisés et alertes en cas de comportement suspect (SIEM, CloudWatch, Azure Monitor) ▪ La protection des bases de données contre les injections SQL avec des requêtes paramétrées et ORM sécurisé ▪ L'optimisation des performances des bases de données : indexation, normalisation, partitionnement • La gestion des transactions et des verrous : les mécanismes de verrouillage pour assurer l'intégrité des données • La gestion des rôles et des permissions utilisateurs : les systèmes multi-utilisateurs • Implémentation des rôles utilisateurs avec RBAC (Role-Based Access Control) et gestion des accès avec OAuth 2.0 • La sécurité des bases de données : la gestion des permissions, le chiffrage de données sensibles etc. • Implémentation d'un monitoring de sécurité avec audit logs pour détecter les accès suspects • La gestion des sauvegardes et de la réplication des bases de données • La sécurisation des sauvegardes avec chiffrement et politique de rétention des données sensibles • Sécurisation des sauvegardes avec chiffrement et politique de rétention des données sensibles	*afin de travailler efficacement avec des systèmes comme MongoDB* *Utiliser MongoDB, en exploitant l'ODM Mongoose, pour manipuler des documents et gérer des bases NoSQL dans des cas d'utilisation spécifiques* *Intégrer une base de données à une application Node.js, en utilisant des ORM, afin de faciliter la gestion des données entre l'application et la base de données* *Optimiser les performances d'une base de données, en appliquant des techniques d'indexation, de normalisation et de partitionnement, afin d'améliorer la vitesse et l'efficacité des requêtes* *Assurer l'intégrité des données, en utilisant des mécanismes de gestion des transactions et de verrouillage, pour garantir la fiabilité des opérations dans des environnements multi-utilisateurs* *Sécuriser les bases de données, en mettant en place des permissions d'accès, du chiffrement de données sensibles et en les protégeant contre les injections SQL, pour garantir la sécurité des données et des utilisateurs* *Gérer les sauvegardes et la réplication des bases de données, en mettant en place des plans de reprise après sinistre, pour garantir la disponibilité des données en cas de défaillance système*

• Les plans de reprise après sinistre (Disaster Recovery)
• Les outils pour la gestion des bases de données
  ▪ MySQL Workbench et pgAdmin pour la gestion des bases de données relationnelles
  ▪ MongoDB Compass pour l'exploration de bases de données MongoDB
  ▪ Sequelize et Mongoose pour l'ORM et l'ODM
  ▪ Audit de sécurité des bases de données avec SQLMap et MongoDB Security Checklist
• Le développement d'une API de gestion d'un catalogue avec une base de données relationnelle MySQL
• La base de données MongoDB pour le stockage des profils utilisateurs et les préférences dans une application de réseau social	

3. PHP et API RESTful : le PHP moderne avec des Frameworks (Laravel et Symfony)
• Présentation de PHP moderne
  ▪ Les évolutions de PHP (version 7.x et 8.x)
  ▪ La programmation orientée objet (POO) avec PHP : les classes, les objets, l'héritage, les interfaces
• Les frameworks PHP
  ▪ Introduction à Laravel et Symfony : l'architecture MVC (Modèle-Vue-Contrôleur), le routage, les middlewares
  ▪ La création d'APIs RESTful avec PHP : la gestion des requêtes HTTP, la structuration des routes et des contrôleurs
  ▪ Eloquent ORM (Laravel) pour la gestion des interactions avec une base de données relationnelle (MySQL ou PostgreSQL)
• La sécurité et les performances des API PHP
  ▪ Introduction aux bonnes pratiques de sécurisation des API avec PHP (gestion des tokens, protection CSRF etc.)
  ▪ Intégration de l'authentification OAuth 2.0 et JWT pour la sécurisation des API PHP
  ▪ La mise en place de HTTPS et TLS/SSL pour sécuriser les connexions serveur-client
  ▪ Automatisation des tests de sécurité et validation de code avec GitLab CI/CD et SonarQube
  ▪ L'optimisation des performances avec des techniques comme le caching, les queues (avec Redis ou RabbitMQ)
  ▪ Introduction à la gestion des API GraphQL avec Laravel pour permettre des requêtes dynamiques
	Analyser les évolutions de PHP, en analysant les versions 7.x et 8.x, afin d'utiliser les dernières fonctionnalités optimisées du langage
	
	Programmer en PHP orienté objet, en utilisant les classes, l'héritage, et les interfaces, pour structurer les applications selon les principes de la POO
	
	Développer des applications web, en utilisant les frameworks Laravel ou Symfony, avec l'architecture MVC, afin d'organiser le code de manière modulaire et efficace
	
	Créer des APIs RESTful en PHP, en structurant les routes et les contrôleurs, pour gérer les requêtes HTTP et permettre une communication efficace entre le client et le serveur
	
	Gérer les interactions avec une base de données relationnelle, en utilisant Eloquent ORM, afin de simplifier les requêtes SQL et automatiser la gestion des données
	
	Sécuriser les APIs PHP, en mettant en place des pratiques de sécurité comme la gestion des tokens et la protection CSRF, pour protéger les échanges de données et prévenir les attaques
	
	Optimiser les performances des APIs, en utilisant des techniques de caching et des queues, pour améliorer la vitesse d'exécution des requêtes et la fluidité des applications

▪ L'optimisation des requêtes complexes SQL et GraphQL ▪ Sécurisation des API GraphQL : gestion des limites de requêtes pour éviter les abus ▪ La sécurisation des API GraphQL : la gestion des limites de requêtes pour éviter les abus ▪ Les méthodes et techniques avancées de cache (ex : Redis) pour l'optimisation des performances des applications ▪ Surveillance des performances backend avec ELK Stack (Elasticsearch, Logstash, Kibana) et Prometheus ▪ TDD (Test-Driven Development) : introduction au développement piloté par les tests (TDD) avec PHPUnit pour les tests d'API Laravel ▪ Les méthodes de déploiement et CI/CD : les services GitLab CI, Travis CI, ou Jenkins, pour automatiser les tests et les déploiements des applications PHP sur des serveurs de production ▪ Le déploiement des applications sur le cloud et la gestion des bases de données distantes : AWS, Heroku DigitalOcean etc. • Les outils PHP et API ▪ Intégration de GitLab CI/CD avec des tests de sécurité automatisés (SAST, DAST, SonarQube) ▪ Composer pour la gestion des dépendances PHP ▪ Laravel et Symfony pour le développement d'APIs ▪ Postman pour tester les requêtes d'API	*Tester les APIs PHP, en appliquant le développement piloté par les tests TDD, afin de garantir la fiabilité du code et automatiser les tests dans un processus CI/CD* *Déployer des applications PHP sur le cloud, en utilisant des plateformes dédiées, pour assurer la disponibilité et la scalabilité des applications dans des environnements distants*
Contenu	**Capacités attendues**
Développement avancé d'API et microservices : architectures API-first, GraphQL et serverless (40 à 50 h)	
Objectifs pédagogiques globaux : - **Développer des API modernes en appliquant les principes avancés des architectures RESTful, API-First et GraphQL, tout en optimisant les performances et la scalabilité** - **Mettre en œuvre une architecture microservices sécurisée en utilisant Node.js, Spring Boot, API Gateway et des modèles de communication modernes pour assurer l'interopérabilité et la résilience des services** - **Déployer des applications en environnement Serverless en exploitant AWS Lambda, Firebase Functions et des bases de données Serverless, pour garantir une haute disponibilité et une gestion optimisée des coûts** - **Automatiser les tests et les déploiements via CI/CD, GitHub Actions, GitLab CI/CD et Docker/Kubernetes, afin d'assurer la qualité logicielle et la fluidité des mises en production** - **Renforcer la sécurité des API et microservices en intégrant des protocoles d'authentification avancés, en appliquant des tests de vulnérabilités et des stratégies de monitoring**	
1. Conception avancée d'API : RESTful, API-First et GraphQL • Les principes avancés des API RESTful	

▪ Rappels sur l'architecture REST : stateless, CRUD, HATEOAS ▪ Versioning et gestion des évolutions d'API (URI, Headers, Content-Type) ▪ La mise en cache et l'optimisation : Cache-Control, ETags, Redis ▪ La protection des API contre les abus : Rate-limiting, Throttling, CORS	*Développer des API RESTful avancées, en implémentant les principes de stateless, HATEOAS et la gestion des versions, afin d'assurer une évolutivité et une interopérabilité optimales*
• L'approche API-First et sa mise en œuvre ▪ Définition et avantages de l'API-First ▪ L'utilisation d'OpenAPI (Swagger) et Postman pour la documentation et la validation d'API ▪ Le design des endpoints selon les bonnes pratiques : modularité et évolutivité	*Mettre en œuvre l'approche API-First, en utilisant OpenAPI et Postman, afin de structurer la documentation et garantir la compatibilité des endpoints*
• Le développement d'API GraphQL ▪ Les différences entre REST et GraphQL ▪ La mise en place d'un serveur GraphQL avec Apollo Server et Node.js ▪ L'optimisation des requêtes avec DataLoader et Batch Processing ▪ La sécurisation des API GraphQL : gestion des permissions et des rôles	*Créer des API GraphQL, en exploitant Apollo Server et l'optimisation des requêtes avec DataLoader, afin d'améliorer les performances et réduire la surcharge réseau*
2. Microservices : architecture, communication et orchestration • Les principes fondamentaux des microservices ▪ Monolithique vs Microservices ▪ Les Patterns d'architecture microservices : Event-Driven, CQRS, SAGA ▪ Introduction aux API Gateway (Kong, Traefik, AWS API Gateway)	*Concevoir une architecture microservices, en intégrant les patterns Event-Driven, CQRS et API Gateway, afin de rendre l'application modulaire et scalable*
• Développement de microservices avec Node.js et Spring Boot ▪ La création de microservices avec Express.js et Spring Boot ▪ L'interconnexion des services via REST, gRPC et WebSockets ▪ La gestion de la persistance : les bases de données spécifiques par service (MySQL, PostgreSQL, MongoDB)	
• Sécurisation et authentification des microservices ▪ L'implémentation de OAuth 2.0 et JWT pour l'authentification des services ▪ La sécurisation des communications avec Mutual TLS (mTLS) ▪ La mise en place de rôles et permissions avec RBAC (Role-Based Access Control)	*Sécuriser les API et microservices, en intégrant OAuth 2.0, JWT et Mutual TLS, afin de garantir la confidentialité et l'authentification des échanges*
• La gestion des échanges entre microservices ▪ Kafka et RabbitMQ pour la gestion des événements asynchrones	

▪ Les Patterns de communication : Request-Response, Publish-Subscribe ▪ La gestion des erreurs et résilience : Circuit Breaker, Retry Pattern **3. Déploiement et scalabilité des API en Serverless** • Introduction au Serverless et cas d'usage ▪ Définition du Serverless ? ▪ Les avantages et les limites du Serverless : scalabilité automatique, coûts réduits, absence de gestion d'infrastructure ▪ Les cas d'usage de AWS Lambda, Firebase Functions ou Azure Functions • Le déploiement d'API Serverless avec AWS Lambda et Firebase ▪ La création de fonctions Serverless avec AWS Lambda et Firebase Functions ▪ La gestion des triggers : API Gateway, DynamoDB Streams, Cloud Storage Events ▪ La connexion à une base de données Serverless (Firestore, DynamoDB) ▪ Le monitoring et la gestion des logs avec AWS CloudWatch et Firebase Monitoring • L'optimisation et la sécurité des API Serverless ▪ La gestion des permissions et rôles avec AWS IAM ▪ La protection contre le cold start avec provisioned concurrency ▪ La sécurisation des appels API avec AWS WAF et OAuth2.0 ▪ L'implémentation de tests Serverless avec Serverless Framework et SAM **4. Automatisation et DevOps pour API et microservices** • La mise en place de CI/CD pour les API ▪ Automatisation des tests et déploiements avec GitHub Actions, GitLab CI/CD et Jenkins ▪ Déploiement des API dans des containers Docker et orchestration avec Kubernetes ▪ Infrastructure as Code (IaC) avec Terraform et AWS CloudFormation • Les tests de performance et de sécurité des API ▪ Les tests de charge avec K6 et Artillery ▪ Les scanners de vulnérabilités API avec OWASP ZAP et Snyk ▪ La mise en place de Security-Driven Development (SDD)	*Déployer des services Serverless, en utilisant AWS Lambda, Firebase Functions et les API Gateway, afin d'optimiser l'exécution sans gestion d'infrastructure* *Configurer l'automatisation des tests et du déploiement en intégrant GitHub Actions, GitLab CI/CD et Docker, afin d'assurer une livraison continue fiable* *Tester la performance et la sécurité des API, en réalisant des tests de charge et d'intrusion avec K6, OWASP ZAP et SonarQube, afin de garantir la solidité des services*
Contenu	**Capacités attendues**
Développement mobile natif et hybride : iOS, Android, Flutter (60 à 70 h)	

Développement d'applications et solutions connectées

Objectifs pédagogiques globaux :

- **Maîtriser la conception d'interfaces utilisateur mobiles à l'aide de Figma, en tenant compte des bonnes pratiques de design pour les applications mobiles**
- **Développer des applications mobiles multiplateformes performantes en utilisant React Native, tout en intégrant des fonctionnalités natives comme la caméra et la géolocalisation**
- **Utiliser Flutter pour créer des applications mobiles optimisées, avec une gestion efficace des widgets et des états, en respectant les spécificités de chaque plateforme (iOS/Android)**
- **Mettre en œuvre des tests unitaires et end-to-end (E2E) pour garantir la qualité des applications développées avant leur mise en production.**
- **Déployer des applications mobiles sur les stores en utilisant les outils de CI/CD pour automatiser le processus de tests et de déploiement**

5. Introduction au maquettage d'applications sur Figma • Présentation de Figma ▪ Les fonctionnalités principales ▪ Les concepts de bases du design d'interface ▪ Les principes d'ergonomie ▪ Les fonctionnalités de collaboration d'équipe et de partage en temps réel	*Maîtriser les concepts fondamentaux du design d'interface, en explorant les fonctionnalités principales de Figma, afin de concevoir des interfaces utilisateur modernes et ergonomiques* *Créer des maquettes d'applications, en utilisant les outils de conception de Figma, pour planifier et organiser les interfaces avant leur développement* *Collaborer sur des projets de design, en utilisant les fonctionnalités de partage et de travail en temps réel de Figma, afin de faciliter les échanges d'idées au sein des équipes de développement* *Optimiser la présentation visuelle d'une application, en appliquant les principes d'ergonomie et de design d'interface de Figma, pour améliorer l'expérience utilisateur et la cohérence du design*
6. React Native : Développement d'applications multiplateformes (iOS/Android) • Introduction à la multiplateforme React Native : historique et contexte ▪ React native vs Swift pour iOs et Kotlin/Java pour Android : leurs avantages et inconvénients ▪ L'architecture des composants React Native : les composants réutilisables • L'écosystème React Native ▪ Installation et configuration de l'environnement de développement : Node.js, npm, Expo CLI, Xcode pour iOS, Android Studio pour Android ▪ Les API React Native pour accéder aux fonctionnalités natives (ex : caméra, GPS, notifications push, etc.) ▪ La sécurisation des échanges entre mobile et backend avec TLS/SSL et Mutual TLS (mTLS) ▪ La gestion des permissions pour accéder aux fonctionnalités sensibles (ex. : appareil photo,	*Evaluer les avantages et les inconvénients de React Native en le comparant à Swift pour iOS et Kotlin/Java pour Android, afin de choisir la meilleure approche pour le développement d'applications multiplateformes* *Configurer un environnement de développement React Native en utilisant Node.js, npm, Expo CLI, Xcode, et Android Studio, pour développer des applications mobiles sur iOS et Android* *Accéder aux fonctionnalités natives des téléphones, en utilisant les API de React Native, afin d'intégrer des services telles que la caméra, le GPS, et les notifications push dans des applications mobiles* *Créer des interfaces utilisateurs réactives, en utilisant les composants intégrés de React Native, pour*

géolocalisation) en appliquant le principe du moindre privilège • La création des composants et la gestion de l'interface utilisateur ▪ Les composants intégrés : View, Text, Image, ScrollView, FlatList ▪ Les concepts de StyleSheet pour le stylisme des composants ▪ La responsivité et l'adaptabilité des interfaces : dimensions, PixelRatio, et la gestion des tailles d'écran ▪ Flexbox pour le positionnement des éléments sur l'écran mobile • La gestion des états et des données ▪ Introduction à state et props dans React Native ▪ Utilisation des Hooks (useState, useEffect) pour la gestion des états locaux et des effets secondaires ▪ Introduction à Redux et Context API pour la gestion globale des états dans une application mobile complexe ▪ La protection des données stockées localement avec chiffrement AES et stockage sécurisé (SecureStore, EncryptedSharedPreferences) • La navigation dans une application mobile ▪ Introduction à React Navigation pour la gestion des routes et la navigation entre les écrans ▪ Les différents types de navigation : Stack navigation, Tab navigation, Drawer navigation etc. ▪ La passation de données entre les écrans et la gestion des transitions • L'accès aux APIs et l'intégration de services externes ▪ Utilisation de Fetch API et Axios pour communiquer avec des APIs RESTful externes ▪ La sécurisation des communications entre l'application mobile et le backend avec OAuth 2.0 et JWT ▪ L'implémentation de CORS et restrictions d'accès API pour empêcher les requêtes non autorisées ▪ La gestion des permissions d'accès aux fonctionnalités du téléphone (ex : appareil photo, géolocalisation, etc.) • Les outils spécifiques au développement mobile ▪ Visual Studio Code et WebStorm	*concevoir des applications adaptées aux différentes tailles d'écran* *Styliser des composants React Native, en utilisant StyleSheet et Flexbox, afin de gérer la responsivité et la position des éléments dans les interfaces mobiles* *Gérer les états locaux d'une application mobile, en utilisant les Hooks, pour contrôler l'interaction entre les composants et les effets secondaires* *Implémenter la gestion globale des états, en utilisant Redux ou Context API, pour gérer les états complexes dans des applications mobiles comportant de multiples interactions* *Naviguer entre les écrans d'une application mobile, en utilisant React Navigation, pour gérer les routes et la transmission de données entre les différentes interfaces* *Communiquer avec des APIs externes, en utilisant Fetch API et Axios, pour récupérer et afficher dynamiquement des données dans une application mobile* *Tester des applications mobiles React Native, en utilisant Expo CLI sur un simulateur ou un appareil*

▪ Expo CLI pour le développement et le test rapide sur un simulateur ou un appareil réel ▪ React Navigation pour la gestion de la navigation dans les applications React Native ▪ Redux ou Context API pour la gestion des états globaux	*réel, afin de valider les fonctionnalités et l'ergonomie des applications avant leur déploiement*
7. Flutter : développement d'applications mobiles performantes • Introduction à Flutter ▪ Présentation de Flutter (Google), de Dart et son langage sous-jacent ▪ Les avantages de Flutter : Interface, widgets, développement unifié pour iOS et Android ▪ Flutter vs React Native : les différences en termes de performances et d'écosystème	*Appréhender les bases de Flutter et son langage sous-jacent, en comparant les avantages avec ceux de React Native, afin de choisir la meilleure technologie de développement d'applications mobiles multiplateformes*
• Installation et configuration de Flutter ▪ Les méthodes d'installation de Flutter SDK et de Dart ▪ La configuration de Android Studio et Xcode pour le développement iOS et Android	*Installer et configurer Flutter SDK et Dart, en configurant Android Studio et Xcode, pour préparer l'environnement de développement sur iOS et Android*
• La création de widgets dans Flutter ▪ Introduction aux widgets stateless et stateful ▪ Les widgets pré intégrés (Container, Column, Row, ListView etc.) ▪ Le stylisme et la personnalisation des widgets avec BoxDecoration, TextStyle, etc. ▪ La gestion des tailles d'écran et de la densité de pixels pour les interfaces adaptatives	*Créer des interfaces utilisateurs réactives, en utilisant les widgets stateless et stateful dans Flutter, afin de construire des applications performantes avec une interface unifiée* *Styliser les composants Flutter, en utilisant les widgets pré intégrés, pour personnaliser et adapter les interfaces aux différents écrans et densités de pixels*
• La gestion des états et du cycle de vie dans Flutter ▪ La gestion des états avec StatefulWidget et l'utilisation des classes State ▪ Introduction à Provider pour la gestion des états globaux ▪ StreamBuilder et FutureBuilder pour la gestion des états asynchrones	*Gérer les états d'une application mobile, en utilisant StatefulWidget, les classes State, et Provider, afin de contrôler les interactions et la réactivité des composants dans Flutter* *Optimiser la gestion des états asynchrones, en utilisant StreamBuilder et FutureBuilder, pour traiter les données et les flux dans des applications performantes et réactives*
• La navigation et la gestion des routes dans Flutter ▪ Introduction à Navigator et MaterialPageRoute pour la gestion de la navigation ▪ La gestion des routes dynamiques et de la navigation imbriquée	*Naviguer entre les écrans d'une application mobile, en utilisant Navigator et MaterialPageRoute, pour gérer les routes dynamiques et la navigation imbriquée dans les applications Flutter*
• L'accès aux APIs externes et l'intégration de services natifs ▪ L'utilisation de Dio et http pour appeler des APIs RESTful externes ▪ La gestion des permissions et l'intégration des services natifs (ex : accès à la caméra, aux fichiers locaux, aux notifications etc.) ▪ La sécurisation des requêtes API avec OAuth 2.0 et la gestion des authentifications	*Intégrer des services natifs dans une application Flutter, en appelant des APIs RESTful externes, afin de récupérer des données et interagir avec les fonctionnalités du téléphone*

utilisateurs via Firebase Authentication ou Auth0 ▪ La mise en place d'un stockage sécurisé des jetons d'accès avec Secure Storage sur Android et Keychain Services sur iOS • Les différents outils de l'environnement Flutter ▪ Flutter SDK et Dart comme environnement de développement ▪ Android Studio ou VS Code comme IDE ▪ Provider ou Riverpod pour la gestion des états	*Utiliser des outils de développement, en utilisant Flutter SDK, Dart, Android Studio, et VS Code, afin de créer un environnement de travail adapté au développement d'applications mobiles multiplateformes*
8. Tests et déploiement d'applications mobiles • Tests unitaires dans les applications mobiles ▪ Introduction aux tests unitaires pour les composants React Native et Flutter ▪ React Native : Utilisation de Jest et React Native Testing Library pour tester les composants, les états, et les interactions ▪ Flutter : utilisation de Flutter Test et WidgetTester pour les tests unitaires des widgets et la gestion des états ▪ Les bonnes pratiques de test : création de tests modulaires, automatisation des tests pour des déploiements rapides ▪ Les tests de sécurité automatisés pour détecter les failles potentielles en phase de développement	*Réaliser des tests unitaires pour les composants en utilisant Jest, React Native Testing Library, Flutter Test et WidgetTester, afin de vérifier la fonctionnalité des composants, des états, et des interactions dans les applications mobiles*
• Tests end-to-end (E2E) ▪ Introduction aux tests end-to-end pour simuler des interactions utilisateur complètes ▪ React Native : utilisation de Detox pour les tests E2E sur des applications React Native ▪ Flutter : utilisation de Flutter Driver pour les tests de bout en bout sur des applications Flutter ▪ Les tests de pénétration automatisés avec OWASP Mobile Security Testing Guide (MSTG) pour identifier les vulnérabilités critiques	*Appliquer les bonnes pratiques de test, en créant des tests modulaires et en les automatisant, pour améliorer la qualité des applications mobiles et accélérer le processus de déploiement* *Simuler des interactions complètes utilisateur, en effectuant des tests end-to-end, afin de valider l'intégrité de l'application à travers différents scénarios d'utilisation*
• Déploiement sur les stores : App Store (IOS) et Google Play (Android) ▪ La préparation des applications pour le déploiement ○ La création des builds (APK pour Android, IPA pour iOS) pour publication ○ Les fichiers de configuration pour chaque plateforme (AndroidManifest.xml pour Android, Info.plist pour iOS) ○ Les certifications et signatures requises pour le déploiement sur les stores	*Préparer les applications mobiles pour le déploiement, en générant des builds et en configurant les fichiers nécessaires pour chaque plateforme, afin d'assurer la compatibilité avec les différents stores*

○ La configuration des certificats SSL/TLS pour garantir des connexions sécurisées avant le déploiement • Les processus de soumission ▪ La procédure sur le Google Play Store ○ La gestion des permissions ○ La vérification des politiques ○ La soumission de l'application. ▪ La procédure sur l'App Store ○ Le respect des règles d'Apple ○ La gestion des screenshots et des métadonnées. • Les bonnes pratiques pour le déploiement ▪ La gestion des mises à jour, les stratégies de rollback en cas de bug lors de la mise en production • L'automatisation du déploiement ▪ Introduction aux outils d'intégration continue/déploiement continu (CI/CD) pour le déploiement d'applications mobiles ▪ Utilisation de Fastlane pour automatiser le processus de déploiement sur les stores ▪ Intégration avec des services (ex : GitHub Actions ou CircleCI) pour automatiser les tests et les déploiements ▪ La vérification de la conformité aux politiques de sécurité des stores avant validation (Google Play Protect, App Transport Security - ATS d'Apple)	*Gérer le processus de soumission sur le Google Play Store, en respectant les politiques de Google et en soumettant les permissions requises, pour assurer la conformité et la validation de l'application* *Respecter les règles de l'App Store, en gérant les screenshots et les métadonnées et en assurant la conformité avec les directives d'Apple, pour publier les applications sur iOS* *Déployer des applications mobiles, en appliquant les bonnes pratiques de mise à jour et les stratégies de rollback, pour assurer une gestion efficace des nouvelles versions et des corrections de bugs en production* *Automatiser le déploiement des applications mobiles, en utilisant Fastlane, afin de simplifier et accélérer le processus de publication sur les stores Android et iOS* *Intégrer des outils CI/CD, en utilisant des services comme GitHub Actions ou CircleCI, pour automatiser les tests et les déploiements dans les environnements de production*
Contenu	**Capacités attendues**
Tests logiciels et automatisation QA (25 à 30 h)	
Objectifs pédagogiques globaux : - **Identifier les différents types de tests automatisés et leur rôle dans l'assurance qualité des applications front-end, back-end, et mobile** - **Implémenter des tests unitaires et d'intégration pour valider le bon fonctionnement des composants, des services, et des interactions entre les modules d'une application** - **Automatiser les tests end-to-end (E2E) pour simuler des scénarios utilisateurs complets et s'assurer que l'application répond aux exigences fonctionnelles en tout point** - **Intégrer les tests automatisés dans les pipelines CI/CD, pour garantir que chaque modification de code est validée automatiquement, et améliorer ainsi la qualité continue des livrables** - **Appliquer des pratiques avancées et la gestion des environnements de test, pour améliorer la fiabilité et la maintenabilité des systèmes**	
1. Introduction aux tests automatisés • Les différents types de tests : tests unitaires, tests d'intégration, tests E2E, tests de performance. • L'importance des tests automatisés dans le cycle de développement logiciel	*Identifier les différents types de tests, en analysant leurs caractéristiques, afin de choisir le test le plus approprié pour chaque étape du développement*

• Introduction aux tests de sécurité (Static Application Security Testing - SAST et Dynamic Application Security Testing - DAST) pour détecter les vulnérabilités dans le code source.	*Expliquer l'importance des tests automatisés, en étudiant leur impact sur le cycle de développement logiciel, pour améliorer la qualité du code et éviter les régressions*
2. Les tests automatisés front-end • Jest et React Testing Library pour les tests unitaires et d'intégration • La mise en œuvre de tests E2E avec Cypress ou Puppeteer • Les tests de performance front-end pour s'assurer que les interfaces utilisateurs répondent rapidement aux interactions • Les tests de sécurité sur les interfaces utilisateur avec OWASP ZAP pour détecter les failles XSS et CSRF • La validation de la protection des entrées utilisateur contre les attaques par injection de code	*Réaliser des tests unitaires et d'intégration, en utilisant Jest et React Testing Library, afin de valider le bon fonctionnement des composants front-end* *Effectuer des tests E2E, en utilisant Cypress ou Puppeteer, pour simuler des interactions utilisateur complètes et s'assurer du bon fonctionnement des interfaces*
3. Les tests automatisés back-end • Introduction à Mocha, Chai, et Jest pour tester les API et les services backend • Les tests d'intégration pour valider la communication entre les modules backend (ex : base de données et API) • Les tests de charge pour tester la performance des serveurs • Pentesting automatisé des API REST avec OWASP ZAP et Postman Security Tests. • Les tests d'injection SQL et détection des failles liées aux autorisations non sécurisées. • La vérification de l'implémentation des mécanismes de gestion des sessions et des cookies sécurisés (Secure, HttpOnly, SameSite)	*Tester les API et les services backend, en utilisant Mocha, Chai, et Jest, afin de garantir la validité des réponses aux requêtes* *Valider la communication entre les modules backend, en réalisant des tests d'intégration, pour s'assurer que la base de données et les API fonctionnent en harmonie* *Évaluer la performance des serveurs backend, en exécutant des tests de charge, pour identifier les points de saturation et optimiser la gestion des ressources*
4. Les tests automatisés dans les pipelines CI/CD • L'intégration des tests dans les pipelines de déploiement continu (CI/CD) avec Jenkins, GitLab CI, GitHub Actions etc. • La mise en œuvre de l'automatisation des tests afin de garantir que chaque modification de code est validée avant d'être déployée en production • L'intégration des scanners de vulnérabilités dans les pipelines CI/CD (Snyk, SonarQube, GitHub Security). • L'automatisation des tests de sécurité API avec OWASP Dependency-Check et Trivy pour détecter les dépendances vulnérables	*Intégrer des tests automatisés dans les pipelines de CI/CD, en utilisant Jenkins, GitLab CI, ou GitHub Actions, pour automatiser la validation de chaque modification du code avant déploiement* *Garantir la validation continue des modifications de code, en automatisant les tests dans le déploiement, afin de réduire les risques d'erreurs en production* *Améliorer la qualité des livrables, en optimisant les pipelines CI/CD, pour garantir un cycle de développement fiable*
5. Les pratiques avancées de tests automatisés	

• La mise en place du Test-Driven Development (TDD) • La gestion des environnements de test, l'isolation des tests, des mocks et stubs • La surveillance des tests en production pour détecter les régressions • L'utilisation du Security-Driven Development (SDD) pour intégrer la sécurité dès la conception des tests • La détection des comportements suspects en production via des outils de monitoring de logs de sécurité (ELK Stack, Splunk)	*Mettre en œuvre le Test-Driven Development, en rédigeant les tests avant d'écrire le code, afin de s'assurer que chaque fonctionnalité réponde aux exigences spécifiées* *Gérer les environnements de test, en isolant les tests avec des mocks et stubs, pour garantir des tests fiables et indépendants des autres composants* *Surveiller les tests en production, en analysant les résultats des tests automatisés, pour détecter les régressions et maintenir la qualité des systèmes en continu*
Contenu	**Capacités attendues**
Sécurisation des applications web et mobiles : OWASP, RGPD, API Security (40 à 50 h)	
Objectifs pédagogiques globaux : - **Identifier les vulnérabilités des applications web et mobiles en s'appuyant sur l'OWASP Top 10, afin de prévenir les risques de cybersécurité et renforcer la protection des systèmes** - **Mettre en œuvre des stratégies de sécurisation des applications et des API en appliquant des bonnes pratiques d'authentification, de gestion des accès et de chiffrement, afin de garantir l'intégrité et la confidentialité des échanges de données** - **Automatiser la détection et la correction des failles de sécurité en intégrant des outils de test et d'audit dans un pipeline DevSecOps, afin d'assurer une protection continue des applications** - **Assurer la conformité RGPD en intégrant les principes de Privacy by Design et en mettant en place des solutions de gestion des consentements, afin de garantir la transparence et le respect des droits des utilisateurs** - **Évaluer et améliorer la conformité réglementaire et sécuritaire des applications en réalisant des audits de protection des données et en supervisant les pratiques de sous-traitance cloud, afin de proposer des plans d'action correctifs adaptés**	
1. Sécurité avancée des applications web et mobiles : les principales menaces de sécurité (OWASP, attaques XSS, CSRF, injections) • Introduction aux menaces de cybersécurité ▪ Les risques de sécurité applicative et les principales vulnérabilités ▪ Les attaques les plus courantes et leur impact ▪ OWASP Top 10 : panorama des failles majeures affectant les applications web et mobiles • La protection contre les attaques XSS (Cross-Site Scripting) ▪ L'identification des types d'attaques XSS (stocké, réfléchi, DOM-based) ▪ Les stratégies de mitigation ○ La validation et l'échappement des entrées utilisateurs La mise en place de politiques de sécurité HTTP (Content Security Policy - CSP)	*Analyser les risques de sécurité applicative, en identifiant les vulnérabilités majeures selon l'OWASP Top 10, afin d'anticiper les attaques potentielles* *Détecter les attaques XSS, en utilisant des outils d'analyse et de simulation, afin de proposer des mesures de protection adaptées*

Développement d'applications et solutions connectées

○ Les tests et les simulations d'attaques XSS sur des applications web réelles • La protection contre les attaques CSRF (Cross-Site Request Forgery) ▪ L'impact des attaques CSRF ▪ Les mécanismes de protection avancés ○ L'implémentation de tokens CSRF et la validation des sessions utilisateurs ○ Les stratégies avancées anti-CSRF dans les Frameworks modernes (Django, Spring Security, Express.js etc.)	*Mettre en œuvre des stratégies de mitigation contre les attaques XSS et CSRF, en appliquant les politiques de sécurité HTTP et en intégrant les tokens anti-CSRF, afin de renforcer la robustesse des applications web*
• Sécurisation avancée des API REST et GraphQL ▪ L'identification des vulnérabilités spécifiques aux API	*Identifier les vulnérabilités spécifiques aux API REST et GraphQL, en réalisant des tests d'intrusion, afin d'évaluer les risques et améliorer la sécurité des services exposés*
▪ La mise en place des bonnes pratiques en authentification et gestion des accès ○ OAuth 2.0, JWT (JSON Web Tokens) et API Gateway ○ Le chiffrement des données en transit (HTTPS, TLS/SSL, Mutual TLS).	*Chiffrer les communications et les données en transit, en utilisant HTTPS, TLS/SSL et Mutual TLS, afin de garantir la confidentialité et l'intégrité des informations échangées*
▪ La protection contre les abus d'API ○ Implémentation de quotas et rate-limiting (ex : Cloudflare, API Gateway etc.) ○ La gestion des CORS (Cross-Origin Resource Sharing) et la sécurisation des requêtes	*Protéger les API contre les abus, en mettant en place des mécanismes de limitation de requêtes et en configurant des politiques de sécurité CORS, afin de prévenir les surcharges système et les accès non autorisés*
• Sécurisation des connexions et du stockage des données sensibles ▪ L'implémentation et gestion des certificats SSL/TLS pour la protection des communications	*Stocker les mots de passe, en respectant les bonnes pratiques de hachage sécurisé, afin de limiter les risques d'exfiltration de données*
▪ Les meilleures pratiques pour le stockage sécurisé des mots de passe (BCrypt, Argon2) ▪ La gestion des clés et secrets avec des outils dédiés (AWS KMS, HashiCorp Vault, Azure Key Vault)	*Gérer les clés et secrets de manière sécurisée, en utilisant des solutions dédiées, afin d'empêcher l'accès non autorisé aux données sensibles*
• Outils et méthodologies d'audit et de pentesting des applications ▪ Audit et tests de sécurité avancés ○ Introduction au pentesting des applications web et mobiles	*Réaliser des tests d'intrusion avancés sur les applications web et mobiles, en utilisant des outils comme Burp Suite, OWASP ZAP et Postman, afin d'identifier et corriger les failles de sécurité*
○ Simulation d'attaques avec Burp Suite, OWASP ZAP et Postman pour identifier les vulnérabilités ○ Détection des failles via Dynamic Application Security Testing (DAST)	*Intégrer des scanners de sécurité, en configurant des analyses statiques et dynamiques dans un pipeline CI/CD, afin d'assurer une détection précoce des vulnérabilités*
▪ Automatisation de la sécurité dans les workflows DevOps (DevSecOps) ○ L'intégration de scanners de sécurité automatisés dans un pipeline CI/CD (GitHub Security, Snyk, SonarQube)	*Exploiter des solutions basées sur l'intelligence artificielle, en utilisant des algorithmes de machine learning et des systèmes de détection comportementale, pour surveiller les anomalies protéger les applications contre les attaques évolutives*

○ L'utilisation de l'IA pour la détection des anomalies et la protection contre les attaques	
2. Conformité et réglementation : Sécurisation des données et RGPD	
• Introduction à la conformité RGPD et aux normes de protection des données ▪ Présentation du Règlement Général sur la Protection des Données (RGPD) ▪ Le champ d'application du RGPD ○ Les règles pour les entreprises européennes et internationales manipulant des données d'utilisateurs de l'UE ○ Les impacts sur les développeurs et concepteurs d'applications	*Analyser le cadre réglementaire du RGPD et ses implications sur le développement d'applications, afin d'assurer la conformité aux exigences légales*
• Intégration de la conformité RGPD dans le développement d'applications ▪ Les principes fondamentaux du RGPD appliqués au développement ○ Privacy by Design & Privacy by Default : Intégration dès la conception des applications ○ Les droits des utilisateurs : consentement, droit à l'oubli, portabilité des données ○ La gestion des logs et la traçabilité des accès aux données sensibles	*Mettre en œuvre les principes du Privacy by Design & Privacy by Default, en intégrant des mesures de protection, pour les appliquer dès la conception des applications* *Gérer les droits des utilisateurs, en mettant en place des interfaces de gestion des demandes de confidentialité, afin de garantir la conformité aux réglementations sur la protection des données et d'assurer la transparence pour les utilisateurs*
▪ La sécurisation des données personnelles ○ La mise en place de mesures de protection des données : chiffrement des données au repos et en transit, pseudonymisation et anonymisation des informations personnelles ○ La gestion des violations de données : les obligations de déclaration aux autorités (CNIL, DPO), et les stratégies de réponse rapide en cas de fuite d'informations	*Appliquer des stratégies de chiffrement des données sensibles, en utilisant des algorithmes de cryptographie avancés, afin de garantir la confidentialité des informations utilisateurs* *Élaborer un plan de gestion des violations de données, en définissant les procédures de notification aux autorités compétentes et aux utilisateurs impactés, afin de minimiser les impacts des incidents de sécurité et d'assurer la conformité aux réglementations en vigueur*
▪ Les outils et technologies pour la mise en conformité RGPD ○ La gestion des consentements utilisateurs : implémentation de bandeaux cookies conformes avec Tarteaucitron.js, Google Tag Manager, contrôle du tracking utilisateur via des solutions RGPD-compliant	*Déployer des solutions de gestion des consentements conformes au RGPD, en utilisant des outils adaptés, afin d'assurer la transparence des traitements de données et de garantir le respect des droits des utilisateurs*
▪ La protection des données dans le cloud ○ Les meilleures pratiques pour les contrats de sous-traitance des données ○ La conformité des services cloud avec AWS, Azure et GCP	*Superviser la conformité des applications cloud, en analysant les contrats de sous-traitance des données*

	avec AWS, Azure et GCP, afin d'identifier et de corriger les éventuels risques liés à la protection des données personnelles
Contenu	**Capacités attendues**
Cloud computing et architectures serverless (40 à 50 h)	
Objectifs pédagogiques globaux : - **Assimiler les principes fondamentaux du cloud computing, ainsi que les modèles de service offerts par les principaux fournisseurs cloud** - **Déployer des applications dans le cloud, en utilisant des machines virtuelles ou des services serverless et en configurant les environnements de production, de test et de préproduction** - **Utiliser des outils de conteneurisation et d'orchestration pour gérer les déploiements d'applications conteneurisées dans le cloud, en assurant leur évolutivité et haute disponibilité** - **Sécuriser les données et les services dans le cloud, en utilisant des stratégies de chiffrement, de gestion des accès et de sauvegarde, tout en assurant la conformité avec les normes de sécurité** - **Optimiser les coûts et la gestion des ressources cloud, grâce à des pratiques de scalabilité automatique et une surveillance continue des performances et des logs**	
1. Présentation et introduction au cloud computing • Définition du cloud computing et présentation des principaux modèles de service : IaaS, PaaS, et SaaS • Les avantages du cloud computing : la scalabilité, la réduction des coûts, la flexibilité et la sécurité • Les principaux fournisseurs de cloud ▪ AWS (Amazon Web Services) et les services clés (EC2, S3, RDS, Lambda, etc.) ▪ Microsoft Azure : les services Azure (App Services, Azure Functions, Blob Storage, etc.) ▪ Google Cloud Platform (GCP) : les services Google Cloud (Compute Engine, Cloud Functions, Cloud Storage, etc.) ▪ L'ensemble des services Alibaba (spécifiques au continent asiatique) • Comparaison entre les différents fournisseurs de cloud ▪ Les types de services, leur utilisation spécifique, les Services serverless, leur tarification, leurs supports d'intégration etc. ▪ Les avantages et les inconvénients des différents fournisseurs de cloud • Le déploiement d'applications dans le cloud ▪ Les instances de calcul : le déploiement d'applications sur des machines virtuelles (ex : EC2, Compute Engine, Azure VM etc.) ▪ Serverless computing : les services sans serveur (ex : AWS Lambda, Azure Functions, Google Cloud Functions etc.) pour déployer des applications réactives et évolutives ▪ La gestion des environnements de déploiement : la configuration des	*Définir le concept de cloud computing, en identifiant les principaux modèles disponibles, afin de comprendre les différentes options de service offertes par le cloud* *Déterminer les avantages du cloud computing, en analysant des critères tels que la scalabilité, la réduction des coûts, la flexibilité et la sécurité, pour démontrer les bénéfices de la migration vers le cloud* *Comparer les principaux fournisseurs de cloud, en étudiant leurs services respectifs, afin de sélectionner la plateforme la plus adaptée aux besoins d'une entreprise* *Analyser les avantages et inconvénients des différents fournisseurs de cloud, en tenant compte de leurs types de services, tarification et supports d'intégration, pour choisir la meilleure solution de déploiement cloud* *Déployer des applications dans le cloud, en utilisant des instances de calcul, pour assurer le fonctionnement des applications sur des machines virtuelles* *Mettre en œuvre des services serverless, en utilisant des services dédiés, afin de déployer des applications réactives et évolutives sans gérer d'infrastructure de serveur*

environnements de test, de préproduction et de production sur le cloud • Conteneurisation et déploiement dans le cloud ▪ La conteneurisation avec Docker et la gestion des conteneurs avec Kubernetes sur les plateformes cloud (EKS, AKS, GKE) ▪ Le déploiement d'applications conteneurisées dans le cloud et la mise en place de clusters Kubernetes • La gestion des coûts sur le cloud ▪ Les principes de tarification du cloud (facturation à l'usage) ▪ L'optimisation des coûts grâce à la scalabilité et aux instances réservées ou spot • Les outils de déploiement dans le cloud ▪ AWS Management Console, Azure Portal et Google Cloud Console pour la gestion des services cloud ▪ Docker pour la conteneurisation des applications ▪ Kubernetes pour la coordination des conteneurs dans le cloud ▪ CloudFormation, Terraform ou Ansible pour l'automatisation du déploiement d'infrastructure cloud	*Gérer des environnements de déploiement cloud, en configurant des environnements de test, de préproduction et de production, afin de garantir la qualité des applications tout au long de leur cycle de vie* *Utiliser Docker et Kubernetes pour conteneuriser et coordonner les applications sur les plateformes cloud, afin d'assurer la scalabilité et la gestion des ressources des applications* *Optimiser les coûts d'infrastructure cloud, en appliquant les principes de tarification à l'usage et en utilisant des instances réservées ou spot, afin de minimiser les dépenses et améliorer la flexibilité et les performances des ressources cloud*
2. Gestion des données sur le cloud : stockage sécurisé, mise à l'échelle, haute disponibilité • Les différents types de stockage dans le cloud ▪ Le stockage évolutif des fichiers : le stockage d'objets avec S3 (AWS), Blob Storage (Azure), et Google Cloud Storage ▪ Les bases de données dans le cloud : Introduction à RDS (AWS), Azure SQL Database, et Cloud SQL (Google) pour les bases de données relationnelles ▪ Présentation des bases de données NoSQL : DynamoDB, CosmosDB, Google Firestore etc. ▪ Le chiffrement des bases de données et stockage d'objets avec AWS KMS, Azure Key Vault et GCP Secret Manager. • La sécurisation des données dans le cloud ▪ Le chiffrement des données au repos et en transit : le chiffrement AES, la gestion des certificats SSL/TLS ▪ La gestion des accès et permissions : les politiques de contrôle d'accès (IAM sur AWS, RBAC sur Azure) pour sécuriser l'accès aux données sensibles	*Différencier les types de stockage dans le cloud, en comparant les solutions de stockage et les bases de données relationnelles et NoSQL, pour choisir la meilleure solution de gestion des données en fonction des besoins* *Utiliser des bases de données cloud, en configurant des plateformes spécifiques, afin de gérer les données relationnelles dans un environnement cloud* *Sécuriser les données dans le cloud, en mettant en œuvre des techniques de chiffrement et en gérant les certificats SSL/TLS, pour protéger les données sensibles au repos et en transit* *Gérer les accès et permissions aux données sensibles, en appliquant les politiques de contrôle, pour sécuriser l'accès aux ressources cloud à l'aide des meilleures pratiques de sécurité*

▪ La mise en place de stratégies Zero Trust pour limiter l'accès aux ressources cloud en fonction des rôles et des contextes • Haute disponibilité et mise à l'échelle ▪ Auto-scaling : les groupes de scalabilité automatique pour répondre à l'évolution de la charge de travail (exemple : AWS Auto Scaling) ▪ La réplication et la sauvegarde des données : les sauvegardes automatiques et la réplication multi-régions pour garantir la haute disponibilité et la résilience des données ▪ La protection contre les attaques DDoS avec AWS Shield, Azure DDoS Protection et Google Cloud Armor. • La gestion des logs et du monitoring dans le cloud ▪ Introduction à CloudWatch (AWS), Azure Monitor, et Google Stackdriver pour le suivi des performances et la gestion des journaux système (logs) ▪ La gestion des alertes et des notifications pour surveiller la santé des services et des applications dans le cloud ▪ L'intégration des logs de sécurité avec SIEM (Security Information and Event Management) pour détecter les anomalies. • Les outils de gestion des données dans le cloud ▪ AWS S3, Azure Blob Storage, et Google Cloud Storage pour le stockage des données ▪ AWS RDS, Azure SQL, et Google Cloud SQL pour la gestion des bases de données relationnelles dans le cloud ▪ IAM (Identity and Access Management) pour gérer les permissions et sécuriser les accès aux ressources cloud ▪ CloudWatch, Azure Monitor, et Google Stackdriver pour le monitoring et la gestion des logs dans le cloud ▪ L'implémentation de WAF (Web Application Firewall) pour la protection des applications web déployées sur le cloud	*Optimiser la disponibilité des données, en utilisant l'auto-scaling, pour garantir la résilience et la haute disponibilité des données en cas de surcharge ou de panne* *Automatiser les sauvegardes et la réplication des données, en appliquant des stratégies de sauvegarde automatique et de réplication multi-régions, pour garantir le maintien et la sécurité des données en cas de sinistre* *Mettre en place des outils de monitoring et de gestion des logs, en employant des outils appropriés, afin de surveiller les performances des services et identifier les anomalies* *Configurer des alertes et notifications, en utilisant des outils de surveillance pour suivre la santé des services cloud, afin de réagir rapidement en cas d'incident ou de défaillance*
Contenu	**Capacités attendues**
Développement IoT et communication entre objets connectés (30 à 40 h)	
Objectifs pédagogiques globaux : - **Explorer les principes fondamentaux de l'Internet des Objets (IoT) ainsi que les composants clés de son architecture, tels que les capteurs, les actionneurs, les réseaux et les plateformes cloud** - **Mettre en œuvre des communications entre dispositifs IoT, en utilisant des protocoles standards et en assurant une connexion efficace via des technologies sans fil adaptées** - **Développer des applications IoT, en programmant des microcontrôleurs et en intégrant des capteurs, actionneurs et plateformes cloud, pour collecter et analyser des données en temps réel**	

- **Assurer la sécurité des dispositifs IoT et des communications, en appliquant des techniques de protection des données, de chiffrement et d'authentification, afin de prévenir les vulnérabilités dans les systèmes connectés**	
1. Introduction à l'IoT • Définition et présentation des concepts de l'Internet des Objets • Les protocoles de communication IoT : MQTT, CoAP, WebSockets, AMQP : comparaison et cas d'usage • L'architecture IoT : les composants principaux : les capteurs, les actionneurs, les réseaux, les plateformes cloud • Le développement d'applications embarquées avec ESP32 et Raspberry Pi pour la collecte et le traitement des données en local • Les exemples d'usage de l'IoT dans différents secteurs professionnels (ex : industrie, santé, agriculture, domotique, etc.)	*Assimiler les concepts de l'Internet des Objets (IoT), en identifiant ses composants principaux, afin de comprendre l'architecture globale des systèmes IoT* *Analyser les usages de l'IoT, en étudiant des exemples d'application dans différents secteurs, pour démontrer l'impact de l'IoT dans des environnements professionnels*
2. Les technologies et protocoles IoT • Les protocoles de communication : MQTT, CoAP, HTTP, WebSockets, etc. ▪ Présentation détaillée de l'utilisation de MQTT pour la communication légère entre capteurs et serveurs ▪ Présentation du protocole CoAP pour les échanges entre objets à faible consommation • Les connexions sans fil pour l'IoT : Wi-Fi, Zigbee, LoRa, Bluetooth Low Energy (BLE) etc. • Implémentation d'une passerelle IoT pour assurer la communication entre les capteurs et une base de données cloud • Les plateformes IoT : AWS IoT, Google IoT Core, Microsoft Azure IoT Hub etc.	*Configurer des connexions sans fil IoT, en utilisant les technologies idoines, afin de garantir une communication fiable et adaptée aux besoins des objets connectés*
3. Le développement d'applications IoT • La programmation de microcontrôleurs pour l'IoT (ex : Arduino, ESP32, Raspberry Pi, MicroPython, etc.) • La sécurisation des microcontrôleurs avec TPM (Trusted Platform Module) et gestion des mises à jour OTA (Over-The-Air) • L'intégration des capteurs et actionneurs avec des plateformes cloud • Utilisation de MQTT pour la communication légère entre capteurs et serveurs • La gestion des flux de données en temps réel avec Kafka, RabbitMQ et AWS IoT Analytics • La collecte et l'analyse des données IoT dans le cloud	*Programmer des microcontrôleurs IoT, en utilisant des plateformes dédiées, afin de concevoir des systèmes IoT réactifs et connectés* *Analyser les données IoT, en utilisant des plateformes cloud, afin d'optimiser les processus et prendre des décisions éclairées appuyées sur les données collectées*
4. La sécurité dans l'IoT	

• Les principaux risques de sécurité dans les systèmes IoT • La protection contre les attaques de type botnet (Mirai), interception de données et spoofing • Les stratégies de protection des appareils IoT et la sécurisation des communications (chiffrement, authentification, etc.) • La sécurisation des échanges IoT avec TLS/SSL et Mutual TLS (mTLS) pour authentifier les objets connectés	*Identifier les principaux risques de sécurité, en analysant les vulnérabilités communes aux appareils et réseaux IoT, afin de comprendre les menaces inhérentes aux objets connectés* *Protéger les appareils IoT, en mettant en place des stratégies de chiffrement et d'authentification, afin de sécuriser les communications entre les objets connectés et les serveurs*
Contenu	**Capacités attendues**
Éthique, réglementation et éco-conception logicielle (20 à 25 h)	
Objectifs pédagogiques globaux : - **Examiner les cadres légaux et réglementaires relatifs à l'informatique, notamment la protection des données, la cybersécurité et les contrats numériques** - **Appliquer les principes de la Responsabilité Sociétale des Entreprises (RSE) dans un contexte informatique, en tenant compte des aspects sociaux et éthiques** - **Promouvoir des pratiques de Green IT en optimisant l'impact environnemental des solutions informatiques**	
1. Introduction au droit informatique • Protection des données personnelles ▪ Les principes du RGPD (Règlement Général sur la Protection des Données) ▪ Les obligations des développeurs concernant la collecte et le traitement des données (chiffrement, gestion des accès etc.) • Les contrats numériques et la propriété intellectuelle ▪ Les droits et obligations des parties dans un contrat logiciel (ex : licence, SaaS, maintenance etc.) ▪ La protection des droits d'auteur et la gestion des licences open source ▪ Analyse d'une licence GNU/GPL et ses implications pour un projet de développement • Cybersécurité et responsabilités juridiques ▪ Les lois générales encadrant la cybersécurité ○ La directive européenne NIS2 (Network and Information Security) sur la sécurité des réseaux et des systèmes d'information ○ Le RGPD (Règlement Général sur la Protection des Données) ▪ Les autres lois et cadres légaux ○ Cybersecurity Act (UE) ○ La loi française sur la cybersécurité (LPM - Loi de Programmation Militaire) ○ Les lois sur la cybercriminalité (Convention de Budapest)	*Identifier les obligations légales liées à la protection des données, en appliquant le RGPD, afin de garantir la conformité d'un projet numérique* *Analyser les clauses des contrats logiciels, en distinguant les obligations et responsabilités des parties, pour protéger les intérêts des utilisateurs et des développeurs* *Évaluer les risques juridiques en cas de négligence en cybersécurité, afin d'assurer la conformité avec les lois en vigueur*

▪ Les responsabilités en cas de cyberattaques ou de négligence en matière de sécurité	
2. La Responsabilité Sociétale des Entreprises (RSE) appliquée à l'informatique • Introduction à la RSE ▪ Les principes clés de la RSE : sociaux, environnementaux, et économiques ▪ L'intégration des principes éthiques dans les processus informatiques (inclusion, diversité etc.) ▪ Exemples de pratiques discriminatoires dans un algorithme	*Intégrer les principes de diversité et inclusion dans les projets informatiques, en identifiant les biais potentiels, pour favoriser une informatique équitable*
• L'impact des systèmes numériques sur les conditions de travail ▪ Le télétravail et l'ergonomie numérique ▪ Le respect des horaires et la prévention des risques psychosociaux liés aux technologies ▪ Les impacts des plateformes numériques sur le bien-être des employés	*Concevoir des systèmes numériques respectueux des droits sociaux et humains, en analysant leur impact sur les conditions de travail, afin de promouvoir le bien-être des utilisateurs et des employés*
• RSE et chaîne d'approvisionnement technologique ▪ La traçabilité des composants électroniques (ex : minerais rares) ▪ Le respect des droits humains dans la chaîne de production informatique.	*Superviser la chaîne d'approvisionnement technologique, en identifiant les enjeux éthiques, pour garantir le respect des normes sociales*
3. Green IT : pour une informatique durable et écoresponsable • Introduction au Green IT ▪ Définition et objectifs du Green IT ▪ Analyse de l'impact environnemental de l'informatique (ex : consommation d'énergie, e-déchets etc.) et de son empreinte carbone	*Promouvoir une démarche Green IT, en sensibilisant les équipes aux pratiques durables, afin de minimiser l'impact environnemental des solutions développées*
• Les pratiques écoresponsables ▪ L'optimisation des ressources informatiques (ex : usage de serveurs partagés, virtualisation etc.) ▪ Le développement logiciel durable (ex : réduction de la consommation énergétique d'un site web, rédaction de codes minimalistes etc.)	*Optimiser l'utilisation des ressources informatiques, en adoptant des pratiques de virtualisation et d'économies d'énergie, afin de réduire l'empreinte écologique des projets numériques*
• Recyclage et gestion des déchets numériques ▪ Les bonnes pratiques pour le recyclage des équipements informatiques ▪ Les initiatives et certifications en matière de durabilité (ex : label TCO Certified) ▪ Exemples de projets Green IT réussi (ex : Google Data Centers qui a intégré l'IA pour gérer sa consommation avec une amélioration de 15% de son efficacité énergétique)	*Mettre en œuvre des solutions de recyclage et de gestion des e-déchets, en suivant les standards environnementaux, pour limiter l'impact des équipements informatiques*

Développement d'applications et solutions connectées

Développement d'applications IA et Machine Learning appliqué (20 à 25 h)	
Objectifs pédagogiques globaux : - **Examiner les principes fondamentaux de l'intelligence artificielle et du machine learning en explorant les différentes approches et en identifiant leurs applications pratiques dans les domaines du développement logiciel** - **Maîtriser la préparation et la manipulation des données pour l'entraînement des modèles d'IA en appliquant des techniques de nettoyage, de transformation et d'optimisation des datasets, tout en assurant la conformité avec les normes de protection des données** - **Concevoir et entraîner des modèles d'apprentissage automatique et de Deep Learning en développant des algorithmes performants, et en optimisant leurs hyperparamètres pour améliorer la précision et la généralisation** - **Intégrer des solutions d'intelligence artificielle dans des applications web et mobiles en exploitant des API IA et en embarquant des modèles optimisés dans des environnements cloud, serverless ou mobiles, afin de proposer des services intelligents et interactifs** - **Industrialiser et automatiser le cycle de vie des modèles d'IA en mettant en œuvre des pratiques MLOps, incluant le déploiement via Docker et Kubernetes, l'intégration continue (CI/CD) et la surveillance des performances en production, pour garantir la scalabilité et la robustesse des solutions IA**	
1. Introduction à l'Intelligence Artificielle et au Machine Learning • Définition et applications courantes de l'IA (vision par ordinateur, NLP, recommandations, etc.) • Distinction entre IA symbolique, Machine Learning et Deep Learning • Présentation des algorithmes de base ▪ Apprentissage supervisé (régression, classification) ▪ Apprentissage non supervisé (clustering, réduction de dimensionnalité) ▪ Apprentissage par renforcement (Q-learning, DQN) • Présentation des outils et Frameworks ▪ TensorFlow, PyTorch, Scikit-learn (Python) ▪ ML.NET, OpenCV, FastAI **2. Traitement et préparation des données pour l'IA** • La manipulation et le nettoyage des datasets ▪ Introduction aux bases de données pour l'IA : SQL, NoSQL, Big Data ▪ La collecte et le nettoyage des données avec Pandas, NumPy, OpenRefine ▪ La gestion des données déséquilibrées : Oversampling, Undersampling, SMOTE ▪ Feature Engineering : extraction et transformation des données • Le stockage et la manipulation des données pour l'IA ▪ Intégration avec Google BigQuery, AWS S3, Azure Data Lake	*Identifier et expliquer les concepts fondamentaux de l'intelligence artificielle et du machine learning en analysant les différentes approches existantes, afin de distinguer leurs applications respectives dans les systèmes modernes* *Préparer et structurer des données pour l'apprentissage automatique, en collectant et nettoyant des datasets à l'aide d'outils dédiés, afin d'optimiser la qualité des données et d'améliorer la performance des modèles*

▪ La structuration des données pour le Machine Learning et le Deep Learning ▪ La sécurité des données IA : anonymisation, conformité RGPD **3. Développement et entraînement de modèles IA** • La création et l'entrainement des modèles IA dans une application ▪ L'implémentation des modèles supervisés avec Scikit-learn (régression linéaire, SVM, arbres de décision) ▪ Introduction aux réseaux de neurones et Deep Learning avec TensorFlow/Keras et PyTorch ▪ Optimisation des hyperparamètres : GridSearch, RandomSearch, AutoML ▪ Les tests et validation des modèles : Split train/test, cross-validation, métriques d'évaluation (Accuracy, F1-score, AUC-ROC) • L'optimisation et l'exportation des modèles ▪ La conversion de modèles pour une exécution rapide : ONNX, TensorFlow Lite, TorchScript ▪ La compression et l'accélération avec Quantization, Pruning, Distillation **4. L'intégration de l'IA dans les applications web et mobiles** • La consommation des modèles IA via API ▪ Le déploiement de modèles IA via Flask / FastAPI pour une consommation backend ▪ L'intégration d'API IA cloud : Google AI, OpenAI GPT, AWS SageMaker, Hugging Face API ▪ La gestion des appels API : REST, GraphQL, WebSockets • Le développement d'applications mobiles et web avec IA embarquée ▪ L'intégration de TensorFlow.js pour le Machine Learning dans le navigateur ▪ L'implémentation de modèles IA sur mobile avec ML Kit (Firebase), CoreML (iOS), TensorFlow Lite (Android) ▪ La détection d'images, la traduction automatique et les recommandations de contenu **5. Le déploiement et l'industrialisation de modèles IA** • Le déploiement d'une IA en production avec le Cloud et les conteneurs ▪ Le déploiement Serverless avec AWS Lambda, Google Cloud Run, Azure Functions	*Développer et entraîner des modèles de Machine Learning et de Deep Learning, en implémentant des algorithmes supervisés et non supervisés, afin de réaliser des prédictions et des classifications sur des jeux de données variés* *Intégrer des modèles d'intelligence artificielle dans des applications web et mobiles, en déployant des solutions d'apprentissage automatique, afin d'optimiser l'expérience utilisateur et les performances applicatives*

▪ La conteneurisation des modèles IA avec Docker et Kubernetes (Kubeflow, TensorFlow Serving) ▪ L'optimisation des modèles pour la scalabilité (batch processing vs streaming data) • Automatisation et MLOps ▪ Introduction au CI/CD appliqué à l'IA avec GitHub Actions et MLflow ▪ Automatisation du monitoring des modèles (drift detection, log monitoring) ▪ Gestion des performances et des coûts avec Inference at Edge et Cloud TPU • Les exemples de création avec l'IA ▪ La création d'un chatbot NLP avec GPT-4 et LangChain ▪ La reconnaissance faciale et la détection d'objets avec OpenCV et YOLO ▪ Les systèmes de recommandation (films, e-commerce etc.) avec Scikit-Learn et TensorFlow ▪ Le déploiement d'un modèle de prédiction sur AWS Lambda • Les outils de développement d'applications IA ▪ Python : TensorFlow, PyTorch, Scikit-Learn, OpenCV, Hugging Face ▪ Cloud : AWS SageMaker, Google Vertex AI, Azure ML ▪ DevOps : Docker, Kubernetes, CI/CD pour IA (GitHub Actions, MLflow) ▪ Développement Web et Mobile : Flask, FastAPI, React avec TensorFlow.js, Firebase ML Kit	*Industrialiser et automatiser le déploiement des modèles d'intelligence artificielle, en mettant en œuvre des pratiques MLOps, le monitoring des modèles et la conteneurisation, afin d'assurer la scalabilité, la sécurité et l'optimisation des performances*
Veille technologique et évolutions des pratiques IT (5 à 6 h)	
Objectifs pédagogiques globaux : - **Explorer les tendances émergentes en développement logiciel et en gestion d'infrastructures afin d'identifier les innovations ayant un impact sur les pratiques du secteur IT** - **Utiliser des outils et méthodologies de veille afin de collecter, trier et analyser des informations pertinentes sur l'évolution des technologies et des méthodologies IT** - **Évaluer les évolutions des pratiques de développement et des architectures logicielles afin d'intégrer des approches adaptées aux exigences du marché et aux innovations technologiques** - **Mettre en œuvre une démarche de veille continue afin de structurer une adaptation proactive aux transformations du secteur IT** - **Restituer et communiquer les résultats d'une veille technologique afin de produire des synthèses exploitables et partager des recommandations stratégiques au sein d'une équipe technique**	
1. Les enjeux de la veille technologique • Définition et importance de la veille technologique dans l'IT • L'impact des nouvelles tendances sur les métiers du développement et de l'architecture logicielle	

• Les évolutions majeures récentes (ex : IA générative, low-code/no-code, microservices, DevSecOps)	*Évaluer les tendances technologiques, en identifiant les innovations émergentes, afin d'anticiper leur impact sur le développement informatique*
2. Les méthodes et outils de veille • Les méthodologies de veille : exploration active vs surveillance passive • Les outils et plateformes pour suivre les tendances ▪ Les agrégateurs de contenu (Feedly, Pocket, Google Alerts) ▪ Les plateformes spécialisées (Hacker News, Dev.to, Medium, GitHub Trends) ▪ Les conférences et publications scientifiques (ACM, IEEE, arXiv) ▪ Les réseaux professionnels et influenceurs IT (Twitter, LinkedIn, podcasts) ▪ L'automatisation de la veille : création d'alertes, filtres avancés, flux RSS	*Sélectionner des sources d'information fiables, en utilisant des outils de veille adaptés, afin d'optimiser la recherche et la collecte de données*
3. Évolutions des bonnes pratiques et méthodologies IT • Les transformations récentes dans les pratiques IT ▪ L'essor du DevSecOps et l'intégration de la sécurité dès la phase de développement ▪ Serverless computing et architectures Cloud : l'évolution du paradigme ▪ Le développement durable et le Green IT : l'impact environnemental et l'optimisation des ressources ▪ La cybersécurité : les nouvelles menaces et les solutions émergentes (Zero Trust, chiffrement post-quantique) • Études des tendances et anticipation des futures évolutions	*Analyser les évolutions des méthodologies IT en comparant les pratiques traditionnelles et récentes afin d'améliorer la gestion des projets et le développement logiciel*
4. La stratégie de veille et sa mise en pratique • L'élaboration d'un plan de veille personnalisé sur une technologie ou un domaine précis • La restitution sous forme d'une revue de tendances IT	*Mettre en œuvre une stratégie de veille personnalisée, en exploitant des plateformes spécialisées, afin de rester à jour sur les évolutions du secteur* *Présenter une synthèse des tendances IT en restituant les informations clés sous un format structuré afin de partager les résultats de la veille avec une équipe technique*

B. Évaluation

Forme de l'épreuve : Épreuve professionnelle écrite (4 h) + étude de cas (2 h)
Durée totale : 6 h

Notation : 120 points

Barème :

- Epreuve écrite résolution de problématiques techniques et théoriques - 60 points
- Etude de cas appliquée - 60 points

Aucun document, support de cours ou outil complémentaire (outil d'intelligence artificielle, calculatrice, dictionnaire, etc.) n'est autorisé durant cette évaluation.

C. Coefficient et ECTS

Ce module vaut coefficient **7** et permet de capitaliser **22 ECTS.**

UC D32

Épreuve Professionnelle de Soutenance

A. Objectifs

La pédagogie doit faire une large place à l'initiative de l'apprenant et à son travail personnel pour mettre en œuvre les connaissances et les compétences acquises. À cette fin, les missions professionnelles effectuées en entreprise impliquent l'élaboration d'un rapport d'activité qui donne lieu à une soutenance orale.

Le Bachelor européen réalise une mise en contact réelle de l'apprenant avec le monde du travail de manière à lui permettre d'approfondir sa formation et son projet professionnel et de faciliter son insertion dans l'emploi.

Une partie de la formation peut être accomplie à l'étranger dans le cadre d'une convention.

B. Évaluation

L'épreuve professionnelle de soutenance permet de valider les capacités de l'apprenant à mener un projet professionnel, à développer une problématique dans un rapport d'activité et à expliquer et défendre sa démarche devant un jury.

En raison de l'intérêt qu'elle représente dans la formation de l'apprenant, cette épreuve est obligatoire.

1. Modalités de préparation

Quel que soit le pays d'exercice, l'élaboration du rapport d'activité peut s'appuyer sur différentes modalités d'expériences formatives :

- Soit un stage en entreprise ;
- Soit un emploi salarié ou un contrat d'alternance ;
- Soit des travaux plus théoriques par le biais d'un projet tutoré.

1.1. Le stage en entreprise

Durée : 12 semaines minimum.
Contenu : Réalisation d'une ou plusieurs actions en rapport avec le développement d'applications et solutions connectées
Capacités attendues : Appréhender les réalités d'une activité liée au développement d'applications et solutions connectées

Le stage doit se dérouler pendant la formation.

La date et la planification de ce stage sont laissées à la libre appréciation de l'établissement de formation, en accord avec sa propre organisation pédagogique.

Par exemple, le stage peut être scindé en 2 parties ou organisé selon un rythme hebdomadaire propre à l'alternance (n jours en école, n jours en entreprise).
Toutefois, il semble préférable, pour des motifs pédagogiques, que le stage ainsi scindé se déroule dans la même entreprise ou organisation.

Le terrain de stage doit être choisi en fonction des possibilités d'actions professionnelles de l'apprenant, et soumis à l'équipe pédagogique de l'école, qui en valide le bien-fondé et l'adéquation avec le diplôme préparé ainsi que le niveau exigé. Il peut s'agir d'une entreprise publique ou privée ou d'une organisation au sens large.

Ce stage donne l'occasion à l'apprenant de déterminer, en relation avec son tuteur en entreprise et, éventuellement, son tuteur-enseignant, les études, les actions ou les missions qui lui seront confiées et qui constitueront la matière de son rapport d'activité.

La production d'un certificat de fin de stage mentionnant la durée, les dates et les missions confiées par l'entreprise sera exigé au moment de l'épreuve de soutenance.

1.2. L'alternance ou l'emploi salarié

La préparation du rapport d'activité peut également s'appuyer sur l'expérience professionnelle de l'apprenant, qu'il soit salarié à temps plein, à temps partiel ou en contrat d'alternance, pourvu que la nature de ses activités professionnelles et le niveau de ses responsabilités soient conformes aux spécificités et aux exigences du présent référentiel et des examens FEDE qui y sont rattachés.

Dans ce cas, ce sont les missions qui sont confiées au salarié qui deviennent la matière de son rapport d'activité.
La production d'un certificat de travail mentionnant la durée, les dates et, éventuellement les études ou missions confiées par l'entreprise, sera exigé au moment de l'épreuve de soutenance.

1.3. Le projet tutoré

En cas de difficulté majeure pour trouver un stage ou un contrat d'alternance en entreprise, l'apprenant a la possibilité de réaliser un projet tutoré en accord avec son centre de formation et la FEDE.

Dans ce cas, le projet de rapport d'activité est négocié et déterminé en début d'année en concertation avec l'équipe pédagogique et plus spécialement un tuteur-enseignant, qui aura pour rôle de superviser le projet et guider l'apprenant.

Toutefois, l'obtention d'un stage ou d'un contrat d'alternance en entreprise doit constituer la priorité.

Durée : ¼ du volume de la formation, hors stage
Contenu : Dans le cadre d'un travail individuel ou collectif, réalisation d'un rapport d'activité retraçant l'ensemble des actions menées pour la réalisation d'une opération en lien avec le développement d'applications et solutions connectées définie en début d'année et validée par le tuteur enseignant.
Capacités attendues : Mettre en œuvre une stratégie permettant la réalisation effective d'une action de développement d'applications et solutions connectées

1.3.1. Contenu du projet

Ce projet peut avoir pour thème, par exemple :
- Le développement d'une application de gestion d'objets connectés domestiques
- Un système de suivi et de gestion de tâches collaboratives
- Une plateforme de suivi de santé connectée
- Le développement d'une application de géolocalisation pour le partage de ressources communautaires.
- Un système de sécurité domestique connecté

Dans la mesure du possible, ce projet aura une dimension européenne et sera élaboré en liaison avec une entreprise ou une organisation professionnelle où il pourrait trouver une application.

1.3.2. Rôle du tuteur

Le tuteur est un des enseignants de l'apprenant. En tant que tuteur, son rôle consiste à :
- Suggérer des idées de projet ou d'étude ;
- Valider le projet et négocier avec l'apprenant l'évolution du projet ;
- Orienter ses recherches bibliographiques et documentaires ;
- Fournir des pistes pour mettre en place des relations avec des entreprises ou des organisations professionnelles ;
- Surveiller la qualité d'ensemble du travail fourni ;
- Participer, le cas échéant au jury d'examen.

2. Le rapport d'activité ou le mémoire

Ce rapport d'activité constitue une partie du travail évalué par le jury. En tant que tel, il est donc un objet d'évaluation et représente **30%** de la note finale.

2.1. Le contenu du rapport d'activité

Le rapport d'activité ne doit pas se résumer à un simple descriptif de l'activité de l'apprenant ou à un simple compte rendu de lecture. Il doit représenter un effort de recherche, d'analyse et d'application concernant un aspect réel et bien délimité de l'activité d'une entreprise (entendue au sens large), dans un contexte économique européen si possible.

L'observation des pratiques de l'entreprise ou de l'organisation et/ou la lecture des ouvrages théoriques en relation avec le sujet doit permettre à l'apprenant de cerner une problématique relative à un contexte précis, et lui donner l'occasion de développer une analyse et des propositions concrètes qu'il doit être capable de justifier.

L'organisation du rapport d'activité est importante, il doit respecter une ordonnance classique, en abordant dans un ordre logique les différentes étapes de l'élaboration du projet, dont voici quelques exemples :

- Introduction ;
- La demande ou la commande ;
- La problématique ;
- L'idée de départ, le projet initial ;
- Les hypothèses de recherche ;
- Les résultats attendus ;
- La méthodologie utilisée ;
- Les arguments du projet, les propositions ;
- L'évaluation, la comparaison avec d'autres projets ;
- La confrontation avec la réalité, le terrain, les entreprises ;
- Les résultats éventuellement obtenus ;
- Les outils de contrôle éventuellement mis en place ;
- Les avantages apportés par le projet ou l'étude.

2.2. Présentation du rapport d'activité

Le rapport d'activité sera saisi au traitement de texte et présentera les caractéristiques suivantes :

- Format A4 ;
- Nombre de pages : environ 30 à 35 pages (hors annexes) ;
- Impression recto seul ;
- Marges 2,5 cm de chaque côté ;
- Interligne 1,5 ;
- Relié.

Le rapport peut contenir quelques annexes essentielles qui ne doivent pas dépasser un volume maximum de 10 feuilles A4.

La provenance de ces annexes doit être clairement indiquée (document élaboré par l'apprenant, tiré de telle publication, fourni par l'entreprise…).

La page de titre doit comporter les mentions suivantes :

- Nom et prénom de l'apprenant ;
- Numéro de candidat attribué par la FEDE ;
- Titre éventuel du rapport ;
- « Examens de la FEDE » ;
- « Rapport d'activité présenté à l'épreuve professionnelle de soutenance du diplôme visé de [année] ».

Il devra contenir un sommaire au début, une bibliographie à la fin et éventuellement une table des annexes.
Il sera exigé la même rigueur que pour les travaux universitaires en ce qui concerne la présentation des références, des citations, etc.
Il faut prévoir une édition en au moins deux exemplaires, un pour le jury, un pour l'apprenant.

2.3. Délai de fourniture du rapport d'activité

Les rapports d'activités doivent être envoyés en deux exemplaires au centre d'examen (pour transmission au jury) au moins 3 semaines avant le début de la période annoncée pour ce type d'épreuve.

3. Déroulement de la soutenance

Le jury est composé d'un enseignant de la spécialité auquel il est adjoint un professionnel. L'épreuve dure 30 minutes. Pas de temps de préparation.
La soutenance orale représente **70%** de la note finale.

3.1. Exposé théorique (de 10 à 15 min)

Dans un premier temps, le jury invitera l'apprenant à justifier le choix de son projet ou de son étude et à livrer les conclusions auxquelles il est parvenu.

Ce travail de soutenance ne doit pas conduire l'apprenant à « lire » son rapport devant le jury. Cette partie de l'épreuve est une évaluation des compétences de communication orale dans un contexte professionnel et technique.

L'apprenant s'efforcera donc de retracer, d'une manière construite et raisonnée, son cheminement dans le choix d'un sujet ou d'un projet, les difficultés qu'il a connues et comment il les a surmontées, la place que ce projet a prise par rapport à son projet professionnel global, l'intérêt qu'il a trouvé, le bénéfice qu'il a tiré d'un travail personnel d'élaboration et de recherche, les contacts qu'il a pu nouer à cette occasion avec des professionnels, des organisations, les suites qui seront éventuellement données…

Il devra savoir introduire et conclure son exposé, et maîtriser son temps de parole.

L'apprenant peut utiliser à sa guise des documents complémentaires qui ne sont pas dans le rapport d'activité remis au jury et qu'il aura apportés.

L'apprenant a aussi la possibilité d'utiliser les techniques de présentation qu'il juge utiles (par exemple : présentation assistée sur ordinateur…) pourvu qu'il soit autonome dans l'utilisation de ces outils et qu'il reste dans le temps imparti.

Pendant cet exposé de 10 à 15 minutes, l'apprenant ne sera pas interrompu.

3.2. Discussion avec le jury (15 à 20 min)

Dans un deuxième temps, le jury reviendra sur des aspects plus techniques ou professionnels, notamment sur le contenu du rapport d'activité, et posera les questions suscitées par la lecture de celui-ci.

Toutefois, s'agissant de la partie « soutenance orale » de l'épreuve, le jury évaluera moins la précision et la justesse des éléments de réponse technique fournis que la capacité de l'apprenant, à maîtriser la situation de communication, à comprendre et à traiter une objection, à organiser un discours, à convaincre…

4. Objectifs et critères d'évaluation

- Grille de notation
- Eléments observables

C. Règles d'utilisation de l'IA générative dans la rédaction du rapport d'activité

L'utilisation de l'IA générative (et des technologies assistées par l'IA) dans le processus de rédaction des rapports est autorisée, sous certaines conditions détaillées ci-dessous.

1. Les règles d'utilisation de l'IA dans la rédaction des travaux rédactionnels

En cas d'utilisation de l'IA dans le processus de rédaction par les candidats, trois règles prévalent et correspondent à trois principes, afin de garantir la possibilité d'une évaluation pertinente de la production personnelle du candidat : le respect des sources, la transparence et la responsabilité.

- Respecter les règles relatives au plagiat, à la copie et à la nécessité de citer les sources utilisées.
- Déclarer dans le manuscrit avoir utiliser l'IA dans le processus rédactionnel et indiquer quelles IA ont été utilisées et pour quel usage.
- Le candidat est ultimement responsable du contenu de son travail et des textes soumis pour évaluation. L'IA ne doit pas être citée comme auteur ou co-auteur.

2. L'IA et le risque de plagiat

Utiliser l'IA pour générer le texte du rapport d'activité est assimilé à du plagiat et peut être sanctionné par une disqualification.

Rappel des règles de base relatives au plagiat :

- L'auteur d'un texte doit citer ses sources sous peine de faire du plagiat et d'être disqualifié.
- La paraphrase sans citation de source est une autre forme de plagiat.
- Si le jury estime qu'il y a plagiat, paraphrase, copie d'un modèle sans apport original ou encore qu'il y a recours à un tiers de substitution, et que cela constitue un obstacle à l'évaluation objective des compétences, savoirs et savoir-faire d'un candidat, le jury est fondé à disqualifier le travail de l'étudiant.

3. L'IA et la transparence

Afin de permettre aux membres du jury d'évaluer de la façon la plus pertinente possible la production personnelle du candidat, celui-ci est tenu de déclarer, si c'est le cas, qu'il a eu recours à l'utilisation de l'intelligence artificielle dans la rédaction de son travail rédactionnel, en précisant : quel(s) outil(s), quel(s) usage(s), quelle(s) finalité(s).

4. L'IA et la responsabilité

L'IA ne doit pas être citée comme auteur ou co-auteur, la qualité d'auteur renvoyant à une personne physique. Le candidat est in fine l'auteur du rapport d'activité. Les candidats sont ainsi ultimement responsables du contenu de leur travail et des textes soumis pour évaluation.

D. Coefficient et crédits ECTS

Ce module vaut coefficient **4** et permet de capitaliser **12 ECTS.**

UC D33

Contrôle continu - projet intégratif de développement d'une application avec IoT intégrée

A. Objectifs

Les évaluations sous la forme du contrôle continu certificatif doivent être réalisées en cours de formation à la fin d'une séquence d'apprentissage et permettre l'évaluation de l'acquisition d'un groupe de compétences.
Ces évaluations nécessitent de s'appuyer sur des modalités provoquant des situations observables faisant appel à la mobilisation de diverses connaissances théoriques, de stratégies, de savoir-être et de savoir-faire. Elles doivent s'approcher de l'action et des situations de travail et traduire une mise en œuvre opérationnelle de plusieurs compétences liées à une activité et un métier en cohérence avec le niveau attendu. L'objectif est de valider au fur et à mesure l'acquisition des compétences avant les évaluations finales de la session d'examens FEDE.
De façon générale, l'UC D33 permet également de prendre en compte l'implication et l'assiduité de chaque apprenant dans la formation.

Le temps de travail requis pour mener à bien le contrôle continu est de 55 heures. Cette activité évaluée doit être encadrée par un formateur, afin d'accompagner les groupes et orienter les apprenants dans leurs réflexions.

B. Evaluation

L'évaluation, sous forme de projet de groupe de 3 ou 4 apprenants, consiste à concevoir, développer, sécuriser et déployer une application web et mobile pour une communauté (club sportif, association culturelle etc.), en intégrant des fonctionnalités IoT pour enrichir l'expérience des utilisateurs. Des capteurs IoT pourront être utilisés pour améliorer la gestion de l'espace, des événements ou de la sécurité au sein de la communauté.
En plus des objectifs initiaux liés aux compétences transversales, l'intégration de l'IoT permettra d'évaluer les étudiants sur leur maitrise des méthodes de connexion d'objets physiques à une application numérique, à la collecte des données et à leur gestion via des services backend.

Spécifications et exigences techniques requises pour la conception de l'application :

1. **Fonctionnalités de l'application**
   - Création du profil utilisateur : inscription, création et modification du profil utilisateur
   - Gestion des événements : création, inscription, désinscription, liste des participants
   - Messagerie et groupes : messagerie entre utilisateurs et groupes thématiques
   - Fonctionnalités IoT : utilisation de dispositifs connectés pour enrichir les interactions

2. **Développement frontend :**
   - Technologies : utiliser React ou Vue.js pour le frontend web et React Native ou Flutter pour le mobile
   - Intégration IoT : ajouter des fonctionnalités permettant de visualiser les données

3. **Développement backend :**
   - API RESTful : gérer la collecte et l'interaction avec les capteurs IoT (Node.js/Express ou PHP/Laravel).
   - Base de données : stocker les données des capteurs IoT (SQL/NoSQL).
   - Intégration avec les dispositifs IoT : développer une API qui reçoit et enregistre les données collectées par les capteurs IoT via HTTP

4. **Fonctionnalités IoT :**
   - ESP32 ou Raspberry Pi pour connecter des capteurs à Internet
   - Communication : utilisation du protocole HTTP pour envoyer des données à l'API
   - Application des données : exploiter les données des capteurs pour enrichir l'application (ex. : notifications spécifiques).

5. **Sécurité :**

- Authentification et sécurisation IoT : sécuriser la communication entre les dispositifs IoT et l'API, implémenter des tokens JWT pour l'authentification des dispositifs
- Conformité RGPD : assurer la conformité des données utilisateurs collectées

6. **Cloud et déploiement**
   - Utilisation de AWS, Azure ou Google Cloud pour héberger l'API, et connecter les dispositifs IoT via un hub cloud
   - Utilisation de AWS S3 pour stocker les données historiques issues des capteurs IoT

7. **Architecture logicielle et méthodologies :**
   - L'architecture backend suit une logique MVC, intégrant des services IoT dans un contexte de micro-services
   - Gestion du projet à l'aide des méthodologies Agiles (ex : Scrum)

8. **Tests :**
   - Tests unitaires et d'intégration pour les fonctionnalités backend, y compris l'intégration IoT
   - Tests des dispositifs IoT pour vérifier l'envoi des données vers l'API et la réaction de l'application en fonction des valeurs reçues

Contenu	Capacités attendues
Analyse des besoins et spécifications (5 h)	
• Identifier les besoins de la communauté cible • Définir les fonctionnalités principales de l'application, comme la gestion des profils utilisateurs, des événements, des messages et des groupes thématiques etc. • Intégrer les spécifications fonctionnelles et techniques du projet.	*Analyser les besoins clients pour concevoir une application répondant aux attentes utilisateurs* *Élaborer des spécifications techniques et fonctionnelles complètes* *Justifier les choix technologiques et les outils en fonction des exigences du projet*
Conception de l'architecture logicielle (5 h)	
• Définir l'architecture logicielle adaptée au projet : architecture MVC pour le backend et structuration en micro-services pour des fonctionnalités indépendantes • Concevoir la structure de la base de données SQL et NoSQL et la logique de communication avec les dispositifs IoT	*Concevoir une architecture logicielle fiable, adaptée à une application Full Stack moderne* *Appliquer des modèles d'architecture comme MVC et les concepts de micro-services* *Créer des schémas de base de données pour une application connectée*
Développement backend de l'application (15 h)	
• Implémenter l'API RESTful permettant de gérer les fonctionnalités backend, l'authentification, la gestion des utilisateurs, des événements, des messages, et l'intégration avec les dispositifs IoT via HTTP	*Créer et sécuriser des API RESTful pour gérer les fonctionnalités backend* *Intégrer des dispositifs IoT au backend, en utilisant les protocoles de communication adaptés* *Automatiser les tests backend pour garantir la robustesse des API et prévenir les régressions*
Développement Frontend (10 h)	
• Créer une interface utilisateur dynamique et intuitive en utilisant React ou Vue.js pour l'application web, et React Native ou Flutter pour l'application mobile	*Concevoir et développer une interface utilisateur attractive et réactive.*

Développement d'applications et solutions connectées

• Mettre en œuvre une interface adaptée aux mobiles, permettant la visualisation des données issues des dispositifs IoT.	*Assurer la compatibilité multi-plateformes (web et mobile) et l'affichage des données en temps réel* *Mettre en place un design system cohérent et documenté pour garantir l'uniformité des interfaces*
Développement et intégration IoT (5 h)	
• Connecter des dispositifs pour collecter des données et les intégrer au backend de l'application • Utiliser HTTP pour la communication et s'assurer que les données sont enregistrées dans les bases appropriées (SQL/NoSQL)	*Connecter des dispositifs IoT et intégrer les données dans une application Full Stack* *Configurer la communication des dispositifs via les protocoles adéquats (ex : HTTP)*
Déploiement Cloud et gestion des bases de données (5 h)	
• Déployer l'application sur une plateforme Cloud et configurer des services comme les machines virtuelles, le stockage d'objets, la base de données etc. • Assurer la scalabilité et la disponibilité des services	*Déployer des applications dans un environnement Cloud en utilisant des services adaptés* *Configurer et gérer les bases de données cloud pour assurer leur sécurité et disponibilité*
Sécurité et conformité RGPD (3 h)	
• Implémenter des mesures de sécurité, telles que la protection contre XSS, CSRF, la sécurisation des APIs avec JWT etc. • Garantir la conformité RGPD des données utilisateurs en assurant la sécurité et la confidentialité des informations	*Appliquer les mesures de sécurité adaptées aux applications web et mobile* *Intégrer la conformité RGPD dans le traitement des données utilisateurs*
Analyse des enjeux éthiques, RSE et Green IT (3 h)	
• Identification des enjeux éthiques, sociaux et environnementaux dans le développement d'applications • Intégrer des pratiques responsables : inclusion, accessibilité, optimisation énergétique (Green IT) • Analyse de l'impact sociétal et environnemental des choix techniques et propositions de solutions durables	*Analyser les impacts sociétaux et environnementaux d'un projet pour anticiper les défis éthiques et durables* *Implémenter des pratiques écoresponsables et inclusives dans le développement d'applications* *Évaluer la responsabilité numérique d'un projet et proposer des solutions d'amélioration*
Tests et assurance qualité (5 h)	
• Réaliser des tests unitaires et d'intégration pour l'application web et mobile. • Tester la communication entre les dispositifs IoT et le backend, ainsi que la fiabilité des données collectées	*Mettre en œuvre des tests pour valider la qualité de l'application* *Assurer la cohérence et la fiabilité des données collectées par les dispositifs IoT* *Intégrer des outils de tests automatisés dans un pipeline CI/CD pour garantir la stabilité de l'application*
Gestion de projet avec Scrum (2 h)	
• Organiser le travail autour du projet en utilisant les méthodologies Agile (ex : Scrum) : gestion du backlog, suivie de la progression des tâches etc.	*Appliquer la méthodologie Scrum pour participer aux diverses étapes d'un projet logiciel*

	Utiliser les outils de gestion agile pour collaborer efficacement au sein d'une équipe

C. Livrables

1. Le code source

1.1. Le fond

- Structuration selon une architecture adaptée (MVC ou microservices)
- Documentation complète du code (commentaires explicites, normes de nommage et conventions de style)
- Gestion de versions via Git, avec un historique de commits clair et détaillé (incluant l'usage des branches et merge requests)
- Sécurisation des données et des accès (validation des entrées utilisateur, chiffrement des mots de passe, protection contre les attaques XSS, CSRF, injections SQL).
- Implémentation des fichiers de configuration pour CI/CD (GitHub Actions, GitLab CI/CD, Jenkins) et déploiement automatisé
- Documentation détaillée des outils DevOps utilisés (orchestration Kubernetes, Docker, Terraform)
- Intégration de tests automatisés (tests unitaires, tests d'intégration, tests de sécurité avec SAST et DAST)

1.2. La forme

- Le code source devra être hébergé sur GitHub ou GitLab, avec un README clair, comprenant :
- Des instructions pour installer, configurer et exécuter le projet
- La liste des dépendances et Frameworks utilisés
- Un diagramme simplifié de l'architecture technique (UML ou équivalent)
- La documentation des pipelines CI/CD et des workflows DevOps
- La présentation des scripts d'automatisation, des configurations Docker/Kubernetes, et des logs d'exécution.
- Le rapport d'historique Git montrant les contributions des membres de l'équipe, PRs (Pull Requests) et merge requests

2. La documentation technique

2.1. Le fond

La documentation technique devra inclure :

- Une description détaillée de l'architecture logicielle avec des diagrammes UML (cas d'utilisation, diagramme de classe, séquence).
- Les explications des choix technologiques basés sur la veille technologique (ex : GraphQL vs REST, Docker vs Kubernetes, choix du framework frontend)
- Les détails sur l'intégration IoT : protocole utilisé (MQTT, HTTP, WebSockets), configuration des capteurs/actionneurs, gestion des flux de données
- Les spécifications des API RESTful et GraphQL (endpoints, méthodes HTTP, exemples de requêtes/réponses)

La validation de la conformité DevOps devra inclure :

- Les Pipelines CI/CD automatisés et capture des résultats de tests.
- Les Logs des tests de performance et de sécurité
- Les captures des workflows de déploiement Cloud

2.2. La forme

La documentation devra être claire et bien structurée avec :

- Une table des matières interactive
- Une introduction expliquant les objectifs du projet et sa finalité
- Des sections distinctes pour chaque composante (Frontend, Backend, IoT, API, DevOps)
- Des captures d'écran des workflows CI/CD, logs de déploiement et résultats des tests
  - Une présentation des outils collaboratifs utilisés (Trello, Jira, GitHub Projects).

Document au format PDF (15 à 20 pages) avec annexes, schémas et configurations complexes

3. Le rapport de sécurité

3.1. Le fond

Ce rapport devra démontrer que le projet respecte les standards de sécurité informatique, avec :

- La liste des mesures de sécurité implémentées :
  - Chiffrement des données sensibles (AES-256, RSA)
  - Authentification sécurisée avec JWT, OAuth 2.0, SSO
  - Protection contre XSS, CSRF, injections SQL, DDoS
  - Sécurisation des API avec CORS, API Gateway
- L'analyse des risques liés aux données IoT (protection des communications entre capteurs et backend)
- La conformité DevOps : sécurisation des pipelines CI/CD, gestion des secrets avec HashiCorp Vault
- La mise en conformité RGPD :
  - Gestion des consentements et droits des utilisateurs.
  - Stratégie de pseudonymisation et anonymisation des données

3.2. La forme

La rapport de sécurité devra être structuré comme suit :

- Une introduction expliquant les enjeux de sécurité et de conformité
- Une matrice des risques avec les solutions mises en place
- Des graphiques et schémas illustrant les mécanismes de sécurité
- Une conclusion et des recommandations pour améliorer la sécurité et la conformité

Nombre de pages recommandé : **10 à 15 pages (format PDF)**

4. Le rapport d'utilisation (manuel utilisateur)

4.1. Le fond

- Le guide d'installation pour les utilisateurs finaux
- Les instructions détaillées pour naviguer dans l'application (Frontend et fonctionnalités mobiles)
- Les explications des fonctionnalités IoT : connexion des dispositifs, visualisation des données
- Une FAQ pour résoudre les problèmes courants

4.2. La forme

- Manuel illustré avec des captures d'écran et des étapes numérotées
- Document au format PDF ou web interactif

Nombre de pages recommandé : **10 à 15 pages (au format PDF ou digital)**

5. Rapport d'analyse Éthique, RSE et Green IT

5.1. Le fond

- L'identification des enjeux éthiques, sociaux et environnementaux
- Les stratégies mises en place pour améliorer l'impact du projet :
  - Accessibilité numérique (compatibilité avec lecteurs d'écran, couleurs adaptées aux daltoniens)
  - Optimisation énergétique (Green IT, réduction de la consommation serveur)
  - Impact sociétal et potentiel d'inclusion
- Le calcul de l'empreinte énergétique du projet (simulation ou estimation des économies réalisées grâce à l'optimisation du code et du cloud)

5.2. La forme

Le rapport d'analyse éthique, RSE et Green IT devra présenter la structure suivante :

- Une introduction décrivant les enjeux éthiques, RSE, et environnementaux spécifiques au projet
- Une analyse des défis et des opportunités
- Les solutions mises en place et la justification des choix pris
- Une conclusion avec des recommandations pour une amélioration continue
- De annexes composées de graphiques ou tableaux

Nombre de pages recommandé : **5 à 10 pages (format PDF)**

6. Le dossier de présentation finale

6.1. Le fond

- Présentation des objectifs et solutions du projet.

- Explication de l'impact sociétal et environnemental.
- Démonstration des principales fonctionnalités.
- Justification des choix technologiques en fonction de la veille IT (GraphQL vs REST, Docker vs Kubernetes, IA embarquée).

6.2. La forme

- Projection d'un diaporama (ex : présentation PowerPoint, Google Slides, Canva etc.) comprenant des slides visuellement clairs (texte, schémas et images)
- Une version interactive (ex : simulation sur mobile)

Durée totale 30 min (15 à 20 minutes de présentation + 10 minutes de questions et d'échange avec le jury)

D. Grilles de notation

Dans le cadre de ces évaluations en contrôle continu, et pour chaque candidat, voici le dossier candidat et les grilles de notations à utiliser obligatoirement : Dossier candidat / D33 / Contrôle continu / développement d'applications et solutions connectées

E. Coefficient et ECTS

Ce module vaut coefficient **2** et permet de capitaliser **8 ECTS.**

UE B | Langue Vivante Européenne

UC B31

Langue Vivante Européenne 1
Utilisateur indépendant – Niveau B1 du CECR

Le référentiel de cette unité d'enseignement est commun pour toutes les langues vivantes, qu'il s'agisse d'une langue vivante 1 (UC B31), langue vivante 2 (UC B32) ou langue vivante 3 (UC B33).

Les apprenants ont la possibilité de choisir parmi les langues vivantes suivantes :
- **Langue vivante 1** : Allemand, Anglais, Espagnol, Français, Italien, Portugais ;
- **Langues vivantes 2 et 3 (facultatives)** : Allemand, Anglais, Arabe, Chinois, Espagnol, Français, Italien, Portugais

La langue vivante choisie par l'apprenant doit être différente de celle dans laquelle il passe les épreuves de la Culture et citoyenneté européennes et du domaine professionnel.

A. Objectif

Atteindre le niveau B1 du Cadre européen commun de référence pour les langues (CECR) en compréhension écrite et orale, production écrite et orale, interaction et médiation.

B. Formation

Le volume horaire recommandé de formation en face à face pédagogique est de 60 à 80 heures.

Utilisateur Indépendant, Niveau B1 du **Cadre Européen Commun de Référence du Conseil de l'Europe**

Écouter	Je peux comprendre les points essentiels quand un langage clair et standard est utilisé et s'il s'agit de sujets familiers concernant le travail, l'école, les loisirs… Je peux comprendre l'essentiel de nombreuses émissions de radio ou de télévision sur l'actualité ou sur des sujets qui m'intéressent à titre personnel ou professionnel si l'on parle d'une façon relativement lente et distincte.
Lire	Je peux comprendre des textes rédigés essentiellement dans une langue courante ou relative à mon travail. Je peux comprendre la description d'événements, l'expression de sentiments et de souhaits dans des lettres personnelles.
Prendre part à une conversation	Je peux faire face à la majorité des situations que l'on peut rencontrer au cours d'un voyage dans une région où la langue est parlée. Je peux prendre part sans préparation à une conversation sur des sujets familiers ou d'intérêt personnel ou qui concernent la vie quotidienne (par exemple famille, loisirs, travail, voyage et actualité).
S'exprimer oralement en continu	Je peux m'exprimer de manière simple afin de raconter des expériences et des événements, mes rêves, mes espoirs ou mes buts. Je peux brièvement donner les raisons et explications de mes opinions ou projets. Je peux raconter une histoire ou l'intrigue d'un livre ou d'un film et exprimer mes réactions.
Écrire	Je peux écrire un texte simple et cohérent sur des sujets familiers ou qui m'intéressent personnellement. Je peux écrire des lettres personnelles pour décrire expériences et impressions.

Médiation	Je peux transmettre les points essentiels de contenus clairs sur des sujets familiers ou d'intérêt personnel. Je peux relayer, dans une langue simple, les informations principales d'un document ou d'une discussion.

C. Ressources pédagogiques mises à la disposition des apprenants par la FEDE

Afin de permettre aux apprenants de s'entraîner sur ce module, la FEDE met à la disposition des écoles et des apprenants des annales d'évaluation (sujets et corrigés).

D. Évaluation

UC B31.1 – Langue Vivante Européenne 1 (Epreuve écrite)

Nota : Aucun document, support de cours ni outil complémentaire (outil d'intelligence artificielle, calculatrice, dictionnaire, etc.) n'est autorisé durant cette évaluation.

Durée : 1 heure

1. Compréhension écrite

Étude de deux textes de 250 à 300 mots chacun, portant sur des thèmes listés par la FEDE. Pour chaque texte, quatre questions sont posées à l'apprenant sous la forme d'un questionnaire à choix multiple.

Parmi les quatre réponses proposées pour chaque question, une seule est correcte.

Objectifs : évaluer la compréhension générale et la capacité à repérer des informations spécifiques ainsi qu'à saisir la logique d'ensemble d'un texte écrit.
Barème : 3 points pour une bonne réponse, 0 pour non-réponse ou réponse erronée.

2. Compétences lexicales et grammaticales en contexte

Questionnaire à choix multiple portant sur l'usage du lexique et de la grammaire en contexte, à partir des textes de compréhension et/ou des thèmes du programme.
L'apprenant répond à seize questions ; parmi les quatre réponses proposées pour chaque question, une seule est correcte.
Objectifs : évaluer la connaissance d'un vocabulaire varié et des structures grammaticales de base, ainsi que la capacité à comprendre le sens d'un énoncé et à sélectionner l'élément linguistique approprié en fonction du contexte.
Barème : 3 points pour une bonne réponse, 0 pour non-réponse ou réponse erronée.

3. Production écrite

Rédaction d'un courrier (lettre, fax, mail ou mémo) dans la langue vivante choisie par l'apprenant à partir d'un canevas fourni dans cette même langue étrangère, éventuellement en réaction à un document fourni dans l'énoncé (publicité, offre d'emploi, courrier).

Nombre de mots : 150 à 200.

Objectifs : évaluer la capacité à rédiger un texte simple et cohérent en fonction d'un contexte donné, à demander ou fournir des informations, à décrire des expériences, à exprimer des sentiments ou impressions et à présenter brièvement une opinion ou une explication.

La présentation ne fera pas l'objet d'une notation spécifique, mais peut être prise en compte pour affiner l'évaluation.

La production écrite est évaluée à partir des critères FEDE correspondant au niveau B1 du Cadre européen commun de référence pour les langues (CECR).

Barème :

Compréhension écrite (8 questions x 3) : 24 points
Compétences lexicales et grammaticales en contexte (16 questions × 3) : 48 points
Production écrite : 28 points
Total : **100 points**

UC B31.2 – Langue Vivante Européenne 1 (Epreuve orale)

L'épreuve orale vise à évaluer la capacité de l'apprenant à comprendre, produire, interagir et assurer une médiation simple dans des situations de communication relevant du niveau B1 du CECR.

Nota : Aucun document, support de cours ni outil complémentaire (outil d'intelligence artificielle, calculatrice, dictionnaire, etc.) n'est autorisé durant cette évaluation.

Durée totale de l'épreuve orale : 40 minutes (préparation comprise).

1. **Présentation et commentaire à partir d'un document iconographique (préparation : 10 min ; passation : environ 10 min)**

Objectif : évaluer la capacité de l'apprenant à présenter et commenter un document visuel de manière organisée, à en dégager les informations essentielles, à exprimer un point de vue simple et à interagir avec l'examinateur.

- **Préparation (10 min)**

L'apprenant tire au sort un document iconographique parmi un choix de 6 à 12 documents et prépare une présentation et un commentaire en réaction à ce document.
Le document iconographique est une photographie, un dessin, un graphique ou un montage de plusieurs de ces éléments portant sur les thèmes du référentiel et des sujets d'actualité s'y rapportant.
L'apprenant peut prendre des notes, mais uniquement comme support d'oral ; toute lecture mot à mot sera pénalisée.
L'apprenant est amené à identifier le message principal du document et à en proposer une interprétation simple.

- **Présentation et commentaire par l'apprenant du document iconographique (4-5 min)**

L'examinateur laisse à l'apprenant le temps de s'exprimer afin d'évaluer la structuration du discours et la clarté de l'expression.
Présentation et commentaire par l'apprenant du document iconographique.
L'examinateur doit laisser à l'apprenant le temps de s'exprimer seul afin de juger de la logique du discours.

- **Discussion sur le document (4-5 min)**

Entretien entre l'examinateur et l'apprenant sur le document, permettant à ce dernier d'expliquer ses idées, de justifier brièvement son point de vue et d'illustrer ses propos.

2. **Compréhension et médiation orales (durée indicative : environ 12 minutes)**

L'examinateur lit à l'apprenant un texte clair et structuré d'environ 250 à 300 mots, portant sur les thèmes du référentiel et des sujets d'actualité s'y rapportant. L'apprenant répond à huit questions de compréhension orale.
Le texte est lu une première fois intégralement afin que l'apprenant en saisisse le sens général. Il est ensuite relu par segments ; après chaque segment, l'examinateur pose la question correspondante. Si nécessaire, le passage concerné peut être relu une fois.
La prise de notes est autorisée ; elle sert de soutien à la compréhension et à la formulation des réponses, et non à la mémorisation stricte.
Cet exercice permet d'évaluer la capacité à comprendre les informations principales et secondaires d'un message oral et à en restituer le sens de manière simple.

3. Entretien

Entretien portant sur le parcours de formation, la spécialité professionnelle de l'apprenant (expérience acquise ou en cours, projet tutoré, spécialisation présente et future), ainsi que sur des aspects de sa vie quotidienne ou de ses centres d'intérêt.
L'examinateur pose des questions et propose des relances permettant à l'apprenant de donner des informations, de décrire des expériences, d'exprimer et de justifier brièvement ses opinions.

Liste des thèmes de l'épreuve de Langue Vivante

Pour le niveau B1, les supports proposés sont adaptés aux capacités des apprenants et mobilisent un lexique courant et spécifique simple, en lien avec des situations de la vie quotidienne, académique ou professionnelle. Ils peuvent faire appel à des contenus comportant une dimension descriptive ou explicative et inviter à exprimer un point de vue ou à justifier brièvement une opinion, sans nécessiter de traitement conceptuel complexe.

- L'Europe et la citoyenneté
  - La citoyenneté européenne et mondiale ;
  - Les droits humains et les valeurs démocratiques ;
  - Les enjeux et débats politiques, sociaux et économiques actuels en Europe ;
  - La participation civique et l'engagement citoyen.

- Le monde du travail
  - Relations humaines au travail ;
  - Organisation du travail : temps, formation, mobilité, conflits, équilibre vie privée/vie professionnelle ;
  - Évolutions sociales et professionnelles en Europe ;
  - Orientation, éducation et apprentissage tout au long de la vie.

- Économie et société
  - Les enjeux économiques du quotidien (consommation, emploi, innovation) ;
  - La mondialisation et ses impacts sociaux, culturels et environnementaux ;
  - Développement durable et responsabilité sociétale.

- Vie professionnelle
  - Communication professionnelle écrite (lettres, courriels, messages) ;
  - Offres d'emploi et candidatures ;
  - Communication pratique : téléphone, visioconférence, outils numériques etc.

- Communication
  - Publicité, médias et communication publique ;
  - Nouveaux outils technologiques et réseaux sociaux ;
  - Médias et information : esprit critique, fake news, influence des médias ;
  - Citoyenneté numérique et responsabilité en ligne.

- Arts, culture et patrimoine
  - Histoire, civilisations et sociétés ;
  - Diversité culturelle et dialogue interculturel ;
  - Expressions culturelles et artistiques.

- Sujets d'actualité
  - Enjeux européens et internationaux ;
  - Société et citoyenneté : environnement, santé, inclusion, égalité ;
  - Coopération et gestion des conflits dans la société.

L'évaluation est réalisée à partir d'une grille critériée conforme aux descripteurs du niveau B2 du Cadre Européen Commun de Référence pour les Langues (CECR).

Elle porte sur les capacités suivantes :

- compréhension de messages clairs portant sur des sujets familiers ou liés au domaine d'études ou professionnel ;
- capacité à s'exprimer et à interagir à l'oral de manière simple mais organisée, en développant des réponses et en exprimant un point de vue ;
- médiation orale simple (reformulation ou transmission des informations principales d'un message ou d'un document) ;
- correction linguistique globale permettant une communication efficace malgré certaines erreurs
- prononciation, aisance et intelligibilité du discours ;
- organisation et cohérence du discours.

E. Grilles d'évaluation

Le détail des critères et des niveaux de maîtrise figure dans les grilles officielles d'évaluation :
UC B3.1- Grille de notation - Niveau B1 du CECR
UC B3.2- Grille de notation - Niveau B1 du CECR

F. Coefficient et ECTS

L'épreuve écrite UC B31.1 vaut coefficient 2 et permet de capitaliser 6 ECTS.

L'épreuve orale UC B31.2 vaut coefficient 2 et permet de capitaliser 6 ECTS.

UE A | Culture et Citoyenneté Européennes

UC A2

Le projet européen : culture et démocratie pour une citoyenneté en action

A. Objectifs

- Analyser l'étude de l'histoire et sa relation avec la culture, la démocratie et la citoyenneté ;
- Comprendre le modèle européen et ses particularités, aux plans historique et culturel ;
- Développer une compréhension critique de la politique, du droit et des droits humains ;
- Développer une connaissance et une compréhension critique de la culture, des cultures, et des religions ;
- Acquérir des connaissances précises sur les institutions européennes et leur fonctionnement ;
- Comprendre le modèle européen d'un point de vue réglementaire et juridique ;
- Mobiliser des outils de compréhension de l'espace européen et des actualités européennes ;
- Acquérir des compétences liées à la culture de la démocratie et au dialogue multiculturel.

B. Formation

Le volume horaire recommandé de formation en face à face pédagogique est de 20 à 25 heures.

Contenu	Capacités attendues
Module d'introduction : Importance de l'histoire	OBSERVATOIRE DE L'ENSEIGNEMENT DE L'HISTOIRE EN EUROPE — **OHTE Conseil de l'Europe**
• La discipline de l'histoire et les concepts de la pensée historique : o Causes et conséquences o Continuité et changement o Pertinence historique o Perspective historique o Sources o La dimension éthique • La relation entre l'étude de l'histoire et la démocratie contemporaine • La pertinence de la pensée et de la compréhension historiques dans le présent et l'avenir	*Souligner les aspects clés, les objectifs et les défis de la discipline de l'histoire* *Décrire les six concepts de la pensée historique* *Utiliser les concepts de la pensée historique pour analyser des événements historiques, des périodes et/ou des individus* *Décrire les liens entre l'étude de l'histoire et la démocratie contemporaine* *Reconnaitre la pertinence de la pensée historique et de la compréhension de l'histoire pour le présent et l'avenir*
Chapitre 1 : L'Europe Actuelle	
• La naissance de l'unité européenne et les prémices de la construction européenne • Les organisations (Conseil de l'Europe, UE, OSCE, AELE, EEE, OECD) • Une Europe multiple (espaces européens, adhésion/appartenance) • Le Conseil de l'Europe	*Analyser le processus de la construction européenne et le processus d'intégration* *Décrire le processus d'adhésion* *Reconnaitre la multiplicité des organisations sur le continent européen et leur géographie variable* *Expliquer le rôle des valeurs communes de la démocratie, de la primauté du droit et du respect des droits de*

○ L'origine et les valeurs du Conseil de l'Europe ○ La structure et les membres ○ L'histoire : faits marquants ○ Champs d'intervention ○ OING	*l'homme au sein du Conseil de l'Europe en tant qu'organisation* *Décrire la structure, le fonctionnement et les champs d'action du Conseil de l'Europe*
Chapitre 2 : L'Europe et le monde	
• Élimination des frontières intérieures • Gestion des frontières extérieures • Espaces transfrontaliers • Frontières et crises • La politique étrangère commune et la politique de sécurité et défense de l'Union européenne ○ Le rôle de l'OTAN • L'Union européenne et ses frontières ○ La Politique Etrangère et de Sécurité Commune ○ La Politique Européenne de Voisinage • L'Union européenne et le monde : ○ Focus sur l'Afrique • Enjeux de la politique commerciale • Les organisations internationales : ○ L'ONU et ses institutions (OMS, Unesco, OIT, FAO) ○ Organisations à caractère économique (OMC, FMI, Banque mondiale) ○ Cours internationales (CIJ, Cour Pénale internationale) ○ Alliances de défense (OTAN)	*Identifier les différents types de frontières définissant l'espace européen* *Décrire la coopération transfrontalière et ses espaces* *Expliquer le contexte géopolitique européen* *Mener une réflexion critique sur les évènements actuels concernant la politique étrangère de l'Union européenne* *Analyser les relations entre l'Union européenne et le monde* *Décrire l'ordre international et le multilatéralisme* *Décrire la multiplicité des organisations internationales* *Identifier des exemples des champs d'intervention des organisations internationales*
Chapitre 3 : Cultures et diversité en Europe	
• La diversité des religions en Europe • La liberté de pensée, de conscience et de religion en Europe : limites et enjeux • Minorités et lutte contre les discriminations en Europe • Les mouvements de migration récents et les principaux instruments d'accueil des migrants	*Définir les critères de discrimination et reconnaître les faits de discrimination* *Décrire les réglementations introduites par les institutions européennes en ce qui concerne les minorités et la discrimination et expliquer comment elles sont appliquées* *Mettre à profit le dialogue interculturel pour faciliter la reconnaissance de différentes identités et appartenances culturelles*

Développement d'applications et solutions connectées

Chapitre 4 : La citoyenneté européenne	
• La Charte des droits fondamentaux de l'Union européenne • La libre circulation et les droits liés à la citoyenneté européenne • Démocratie et participation citoyenne dans l'UE • Conférence sur l'avenir de l'Europe et suites institutionnelles	*Identifier les concepts politiques et juridiques de base de la citoyenneté* *Mener une réflexion critique sur les droits humains en tant que cadre de valeurs européennes* *Expliquer les différentes façons dont les citoyens peuvent influencer les politiques*
Chapitre 5 : Le fonctionnement de l'Union européenne	
• L'Union européenne, ses institutions et leur fonctionnement ○ L'origine de l'UE ○ L'histoire : faits marquants ○ La structure institutionnelle ○ La procédure législative ordinaire ○ Les compétences et les actes de l'UE ○ Gouvernance multiniveau ○ Le contrôle par les juges (UE et juges nationaux) • Les actions de l'Union européenne ○ Les compétences exclusives (douanes, Euro) ○ Les compétences partagées (PAC, Politiques régionales, énergie, climat, transports) ○ Les compétences d'appui (éducation, santé, protection civile)	*Identifier les grands principes de fonctionnement des institutions européennes* *Décrire les modalités de la prise de décision dans l'UE* *Identifier la participation des institutions et organes de l'UE dans la vie des citoyens européens* *Analyser le champ d'action de l'Union Européenne* *Montrer la répartition des compétences entre l'UE et ses États membres* *Appréhender les règles fondamentales de mise œuvre des politiques européennes et leur impact sur la vie des citoyens européens*
Chapitre 6 : Enjeux, défis et avenir de la construction européenne	
• La souveraineté européenne • La transition écologique • La Boussole stratégique • Les débats sur l'Europe (euroscepticisme, genre, inclusion, participation) • L'avenir de l'UE dans le monde (Ukraine, Russie, Chine, US)	*Mener une réflexion critique sur l'actualité européenne depuis différentes perspectives en prenant en compte les facteurs historiques qui ont façonné le monde contemporain* *Mesurer les enjeux et défis européens et mondiaux (immigration, populisme, pandémie, guerres...) pour mieux cerner les desseins à venir de la construction européenne* *Réfléchir de manière critique aux différentes possibilités pour l'avenir de l'Europe*
Chapitre 7 : Focus sur la corruption	GRECO **Groupe d'Etats contre la corruption Conseil de l'Europe**

• Définir la corruption • Les différentes formes de corruption • Cartographier et mesurer la corruption • Les causes de la corruption • Endiguer la corruption • Les standards internationaux de lutte contre la corruption	*Appréhender les conséquences de la corruption* *Définir les termes spécifiques du vocabulaire de la lutte contre la corruption* *Distinguer les moyens de lutte contre la corruption, en aval et en amont* *Appréhender les enjeux de la dimension internationale de la lutte contre la corruption*

C. Ressources pédagogiques mises à la disposition des apprenants par la FEDE

La FEDE met à la disposition des écoles et des apprenants :

- Des fiches thématiques comportant des outils d'auto-évaluation pour se préparer à l'épreuve
- Des annales d'évaluation (sujets et corrigés).

D. Évaluation

Forme de l'épreuve : Questionnaire à Choix Multiples (QCM) en ligne
Durée : 40 minutes
Nombre de questions : 40 questions
Nombre de propositions : 2 à 4 propositions de réponses par question. Une seule proposition est exacte.
Total de points : 120

Le barème de notation est le suivant :

- **+ 3 points par bonne réponse**
- **0 point par réponse erronée**
- **0 point par non-réponse**

Nota : Aucun document, support de cours ou outil complémentaire (outil d'intelligence artificielle, calculatrice, dictionnaire, etc.) n'est autorisé durant cette évaluation.

NB : Formation en présentiel : heures d'enseignement réparties selon l'organisation propre à chaque établissement, à la spécialité du diplôme préparé et à la zone géographique du lieu de formation.

E. Coefficient et ECTS

Ce module vaut coefficient 1 et permet de capitaliser 3 ECTS.

UC A3

Le management interculturel et les ressources humaines

A. Objectifs

- Valoriser la diversité culturelle et s'appuyer sur le dialogue interculturel afin de développer une culture du « vivre ensemble » ;
- Développer l'altérité culturelle et la capacité d'interagir et travailler avec des personnes ayant des valeurs, des habitudes, des comportements et des références culturelles différents des siens ;
- S'approprier certains codes culturels afin de comprendre leurs impacts dans les relations interpersonnelles ;
- Mener une réflexion critique sur les différentes conventions de communication appliquées dans un autre groupe social ou une autre culture ;
- Mesurer l'impact de la culture dans la gestion des ressources humaines et participer à l'adaptation et mise en œuvre des pratiques de management interculturel et gestion des ressources humaines ;
- Accompagner et favoriser la mobilité des professionnels afin de leur permettre d'évoluer dans un contexte international.

B. Formation

Le volume horaire recommandé de formation en face à face pédagogique est de 20 à 25 heures.

Contenu	Capacités attendues
Partie 1 : Le management interculturel en Europe	
Chapitre 1 : Culture et diversité culturelle	
• Compréhensions de la culture ○ Bases ethnique, linguistique et religieuse ○ Fondements économiques, technologiques et politiques ○ Culture et style de vie ○ Éducation et culture • Langue et culture ○ La diversité linguistique ○ Plurilinguisme et multiculturalisme • Le multiperspectivisme ○ Définition et enjeux du multiperspectivisime ○ Approches pour reconnaître et prendre en compte les différentes perspectives, expériences et valeurs des individus dans une organisation	*Comprendre que la diversité culturelle au sein d'une société doit être perçue positivement et valorisée* *Analyser l'origine des cultures pour comprendre leur impact sur les valeurs fondamentales de l'individu et leur relation avec le comportement au travail* *Comprendre que le dialogue interculturel doit être mis à profit pour faciliter la reconnaissance de nos différentes identités et appartenances culturelles* *Distinguer l'approche multiperspectiviste dans le contexte de la gestion interculturelle au sein d'une organisation*
Chapitre 2 : La communication interculturelle dans une organisation	
• Valeurs et dimensions culturelles • La culture organisationnelle ○ La construction des valeurs de l'entreprise ○ La communication interculturelle • Ethnorelativisme	*Reconnaitre les enjeux liés à la construction des valeurs de l'entreprise et leur impact sur les individus et l'organisation* *Définir différents styles de communication et faciliter la communication entre des personnes d'origines culturelles différentes*

• Conseils et outils pour une communication interculturelle efficace	*Mener une réflexion critique sur les effets des différents styles d'utilisation de la langue dans des situations sociales et professionnelles*
Chapitre 3 : Gérer l'interculturel et résoudre des conflits culturels	
• Comment gérer et travailler dans une équipe multiculturelle o Diversité ethnique o Diversité linguistique o Les discriminations • Culture générationnelle o Les différences générationnelles o Création et gestion d'équipes transgénérationnelles • Culture européenne o L'identité européenne : identité historique collective et identités locales • La résolution des conflits culturels o Méthodes et cas particuliers	*Distinguer les modes de pensée et la langue des générations pour réunir des équipes multigénérationnelles autour d'un projet commun* *Reconnaître la diversité des influences au sein d'une équipe et percevoir les effets de la diversité culturelle* *Analyser les filtres culturels et les intégrer dans la gestion des équipes* *Communiquer et appréhender les différences culturelles en Europe pour construire dans la diversité une pensée convergente* *Instaurer régulièrement la communication pour contribuer à résoudre des conflits interpersonnels* *Acquérir des outils pour prévenir et gérer les conflits interculturels de manière efficace*
Partie II : Les ressources humaines en Europe	
Chapitre 4 : Travailler en Europe	
• L'Union européenne et la stratégie européenne pour l'emploi o La Stratégie décennale de l'UE en faveur des droits des personnes en situation de handicap • L'Union européenne et le droit du travail o La durée du travail en Europe o Les contrats de travail o Les salaires en Europe o Disparités salariales o Charges sociales et coût du travail o Santé et sécurité au travail o Protection contre les discriminations • L'immigration professionnelle et les espaces européens o L'Espace économique européen o L'Union européenne o L'espace Schengen o Intégration des migrants originaires de pays non-membres de l'UE •Mobilité professionnelle vers et en Europe : les différents statuts	*Identifier les institutions européennes impliquées dans l'élaboration du droit européen du travail* *Connaître les sources principales du droit européen du travail* *Distinguer les relations en droit national et droit européen du travail* *Reconnaitre les principaux objectifs et orientations de l'Union européenne en matière de droit du travail* *Distinguer les textes du Conseil de l'Europe et de l'Union européenne en matière de droit du travail* *Mener une veille réglementaire* *Définir la libre circulation des travailleurs et des citoyens de l'UE et expliquer ses implications* *Distinguer les conditions d'immigration d'un citoyen européen de celles d'un ressortissant non européen*

○ Types de contrats de travail ○ Les principaux éléments d'un contrat de travail • Recrutement en Europe ○ Types de recrutement et contraintes juridiques	*Différencier les différents statuts de la mobilité professionnelle* *Expliquer l'action de l'UE en faveur de la mobilité professionnelle* *Choisir et utiliser les outils et les plateformes européennes de recrutement*
Chapitre 5 : Les systèmes de protection sociale en Europe	
Introduction : L'influence des conventions internationales à portée universelle sur la protection sociale en Europe • Le Conseil de l'Europe et la mobilité des travailleurs ○ Charte sociale européenne (1961) ○ Code européen de sécurité sociale (1964) ○ Les conventions bilatérales et multilatérales de sécurité sociale entre États • La sécurité sociale en Europe : ○ Carte européenne d'assurance maladie ○ Les tendances actuelles en matière de sécurité sociale en Europe ○ Nouveaux défis ○ Variables d'ajustement	*Illustrer les enjeux culturels de la protection sociale en Europe* *Distinguer et respecter les règles de la protection sociale applicables au sein des différents pays d'Europe pour faciliter la mobilité professionnelle et protéger les salariés* *Identifier les possibilités de liaison entre régimes des différents pays de l'UE pour renforcer l'ouverture à l'internationale de l'entreprise* *Comprendre le fonctionnement général des différentes branches de protection sociale dans les pays de l'UE* *Mettre en place un système de veille des évolutions en matière de règles de la protection sociale en Europe*
Chapitre 6 : La responsabilité sociétale des entreprises (RSE)	
• LA RSE : définition et enjeux ○ Des origines au concept : entre volontarisme et obligation ○ Définition de la RSE : entre enjeux sociaux et sociétaux • Le périmètre de la RSE ○ Lignes directrices ○ La notion de parties prenantes et de sphère d'influence ○ Le principe de transparence et son corollaire ○ La RSE et les institutions européennes • La question sociale au cœur de l'entreprise : les textes de références de la RSE ○ RSE et RH, prise en compte des risques psychosociaux et de la Qualité de vie au travail (QVT-QVCT) ○ Respect des Droits humains et lutte contre les discriminations : Devoir de vigilance et prise en compte de la chaîne d'approvisionnement	*Expliquer les bases conceptuelles et courants de pensée à l'origine de la RSE et appréhender l'écosystème de la RSE* *Distinguer les opportunités et les risques d'une démarche RSE* *Décrire les enjeux liés à la RSE en Europe et les envisager comme des leviers d'action pour renforcer l'entreprise* *Illustrer de quelle manière la RSE impacte la gestion des ressources humaines dans un contexte européen* *Participer au développement d'une politique RH et RSE en entreprise* *Analyser des enjeux liés à la diversité et à l'inclusion en milieu professionnel et les enjeux des politiques de gestion de la diversité*

• Politiques d'inclusion ○ Enjeux des politiques de gestion de la diversité ○ Enjeux et des défis liés à l'égalité hommes-femmes (diversité et performance) ○ Le harcèlement en entreprise ○ Les politiques LGBTQIA+ • Politiques d'inclusion des personnes en situation de handicap	*Participer au développement des solutions pour favoriser l'inclusion des personnes en situation de handicap*

C. Ressources pédagogiques mises à la disposition des apprenants par la FEDE

La FEDE met à la disposition des écoles et des apprenants :

- Des fiches thématiques comportant des outils d'auto-évaluation pour se préparer à l'épreuve
- Des annales d'évaluation (sujets et corrigés).

D. Évaluation

Forme de l'épreuve : Questionnaire à Choix Multiples (QCM) en ligne
Durée : 40 minutes
Nombre de questions : 40 questions
Nombre de propositions : 2 à 4 propositions de réponses par question. Une seule proposition est exacte.
Total de points : 120

Le barème de notation est le suivant :

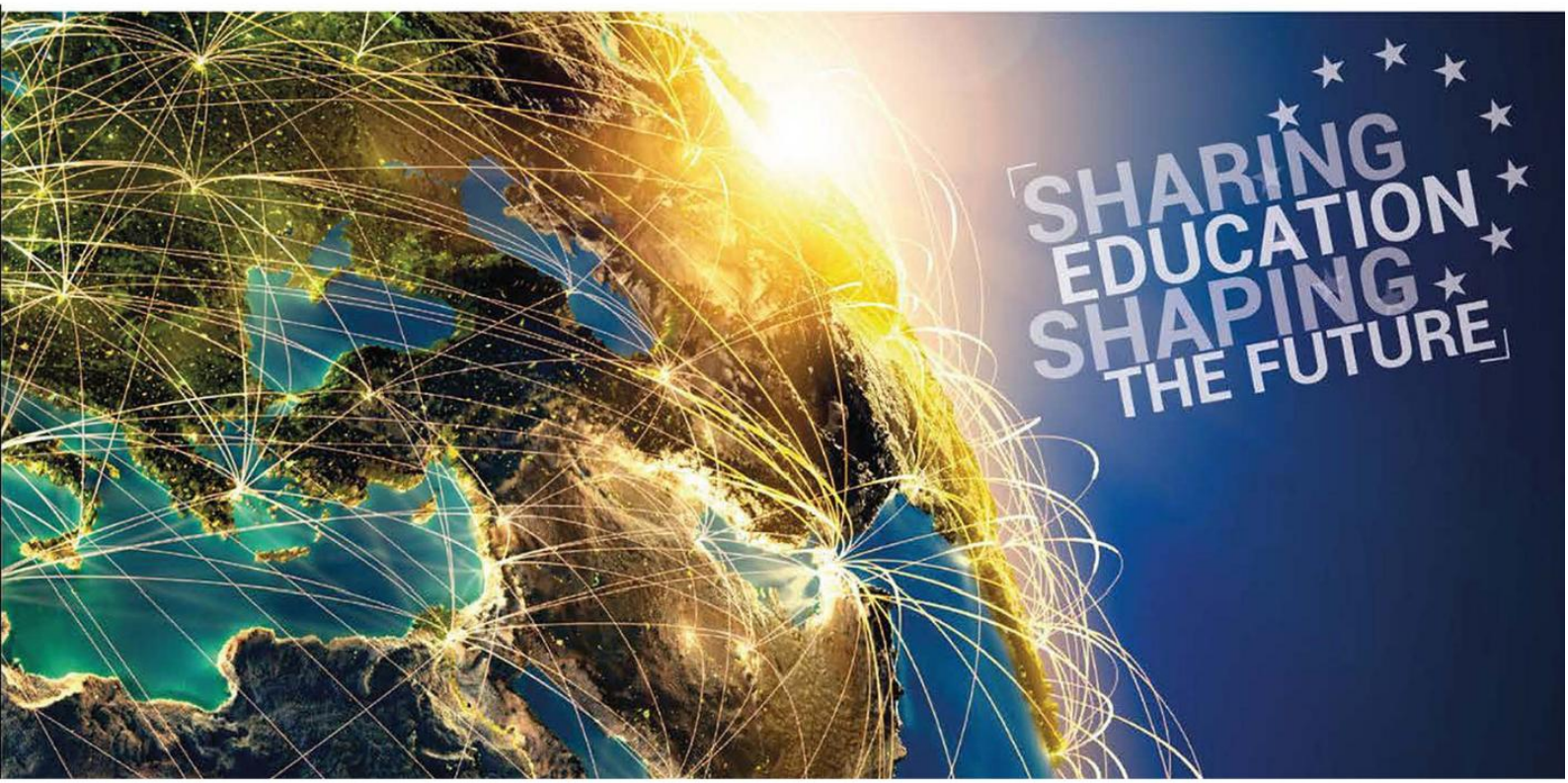
- **+ 3 points par bonne réponse**
- **0 point par réponse erronée**
- **0 point par non-réponse**

Nota : Aucun document, support de cours ou outil complémentaire (outil d'intelligence artificielle, calculatrice, dictionnaire, etc.) n'est autorisé durant cette évaluation.

NB : Formation en présentiel : heures d'enseignement réparties selon l'organisation propre à chaque établissement, à la spécialité du diplôme préparé et à la zone géographique du lieu de formation.

E. Coefficient et ECTS

Ce module vaut coefficient 2 et permet de capitaliser 3 ECTS.



FEDEration for European Education
FÉDÉration Européenne des Ecoles

INGO holding participatory status with the Council of Europe
OING dotée du statut participatif du Conseil de l'Europe
INGO holding consultative status with la Francophonie
OING dotée du statut consultatif auprès de la Francophonie
INGO holding the status of official partner of UNESCO and of ECOSOC
OING dotée du statut de partenaire officiel de l'UNESCO et du CESNU

FEDE - La Voie Creuse 16 - 1202 - Genève - SUISSE
www.fede.education - mailbox@fede.org