

# Projet Semestriel 5A TL - S1 - Bloc 2

## Intervenants du module

Matière	Intervenant
Framework NextJS / TypeScript	Karl MARQUES BERNARDO
Serveless Cloud (Firebase)	Karl MARQUES BERNARDO
Frontend mobile natif (Swift / Kotlin)	CAMARA Hamadi

## Présentation du projet

### Contexte

La mobilité et la logistique instantanée ont transformé nos usages quotidiens : commander un chauffeur, se faire livrer un repas, réserver un service en quelques clics... Ces modèles, popularisés par des plateformes comme **Uber**, ont redéfini l'expérience utilisateur dans de nombreux secteurs.

Aujourd'hui, les mêmes logiques peuvent être appliquées bien au-delà du transport de personnes. Il s'agit de mettre en relation instantanément une **demande** (un client) et une **offre** (un prestataire), tout en garantissant fiabilité, rapidité et sécurité.

### Objectif pédagogique

Les étudiants doivent concevoir et développer une **plateforme numérique de type "Uber"**, en mobilisant leurs compétences techniques avancées (NextJS, TypeScript, Firebase, mobile natif, cybersécurité).

L'enjeu est de construire un écosystème complet comprenant :

- **Une interface utilisateur fluide et intuitive** (application mobile à minima).
- **Un système temps réel** de mise en relation entre clients et prestataires.
- **Une logique métier adaptable** selon le secteur choisi (livraison, services, soins, loisirs, etc.).
- **Une sécurisation forte** des échanges et des données (cybersécurité, authentification, protection des flux).

## Liberté créative

Le sujet est volontairement **ouvert** :

- Chaque groupe doit respecter le **modèle Uber-like** (matching instantané entre clients et prestataires, suivi en temps réel, validation des étapes).
- Mais chaque groupe choisit **son domaine d'application** :
  - Livraison de fleurs,
  - Transport sanitaire (ambulances),
  - Location d'équipements,
  - Services à domicile,
  - ... ou toute autre idée créative et cohérente.

L'important est de démontrer la **compréhension du modèle** et la **capacité à l'adapter** à un secteur particulier, soyez créatif et amusez-vous au passage.

## Enjeux techniques et professionnels

Le projet semestriel doit permettre aux étudiants de démontrer leur capacité à **concevoir et développer une solution logicielle complète**, en respectant les exigences du **Bloc 2 du titre RNCP 38822 – Expert en Architecture et Développement Logiciel**.

Les enjeux sont les suivants :

1. **Conception de l'architecture logicielle**
  - Élaborer et schématiser une architecture claire et robuste (NextJS, TypeScript, Firebase, Swift/Kotlin).
  - Prendre en compte les contraintes techniques, de sécurité, d'accessibilité et de performance.
  - Formaliser cette architecture via des outils adaptés (UML, BPMN, schémas techniques).
2. **Développement front-end et mobile**
  - Créer des interfaces utilisateurs fluides, inclusives et ergonomiques.
  - Intégrer les meilleures pratiques et frameworks modernes (NextJS, mobile natif).
  - Assurer une expérience accessible (UX/UI inclusive, responsive).
3. **Développement back-end et intégration cloud**
  - Concevoir une logique serveur robuste, sécurisée et performante.
  - Mettre en place des API efficaces et sécurisées, interfaçables avec le front et le mobile.
  - Exploiter un environnement serverless (Firebase) pour gérer données, authentification et notifications en temps réel.
4. **Cybersécurité et qualité logicielle**
  - Intégrer la sécurité dès le développement (authentification, RGPD, OWASP).

- Mettre en place une démarche qualité (contrôle de version, tests automatisés, CI/CD).
- Détecter et corriger les vulnérabilités (pentesting, audit de sécurité applicative).
- 5. **Gestion des données et scalabilité**
  - Concevoir un système capable de traiter et stocker des données en temps réel.
  - Garantir la fiabilité, l'intégrité et la protection des données sensibles.
  - Anticiper la montée en charge et l'optimisation des performances.

## Fonctionnalités principales (notées /15 pts)

1. **Gestion des utilisateurs (clients & prestataires)**
  - Création de compte, authentification sécurisée (JWT, OAuth ou équivalent).
  - Profils distincts : **client** (demandeur) et **prestataire** (offreur du service).
  - Gestion des informations personnelles avec respect RGPD.
2. **Création et gestion des demandes**
  - Le client peut créer une demande (ex. commander une course, une livraison, un service).
  - La demande contient des informations essentielles (type de service, lieu de départ/arrivée, horaire, contraintes spécifiques).
  - Suivi de l'état de la demande (en attente, acceptée, en cours, terminée).
3. **Mise en relation et attribution des services**
  - Algorithme de mise en relation automatique entre clients et prestataires disponibles (ex. géolocalisation, proximité, disponibilité).
  - Possibilité pour le prestataire d'accepter ou refuser une demande.
  - Attribution confirmée en temps réel.
4. **Suivi en temps réel**
  - Géolocalisation des prestataires et visualisation de la progression de la demande.
  - Mise à jour dynamique du statut de la prestation (ex. "en route", "service en cours", "terminé").
  - Notifications en temps réel (Firebase, push notifications, etc.).
5. **Validation et clôture du service**
  - Validation des étapes clés (départ, arrivée, livraison, fin de mission).
  - Confirmation finale de la prestation côté client.
  - Archivage de l'historique des demandes (côté client et prestataire).
6. **Système d'évaluation et feedback**
  - Après la prestation, le client peut noter et laisser un commentaire sur le prestataire.

- Le prestataire peut également évaluer le client (optionnel mais recommandé).
  - Les évaluations alimentent un score global visible dans les profils.
7. **Dashboard & historiques par rôle**
- Client : historique complet des demandes (statut, trajets, factures/justifs, évaluations).
  - Prestataire : missions passées/en cours, revenus/bruts estimés, taux d'acceptation, qualité de service, disponibilité planning.
  - Transverse : pagination, tri, filtres combinés, recherche, traces horodatées, règles d'accès par rôle, conformité RGPD, etc.

## Fonctionnalités supplémentaires (notées /5 pts)

- **Paiement en ligne intégré** (Stripe, PayPal, Apple Pay, Google Pay).
- **Optimisation intelligente des trajets** (IA / algorithme d'optimisation type "TSP" ou heuristique).
- **Module de chat temps réel** (client ↔ prestataire) avec notifications.
- **Système avancé d'administration** :
  - Gestion des litiges, remboursements, fraudes.
  - Suivi KPI (CA généré, taux de satisfaction, temps moyen de réponse).
  - Tableau de bord analytique avec graphiques.
- **Gestion multi-rôles étendue** : ajout d'un rôle "opérateur" ou "superviseur" pour modérer et assister les utilisateurs.
- **Accessibilité renforcée**
- **Intégration IoT / capteurs**
- ... (soumis à validation)

*Liste non exhaustive, Chacune de ces fonctionnalités permet d'aller plus loin sur un plan technique et pro, et offre au jury un bon levier de différenciation entre les groupes.*

## Contraintes techniques et pédagogiques

### Framework NextJS / TypeScript

- **TypeScript strict** : "strict": true, aucune erreur any implicite dans le CI.
- **App Router** : utilisation d'app/" avec layouts, route segments et server components quand pertinent.
- **Formulaires** : validation côté client **ET** côté serveur
- **Gestion d'état** : use server vs use client justifiés ; (pas de librairie lourde sans justification)
- **Performance** : Lighthouse  $\geq 92$  sur Perf.
- **Sécurité front** : XSS, CSRF, ...
- **Au moins 1 élément générique** (ex: gestion de tableaux)

### Serveless Cloud (Firebase)

- **Auth** : Email/password + 1 provider
- **Firestore** : Règles de sécurité (rôle + ownership)
- **Cloud Storage** : Règles de sécurité (taille / accessibilité)
- **Cloud Messaging**
- **Crashlytics** : Stats + Alertes

### Frontend mobile natif (Swift / Kotlin)

- **Langages** : Kotlin (Android) & Swift (iOS)
- **IDE** : Android Studio & Xcode
- **Architecture** : MVVM
- **Normes UI** : Material Design (Android) / Human Interface Guidelines (iOS)
- **Outils** : GitHub pour la gestion de version, Gradle et Swift Package
- **Manager** pour les dépendances
- **Documentation** : KDoc / SwiftDoc
- **Linters** : Detekt / SwiftLint
- Bonne utilisation des capteurs (GPS, accéléromètre, orientation, permissions)
- Qualité et clarté du code (architecture MVVM, modularité)
- Ergonomie et expérience utilisateur homogène
- Robustesse (gestion des erreurs, permissions refusées, réseau indisponible)
- Collaboration efficace via GitHub (branches)

#### Frameworks et outils recommandés

Fonctionnalité	Android (Kotlin)	iOS (Swift)
UI	XML	Storyboard / SwiftUI
Réseau/API	Ktor	URLSession
Base de données locale	Room	CoreData
Auth / Cloud	Firebase Auth	Firebase Auth
Géolocalisation	Google Maps SDK	CoreLocation
Capteurs	SensorManager	CoreMotion
Tests	JUnit	XCTest

## Rendus attendus

### Framework NextJS / TypeScript

- Code source
- Procédure de lancement

### Serveless Cloud (Firebase)

- Code source
- Dump des règles de sécurité (firestore/cloud storage)
- Invitation au projet

### Frontend mobile natif (Swift / Kotlin)

- **Cahier des charges + maquettes** : Définition des fonctionnalités, parcours utilisateur.
- **Prototype UI** : Navigation entre écrans, base de l'architecture MVVM.
- **Intégration capteurs et API** : GPS, affichage sur carte, authentification.
- **Fonctionnalités complètes** : Annonces, géolocalisation, profil utilisateur.
- **Tests et documentation** : Tests unitaires + guide technique.
- **Démo Android & iOS** : Présentation des deux versions.

## Commun

- Repository GitHub (commits signés, README complet, répartition des tâches)

- **Documentation technique (Cahier des charges)**
- **Mise en production**
- Archive finale (code, doc, livrables)

## Notation académique

L'évaluation académique des apprenants repose sur deux niveaux complémentaires :

### 1. Notation du module (contrôles continus)

L'objectif est de mesurer la progression et l'implication de l'étudiant **pendant le semestre** à travers un ou plusieurs contrôles continus (en accord avec le syllabus de cours).

Pondération : **40 % de la note finale académique.**

### 2. Notation du partiel du module

L'objectif est d'évaluer la capacité de l'étudiant à **réaliser un projet complet**, dans un contexte professionnel simulé, et à défendre son travail en soutenance ou via un rendu professionnel.

Pondération : **60 % de la note finale.**

## Notation et soutenance RNCP

La soutenance de projet semestriel se déroule en deux temps distincts et complémentaires :

### 1. Présentation collective – 10 minutes (non notée)

L'objectif est de présenter **une seule fois** le projet final au jury, pour éviter la redondance et donner une vision claire et synthétique.

- **Contenu attendu :**
  - Contexte du projet et problématique adressée.
  - Démonstration du produit / rendu final.
  - Mise en avant de la cohérence globale du projet.
- **Organisation :**
  - Le groupe choisit librement qui présente (tous les membres ne sont pas obligés de prendre la parole).
  - La présentation doit être fluide, qualitative et concise.

- Aucun jury n'évalue cette partie : elle sert uniquement à donner un cadre commun avant l'évaluation individuelle.

## 2. Évaluation individuelle – 20 minutes par candidat (notée, bloc RNCP)

L'objectif est de vérifier les compétences individuelles de chaque étudiant au regard du bloc RNCP associé au projet.

- **Format** : échange direct entre le candidat et le jury externe.
- **Contenu** :
  - **Contribution personnelle** : explication claire de ce qui a été réalisé individuellement.
  - **Questions techniques et méthodologiques** : approfondissement sur les choix, la mise en œuvre, les outils et les méthodes utilisés.
  - **Capacité de recul professionnel** : aptitude à justifier ses choix, proposer des alternatives, identifier les limites et perspectives.
- **Méthode d'évaluation** :
  - Chaque critère du **bloc de compétences RNCP** évalué est vérifié par le jury.
  - Pour chaque critère → **"Acquis"** ou **"Non acquis"**.
  - **Règle stricte RNCP** : un seul "non acquis" entraîne l'invalidation du bloc de compétences

## 3. Rôle et composition du jury externe

- **Exclusivement composé de professionnels du secteur.**
- **Indépendant de l'équipe pédagogique** de .decode, pour garantir :
  - La neutralité de l'évaluation,
  - La conformité aux règles de certification,
  - La valeur officielle de la validation RNCP.
- **Mission** : évaluer les étudiants au regard du référentiel officiel et statuer sur l'acquisition ou non du bloc de compétences.

## Bloc de compétences évalué

Chaque projet semestriel est adossé à un bloc de compétences du titre RNCP préparé dans le cadre du cursus .decode.

## Qu'est-ce qu'un bloc de compétences ?

- Un bloc de compétences est **une partie autonome d'un titre RNCP** (Répertoire National des Certifications Professionnelles).

- Chaque bloc correspond à **un ensemble homogène et cohérent de compétences professionnelles**.
- Ces blocs sont **indépendants** : un étudiant peut valider un bloc sans avoir encore validé les autres.
- Pour obtenir le titre complet, il faut valider **tous les blocs** du référentiel.

## Comment se fait l'évaluation d'un bloc ?

- L'évaluation est réalisée **en situation professionnelle reconstituée**, ici, à travers le **projet semestriel**.
- Chaque critère du bloc est vérifié par le **jury externe**.
- Pour chaque critère :
  - **Acquis** : compétence validée.
  - **Non acquis** : compétence non validée.

### Règle stricte RNCP :

- Tous les critères d'un bloc doivent être validés pour obtenir le bloc.
- Un seul "non acquis" sur un critère entraîne l'invalidation du bloc dans son ensemble.

## Grille d'évaluation

L'évaluation du projet semestriel repose sur la **grille officielle du bloc de compétences RNCP concerné**.

Référentiel d'activités	Référentiel de compétences	Référentiel d'évaluation	
		Modalités d'évaluation	Critères d'évaluation
Bloc 2. Concevoir et développer des solutions logicielles			
A2 Construction de la solution technique du projet de développement logiciel			
A2.1 Conception de la schématisation de l'architecture de la solution logicielle	C2.1 <b>Élaborer l'architecture de l'application logicielle</b> , selon les contraintes techniques, les spécifications et les fonctionnalités attendues, les composants de l'interface déterminés, en respectant les exigences de sécurité et les normes réglementaires, en utilisant un langage de modélisation (ex. BPMN, UML), des outils numériques (ex. draw.io) et une méthodologie adaptée (ex. analyse PESTEL), afin de proposer un prototype de la solution informatique qui répond aux scénarios d'utilisation retenus.	C2.1 à C2.5 – Mise en situation professionnelle reconstituée, portant sur la conception et le développement d'une solution informatique innovante (ex. logiciel, application web ou mobile, solution visant l'Internet des objets (IoT), machine learning, etc.).  L'évaluation prendra la forme d'un dossier écrit et d'une présentation devant un jury de composé de 2 professionnels minimum.  Le candidat présente son projet pendant 20 mn devant le jury , suivi d'un entretien avec le jury de 20 mn sur le projet .	Ce2.1.1 La schématisation de l'architecture de la solution logicielle répond aux spécifications techniques et fonctionnelles, ainsi qu'aux contraintes identifiées Ce2.1.2 L'architecture de l'application logicielle répond aux cas d'utilisation identifiés et aux contraintes réglementaires et de sécurité Ce2.1.3 Les langages de modélisations (ex. BPMN, UML) et les outils utilisés (ex. Draw.io) permettent de conceptualiser et visualiser clairement l'architecture de l'application Ce2.1.4 L'architecture proposée est conformes aux exigences de sécurité et aux normes réglementaires Ce2.1.5 La solution d'architecture logicielle proposée répond aux divers scénarios d'utilisation, et elle est adaptée aux personnes en situation de handicap (ex. compatibilité avec les lecteurs d'écran et les sous-titres) assurant ainsi une réponse adaptée et inclusive aux besoins de tous les utilisateurs. Ce2.1.6 L'architecture proposée permet de réduire l'impact environnemental de la solution (ex. pas de redondance inutile, utilisation de microservices, de la virtualisation et de la conteneurisation pour réduire l'utilisation des ressources, gestion dynamique des ressources, caching etc.)
A2.2 Préparation du développement continu de logiciels	C2.2 <b>Préparer l'intégrité du code</b> , en définissant des indicateurs de référence dans une démarche de qualité, en établissant des listes de contrôle détaillées des livrables, en respectant la conformité réglementaire (normes RGPD), en s'appuyant sur des outils automatisés pour garantir le contrôle de version		Ce2.2.1 Les processus et les outils à mettre en place pour intégrer la sécurité du code dès le début du cycle de développement (shift-left) sont clairement spécifiés Ce2.2.2 Les indicateurs de référence définis permettent de garantir continuellement la qualité du code

	du code (ex. GitHub), en assurant le contrôle de la performance, afin de réduire le cycle de développement, de veiller au développement en toute sécurité et de garantir un produit final sans faille de sécurité dans le code		Ce2.2.3 Les listes de contrôle permettent d'assurer la complétude, la conformité, l'intégrité et la qualité du code tout au long du processus de développement Ce2.2.4 Les exigences attendues sur la sécurité dans le développement et la confidentialité du code source, ainsi que le respect réglementaire des normes (ITIL, RGPD) sont intégrés dans le cadre du projet tout au long du processus de développement Ce2.2.5 Les outils et les fonctionnalités collaboratives de versionning (ex. GitHub) sont clairement précisés et permettent de gérer les versions et les mises à jour et d'éviter les conflits Ce2.2.6 Les bonnes pratiques de codage établies permettent l'identification précoce des vulnérabilités, la réduction du temps de mise en production et une gestion des performances adéquate
A2.3 Développement front-end (côté client)	C2.3 <b>Piloter le développement front-end d'une application ou d'un site web</b> , en gérant les contenus graphiques, interactifs et dynamiques, en intégrant les dernières évolutions des outils et techniques de développement, avec des langages spécifiques (ex. HTML5, CSS3, JavaScript), des bibliothèques de fonctions, des frameworks (ex. ReactJS, AngularJS, Vue.js), en collaboration avec d'autres professionnels (ex. web-designer, UX/UI designer), afin de proposer une navigation fluide et une interface ergonomique et inclusive, accessible à tous les utilisateurs, y compris en situation de handicap		Ce2.3.1 Les outils et les solutions de développement utilisés pour le front-end s'appuient sur la veille technologique des meilleures pratiques et standards actuels Ce2.3.2 Les éléments graphiques, dynamiques et le design responsive de l'interface utilisateur intègrent avec succès les contributions d'autres professionnels (ex. web-designer, UX/UI designer) pour enrichir le projet et améliorer la qualité du produit final Ce2.3.3 L'application intègre l'ensemble des dernières évolutions des outils et techniques de développement front-end, en utilisant des langages spécifiques tels que HTML5, CSS3, JavaScript, et en mettant à jour et appliquant les meilleures pratiques et standards actuels dans le développement front-end Ce2.3.4 L'utilisation des bibliothèques de fonctions et de frameworks (ex. ReactJS, AngularJS, VueJS) répond aux besoins spécifiques du projet et améliore la fonctionnalité et l'interactivité des applications

			<p>Ce2.3.5 L'interface propose une navigation fluide, ergonomique, intuitive et inclusive, et elle est accessible à tous, y compris aux personnes en situation de handicap (ex. sous-titres, transcriptions, etc.)</p>
A2.4 Développement back-end (côté serveur)	<p><b>C2.4 Piloter le développement back-end d'une application ou d'un site web</b>, concernant des solutions techniques et fonctionnelles, en gérant le serveur, des bases de données, la liaison avec le front-end (API), le back-office, en intervenant sur le choix des outils, des technologies (ex. SGBDR, versioning), des systèmes d'exploitation et des ressources réseaux, avec des langages objet (ex. Python, Java, Node.js), selon des standards techniques (ex. W3C) et des normes de sécurisation (OWASP, RGPD), en collaborant avec d'autres équipes, afin de rendre la solution informatique fonctionnelle.</p>		<p>Ce2.4.1 L'application back-end répond aux objectifs et aux contraintes techniques identifiées</p> <p>Ce2.4.2 Le back-office permet la gestion de l'application dans le respect du cahier des charges et la liaison avec le front-end est assurée par APIs</p> <p>Ce2.4.3 Les outils, technologies et langages objet (Python, Java ou Node.js) sont correctement mis en oeuvre pour répondre aux besoins spécifiques du projet et en les utilisant pour développer des solutions back-end robustes et conformes aux normes de sécurisation</p> <p>Ce2.4.4 La configuration des serveurs et des bases de données permet de répondre aux objectifs et contraintes du projet, et d'assurer la sécurité des données</p> <p>Ce2.4.5 Les standards techniques, les normes de sécurisation (ex. W3C, OWASP) et les conformités réglementaires (ex. RGPD) sont respectés</p> <p>Ce2.4.6 Les choix d'implémentation garantissent une utilisation efficace des ressources et permet de minimiser l'impact environnemental dans le cadre d'un développement numérique responsable (ex. code efficace et optimisé, gestion dynamique des ressources, caching etc.)</p>
A2.5 Traitement des données massives	<p><b>C2.5 Coordonner le développement des solutions technologiques visant le traitement de la donnée à grande échelle</b>, comportant la collecte (ex. avec Open Refine ou Nifi), le traitement par lots ou en temps réel (ex. Apache Spark ou Storm) et le stockage de données massives structurées ou non structurées, en utilisant des bases de données analytiques ou opérationnelles, et en utilisant l'analyse descriptive ou prédictive (ex. machine learning), afin d'extraire toute la valeur des</p>		<p>Ce2.5.1 La mise en œuvre de la collecte et du traitement par lot ou en temps réel des données par des outils technologiques (ex. Open Refine, Nifi, Apache Spark ou Storm) permet de répondre aux contraintes techniques et fonctionnelles du projet (ex)</p> <p>Ce2.5.2 La mise en oeuvre du stockage de données massives structurées ou non structurées, dans des bases de données analytiques ou opérationnelles, permet de</p>
	données disponibles pour l'aide à la décision et automatisation de tâches répétitives		<p>répondre aux contraintes techniques et fonctionnelles du projet, et d'assurer la sécurité et l'intégrité des données</p> <p>Ce2.5.3 La mise en oeuvre d'analyses descriptives et prédictives sur des données massives, y compris l'utilisation de techniques de machine learning, permettent d'extraire de la valeur de la donnée de manière innovante et de répondre aux objectifs et contraintes du projet</p> <p>Ce2.5.4 Les choix technologiques, les frameworks et les outils utilisés tout au long du projet sont présentés clairement, de manière argumentée, pour démontrer comment ces choix contribuent à l'efficacité et à l'innovation de la solution développée</p>