OFPPT
مكتب التكوين المهني و إنعاش الشغل
Office de la Formation Professionnelle
et de la Promotion du Travail

الشركة الوطنية
للإذاعة و التلفزة

Institut Spécialisé de Technologie Appliquée – Hay Salam

RAPPORT DE STAGE

Développement d'une application mobile : e-Facture

Présenté par :

Abderrahim Oumous

Technicien Spécialisé en Développement Digital

Option : Web Full-Stack

Encadré par :

Ayman Shaim (encadrant de stage)

Khalid Mzibra (encadrant pédagogique)

Stage effectué au sein de :

Société Nationale de Radiodiffusion et de Télévision (SNRT)

Du 03 mars 2025 au 04 avril 2025

Remerciements

Je tiens à exprimer ma sincère gratitude à Monsieur **Ayman Shaim**, mon encadrant durant ce stage au sein de la **Société Nationale de Radiodiffusion et de Télévision (SNRT)**. Son accompagnement rigoureux, sa disponibilité constante ainsi que la pertinence de ses conseils ont grandement contribué à la réussite de mon intégration et au bon déroulement de cette expérience professionnelle.

Je remercie chaleureusement l'ensemble des collaborateurs de la SNRT pour l'accueil qu'ils m'ont réservé, la qualité des échanges, ainsi que leur bienveillance tout au long de mon immersion. J'ai eu l'opportunité d'évoluer dans un environnement de travail stimulant, structuré et résolument tourné vers l'innovation, ce qui m'a permis d'approfondir mes connaissances et d'élargir ma vision du développement d'applications dans un contexte professionnel réel.

Ce stage a représenté une étape clé dans mon parcours de formation. Il m'a permis de confronter les compétences acquises en cours aux exigences du terrain, d'affiner mes méthodes de travail, et de renforcer mon autonomie ainsi que ma capacité à collaborer efficacement au sein d'une équipe technique.

J'adresse également mes remerciements à l'équipe pédagogique de l'**Institut Spécialisé de Technologie Appliquée – Hay Salam**, pour la qualité de l'enseignement dispensé, l'encadrement constant, et le suivi personnalisé qui m'ont accompagné depuis le début de ma formation. Leur investissement a été essentiel dans la réussite de mon parcours.

Enfin, je souhaite remercier ma famille et mes proches pour leur soutien moral, leur patience et leur confiance tout au long de mon apprentissage.

Table des matières

Introduction

Dans un monde où la transformation numérique s'impose comme un levier essentiel de performance, la dématérialisation des processus administratifs devient un enjeu stratégique, tant pour les entreprises que pour les institutions publiques. C'est dans cette dynamique que s'inscrit le projet e-Facture, une application mobile destinée à la gestion numérique des factures, permettant aux entreprises de simplifier leurs démarches, sécuriser leurs données et optimiser leur suivi administratif.

Ce rapport s'inscrit dans le cadre de ma formation de **Technicien Spécialisé en Développement Digital** – option **Web Full-Stack**, dispensée par l'**Institut Spécialisé de Technologie Appliquée – Hay Salam**. Il retrace l'ensemble des étapes de conception, de développement et d'intégration de l'application mobile e-Facture, réalisée lors d'un stage de perfectionnement au sein de la **Société Nationale de Radiodiffusion et de Télévision (SNRT)**.

L'objectif principal de ce projet est de concevoir une solution mobile intuitive, sécurisée et performante, répondant à des exigences fonctionnelles réelles, et exploitant une architecture technique moderne (Flutter en frontal et Node.js en backend). Ce projet a été l'occasion de mettre en pratique mes compétences en développement d'interfaces utilisateur, en gestion d'état, en communication client-serveur et en sécurisation des données.

À travers ce document, je présenterai d'abord le contexte institutionnel de la SNRT, avant de détailler les différents modules développés (authentification, facturation, administration), ainsi que l'architecture technique de l'application. Le rapport s'achèvera sur un glossaire des termes techniques, et une conclusion personnelle sur cette expérience formatrice.

Présentation de la société

1. Fiche d'identité

- **Nom complet** : Société Nationale de Radiodiffusion et de Télévision (SNRT)
- **Date de création** : 2005 (succédant à la Radiodiffusion Télévision Marocaine – RTM)
- **Statut juridique** : Établissement public à caractère stratégique
- **Siège social** : Rabat, Maroc
- **Secteur d'activité** : Audiovisuel public (radio et télévision)

2. Historique et évolution

L'histoire de la SNRT s'enracine dans la création de **Radio-Maroc**, le 15 février 1928, qui débute ses premières émissions le 13 avril de la même année. La RTM devient un établissement public en 1966, doté de la personnalité civile et de l'autonomie financière.

En 2005, dans un objectif de modernisation, la RTM est restructurée pour donner naissance à la SNRT. Cette transformation marque un tournant stratégique dans le paysage audiovisuel marocain, avec une volonté claire de diversification, de professionnalisation et d'adaptation aux enjeux du numérique.

3. Missions et objectifs

La SNRT assure la mission de **service public de l'audiovisuel**. Elle produit et diffuse des contenus à caractère informatif, éducatif et culturel. Ses principaux objectifs sont :

- Promouvoir la diversité culturelle et linguistique du Maroc
- Soutenir la création audiovisuelle nationale
- Accompagner les évolutions technologiques dans le secteur des médias
- Garantir un accès équitable à l'information pour tous les citoyens

4. Organisation interne

La SNRT est structurée autour de plusieurs pôles fonctionnels :

- **Pôle Télévision** : gestion des chaînes publiques

- **Pôle Radio** : supervision des stations radio nationales et régionales
- **Pôle Technique** : en charge de l'infrastructure, de la diffusion et de la maintenance
- **Pôle Ressources** : services administratifs, financiers et ressources humaines

5. Chaînes et stations gérées

Télévision :

- *Al Aoula* : chaîne généraliste (information, culture, divertissement)
- *Arryadia* : chaîne sportive
- *Athaqafia* : chaîne culturelle et éducative
- *Al Maghribia* : orientée vers les Marocains du monde
- *Assadissa* : chaîne religieuse
- *Tamazight* : dédiée à la culture et langue amazighes
- *Laayoune TV* : chaîne régionale des provinces du Sud

Radio :

- *Radio Nationale* (Al Idaa Al Watania)
- *Chaîne Inter*
- *Radio Amazigh*
- *Radio Mohammed VI du Saint Coran*
- *Stations régionales* réparties à travers le territoire

6. Rôle dans le paysage médiatique

Acteur central de l'audiovisuel marocain, la SNRT se distingue par son engagement envers une information fiable, diversifiée et accessible. Elle joue un rôle de régulateur face à la montée en puissance des médias privés et des plateformes numériques, tout en soutenant la production nationale et la formation professionnelle dans le domaine des médias.

7. Conclusion

À travers ses multiples plateformes et contenus, la SNRT remplit une mission essentielle de cohésion sociale, de transmission culturelle et d'ouverture sur le monde. Son ancrage historique et son ambition d'innovation font d'elle un pilier du paysage médiatique marocain.

Aperçu fonctionnel de l'application

L'application mobile **e-Facture** a été conçue dans une logique modulaire, orientée utilisateur et conforme aux bonnes pratiques du développement logiciel. Elle répond à un besoin concret : permettre aux entreprises de **gérer numériquement leurs factures** tout en garantissant **sécurité, accessibilité et traçabilité**.

L'ensemble du projet repose sur une architecture claire, découpée en plusieurs **modules fonctionnels**, chacun ayant un rôle précis et complémentaire. Ces modules sont accessibles selon le profil de l'utilisateur (entreprise ou administrateur), et leur enchaînement permet de couvrir l'ensemble du parcours client, depuis la création de compte jusqu'à la gestion avancée des documents et des statistiques.

Les sections suivantes présentent, de manière détaillée, les **trois modules principaux** de l'application :

- Le **module d'authentification**, qui régit l'accès sécurisé à la plateforme
- Le **module de facturation**, cœur fonctionnel de l'application
- Le **module d'administration**, dédié aux utilisateurs ayant des droits avancés

Chacun de ces modules sera présenté à travers ses objectifs, son fonctionnement, ses validations, ses interactions API et son intégration dans l'écosystème global de l'application.

Module d'authentification

1. Présentation générale

Le module d'authentification constitue la porte d'entrée de l'application e-Facture. Il permet de sécuriser l'accès à la plateforme en assurant une gestion rigoureuse des comptes utilisateurs. Trois interfaces principales sont intégrées :

- **Inscription** : création d'un compte entreprise
- **Connexion** : accès sécurisé à l'espace utilisateur
- **Réinitialisation du mot de passe** : en cas de perte ou d'oubli

Ces interfaces communiquent avec un service central, AuthProvider, chargé de la gestion des sessions, des appels API, du stockage sécurisé (token, utilisateur) et du traitement des retours serveur.

---

2. Interface d'inscription

Objectif fonctionnel :

Permettre à une entreprise de créer un compte en fournissant les informations suivantes :

- Raison sociale
- Numéro ICE
- Adresse e-mail

Validations locales :

- Email : format standard
- ICE : numérique, 8 à 15 chiffres
- Raison sociale : entre 3 et 100 caractères

Appel API :

- Méthode : POST /register
- Données : email, ICE, raison sociale
- Réponse : succès/échec + code d'erreur

Comportement post-inscription :

- Génération automatique d'un mot de passe temporaire
- Envoi par e-mail
- Redirection vers la page de connexion (/login)

---

3. Interface de connexion

Objectif fonctionnel :

Permettre à l'utilisateur d'accéder à son espace via e-mail et mot de passe.

Validations locales :

- Email : format conforme
- Mot de passe : champ requis

Appel API :

- Méthode : POST /login
- Réponse : success, isFirstLogin, isAdmin

Comportement post-authentification :

- Redirection selon le profil :
  - isFirstLogin == true → changement de mot de passe
  - isAdmin == true → dashboard admin
  - Sinon → dashboard utilisateur

---

4. Interface de réinitialisation

Objectif fonctionnel :

Permettre de régénérer un mot de passe temporaire via la combinaison :

- E-mail
- Numéro ICE

- Raison sociale

Appel API :

- Méthode : POST /forgot-password
- Réponse : succès, code
- Envoi automatique d'un nouveau mot de passe

---

5. Tableau de synthèse des erreurs

Code	Description
errorsMissingFields	Champs requis non fournis
errorsInvalidEmail	Format e-mail invalide
errorsInvalidIce	ICE non conforme
errorsInvalidLegalName	Raison sociale incorrecte
errorsEmailAlreadyUsed	Adresse déjà utilisée
errorsInvalidCredentials	Identifiants erronés
errorsUserDisabled	Compte désactivé
errorsNetwork	Problème de connexion
errorsInternal	Erreur interne du serveur

---

6. Résumé technique

- **Service principal :** AuthProvider
- **Validations :** via InputValidators
- **Stockage sécurisé :** flutter_secure_storage
- **Feedback utilisateur :** géré par FeedbackHelper
- **Redirections :** automatisées selon les rôles

- **Internationalisation :** tous les messages sont multilingues (S.of(context))

Le module d’authentification combine une structure robuste, une UX fluide, une sécurité renforcée et une gestion claire des rôles, offrant à l'utilisateur une expérience fiable et encadrée dès les premières interactions avec l'application.

Module de facturation

1. Présentation générale

Le module de facturation constitue le **cœur fonctionnel** de l'application e-Facture. Il permet à chaque entreprise utilisatrice de **soumettre, consulter, filtrer, télécharger** ses factures, tout en accédant à des **statistiques de suivi** en temps réel.

Ce module repose sur une logique métier stricte, notamment en ce qui concerne le format, la taille des fichiers et les seuils de montant autorisés. Il est structuré autour de trois interfaces principales, reliées au backend via un service dédié, InvoiceService.

---

2. Interface de création de facture

Objectif fonctionnel :

Permettre à l'utilisateur de soumettre une nouvelle facture en fournissant :

- Un **montant** strictement supérieur à 5 000 000
- Un **fichier PDF**, de taille maximale 2 Mo

Validations locales (client) :

- Vérification du montant (valide, numérique, > 5 000 000)
- Vérification du fichier (présence, format .pdf, taille ≤ 2 Mo)
- Messages d'erreurs multilingues affichés sous les champs

Appel API :

- Méthode : POST /invoices
- Données : montant, fichier, identifiant utilisateur
- Sécurité : jeton JWT dans le header

Comportement post-soumission :

- Affichage d'un message de confirmation
- Rafraîchissement des statistiques

- Redirection vers la page d'historique des factures

---

3. Interface d'historique et de filtrage

Objectif fonctionnel :

Permet à l'utilisateur de :

- Consulter toutes ses factures passées
- Filtrer les résultats par plage de dates
- Télécharger chaque document individuellement

Comportement UX :

- Chargement infini (scroll automatique)
- Vue en **liste** ou **grille**, selon préférence
- Rafraîchissement manuel possible (pull-to-refresh)

Appels API :

- GET /invoices/:userId : récupération standard
- GET /invoices?startDate=...&endDate=... : filtrage avancé

---

4. Interface de tableau de bord

Objectif fonctionnel :

Offrir à l'utilisateur une vue synthétique de son activité :

- Nombre total de factures
- Montant cumulé
- Statistiques visuelles

Données affichées :

- Formatées localement (ex. : NumberFormat)
- Présentées sous forme de **cards** thématiques

- Graphiques intégrés via fl_chart

---

5. Tableau des erreurs courantes

Code	Description
errorsEmptyField	Champ vide (montant ou fichier manquant)
invoiceAmountRequirements	Montant non conforme (< 5 000 000)
invoiceFileRequirements	Fichier invalide (extension ou taille)
errorsLoginFailed	Session expirée ou utilisateur non authentifié

---

6. Résumé technique

- **Service central :** InvoiceService
- **Gestion d'état :** CreateInvoiceViewModel, UserInvoicesViewModel
- **Téléchargement sécurisé :** via dio.download, avec barre de progression
- **Redirections :** automatisées (Navigator.pushReplacementNamed(...))
- **Validations backend :** montant, format, droits d'accès, utilisateur existant
- **Protection JWT :** middleware serveur auth.middleware.js
- **Internationalisation :** toutes les chaînes via S.of(context).xxx

---

Ce module offre à l'utilisateur une interface complète, fluide et sécurisée pour gérer ses documents comptables. Il combine rigueur fonctionnelle, accessibilité mobile et retour utilisateur dynamique, tout en respectant les exigences métier du traitement de factures à grande échelle.

Module d’administration

1. Présentation générale

Le module d’administration est réservé aux utilisateurs disposant de droits spécifiques (isAdmin). Il a pour but de **centraliser la supervision de l’activité de la plateforme**, en offrant des outils simples et efficaces pour la gestion des utilisateurs, des factures, et la consultation de statistiques globales.

Ce module permet une **vision transversale** de l’ensemble des données et repose sur des interfaces sécurisées, reliées à des services spécialisés (AdminUserService, AdminInvoiceService, AdminDashboardService).

---

2. Interface de gestion des utilisateurs

Objectif fonctionnel :

- Visualiser la liste des comptes utilisateurs (pagination)
- Rechercher un utilisateur par mot-clé (nom, ICE, e-mail…)
- Filtrer par statut (actif / inactif)
- Activer ou désactiver un compte
- Accéder aux factures d’un utilisateur donné

Appels API :

- GET /admin/users
- GET /admin/users/search
- PUT /admin/users/:id/disable
- GET /admin/users/:userId/invoices

Comportement UX :

- Actions contextuelles (Activer / Désactiver)
- Expansion individuelle pour voir les détails
- Bouton d'accès rapide aux factures associées

---

3. Interface de consultation des factures

Objectif fonctionnel :

- Accéder à **toutes les factures** de la plateforme
- Filtrer par plage de dates ou mot-clé
- Rechercher une facture spécifique
- Télécharger les documents PDF

Appels API :

- GET /admin/invoices
- GET /admin/invoices?startDate&endDate&keyword
- GET /admin/invoices/:invoiceId/download

Téléchargement sécurisé :

- Jeton JWT requis
- URL temporaire générée
- Suivi de progression via interface + notification locale

Comportement UX :

- Scroll infini avec rafraîchissement dynamique
- Affichage des détails (montant, utilisateur lié, date)
- Zone de filtre réinitialisable en un clic

---

4. Interface dashboard administrateur

Objectif fonctionnel :

- Présenter des **indicateurs globaux** sur l'utilisation de la plateforme
- Analyser l'évolution de l'activité :
    - Nombre d'utilisateurs
    - Répartition des statuts
    - Activité hebdomadaire / mensuelle des factures

- Taux d’adoption de l’outil

Appels API :

- GET /admin/dashboard-stats
- GET /admin/users/status-stats
- GET /admin/invoices/weekly-stats, etc.

Comportement UX :

- Chargement automatique à l’ouverture
- Affichage graphique (camemberts, histogrammes)
- Cartes thématiques avec styles visuels harmonisés

---

5. Tableau des erreurs et protections

Code	Description
noUsersFound	Aucun utilisateur trouvé
noInvoicesFound	Aucune facture correspondant au filtre
errorsNetwork	Erreur de connexion réseau
errorsDownloadFailed	Échec lors du téléchargement du fichier
errorsUnauthorized	Accès refusé : droits insuffisants

---

6. Résumé technique

- **Services utilisés :** AdminUserService, AdminInvoiceService, AdminDashboardService
- **Sécurité :** authentification via JWT, validation isAdmin côté serveur
- **Interface graphique :** composants stylisés, responsives, adaptés mobile/tablette
- **Téléchargements :** gestion via dio, avec permission système Android
- **Navigation :** déclenchée par actions contextuelles (Navigator.pushNamed(...))
- **Multilingue :** interface traduite intégralement via S.of(context).xxx

Grâce à ce module, les administrateurs disposent d'une **interface de supervision claire, performante et évolutive**, qui facilite la gestion quotidienne de la plateforme. Sa structure modulaire et sa sécurisation rigoureuse en font un outil fiable pour le pilotage opérationnel de l'application.

Architecture technique de l'application

1. Présentation générale

L'application e-Facture repose sur une **architecture modulaire moderne**, composée de deux grandes couches techniques :

- Une **application mobile** développée avec **Flutter**
- Un **backend API** sécurisé construit avec **Node.js** et **Express**

Ce découpage permet une séparation claire des responsabilités, une montée en charge facilitée, et une maintenance évolutive dans le temps.

A. Architecture Flutter (frontend)

1. Structure du projet

Le code source Flutter est organisé selon une **architecture MVVM (Model - View - ViewModel)** avec séparation stricte entre interface utilisateur, logique métier et services.

Dossiers principaux :

Dossier	Rôle
/pages/	Interfaces utilisateur (auth, admin, user…)
/viewmodels/	Logique métier + gestion d'état (Provider)
/core/services/	Appels API centralisés
/core/models/	Modèles de données (User, Invoice, Stats…)
/core/providers/	États globaux (utilisateur, thème, langue…)
/themes/	Gestion du thème clair/sombre
/widgets/	Composants UI réutilisables

Dossier	Rôle
/utils/	Fonctions utilitaires (validation, feedback, etc.)
/l10n/ + /generated/	Fichiers de traduction pour l'internationalisation

2. Stack technique

Package	Utilité principale
provider	Gestion d'état (ChangeNotifier)
dio	Appels HTTP + téléchargement de fichiers
flutter_secure_storage	Stockage sécurisé des tokens
intl, flutter_localizations	Multilingue et formatage localisé
fl_chart	Affichage de graphiques (statistiques)
flutter_local_notifications	Notifications locales lors des téléchargements
permission_handler	Gestion des autorisations système (fichiers)
logger, url_launcher, etc.	Outils de débogage, ouverture de liens…

3. Bonnes pratiques appliquées

- Authentification sécurisée via JWT
- Validations locales strictes avant envoi vers l'API
- Architecture claire et testable
- UI réactive, responsive, et multilingue
- Séparation entre logique de présentation (View) et métier (ViewModel)

B. Architecture Backend (Node.js + Express)

1. Structure du backend

Le backend expose une **API REST sécurisée** et modulaire, connectée à une base de données MongoDB. Il gère les rôles, les authentifications, les factures, les statistiques, et la logique d'administration.

Dossiers principaux :

Dossier / Fichier	Rôle
server.js, app.js	Configuration du serveur Express
/routes/	Définition des endpoints (auth, invoice, admin…)
/controllers/	Traitement des requêtes, logique métier
/models/	Schémas Mongoose (User, Invoice)
/middlewares/	Sécurité : vérification JWT, rôles, droits
/services/	Envoi d'e-mails, génération de mots de passe
/utils/	Fonctions utilitaires (validation, hashing, etc.)

2. Fonctionnalités couvertes

Domaine	Endpoints principaux
Authentification	/auth/register, /auth/login, /auth/forgot-password
Facturation	/invoices, /invoices/:id/download, /stats
Administration	/admin/users, /admin/invoices, /admin/stats

3. Sécurité et contrôle d'accès

- Authentification par **JWT**
- Middleware de vérification des **droits (admin / user)**
- Téléchargement sécurisé via **URLs temporaires**
- Validation stricte des entrées via **express-validator**

4. Base de données

- Utilisation de **MongoDB** via **Mongoose**

- Données stockées : utilisateurs, factures, statistiques
- Agrégations pour les rapports d'activité et le dashboard admin

---

Cette architecture permet de garantir une **expérience utilisateur fluide**, une **communication client-serveur sécurisée**, et une **extensibilité** de l'application dans le temps.

Interfaces utilisateur et rendu visuel

Cette section présente un aperçu visuel des principales interfaces de l'application mobile **e-Facture**.

L'application prenant en charge plusieurs langues (français, arabe, anglais) ainsi que deux thèmes visuels (clair et sombre), chaque capture d'écran est affichée dans un contexte spécifique, illustrant à la fois la richesse fonctionnelle et l'adaptabilité de l'interface utilisateur.

Les fonctionnalités associées à chaque écran ont été détaillées dans les sections précédentes. Ici, l'accent est mis sur la présentation visuelle, la diversité d'usage et la cohérence de l'expérience utilisateur à travers les différentes configurations disponibles.

Figure 1 – Écran de lancement (Splash)

Animation d'introduction lors du chargement de l'application. Initialise la session utilisateur et redirige automatiquement vers l'écran correspondant si un utilisateur est déjà enregistré, sinon vers la page d'accueil.

Figure 2 – Page d'accueil

La page d'accueil s'affiche pour les utilisateurs non connectés. Elle présente brièvement l'application et propose un accès direct à la connexion ou à l'inscription.

Figure 3 – Page de connexion

Permet à l'utilisateur de se connecter via e-mail et mot de passe. Redirection automatique vers le bon espace (utilisateur ou admin) selon les rôles après authentification.

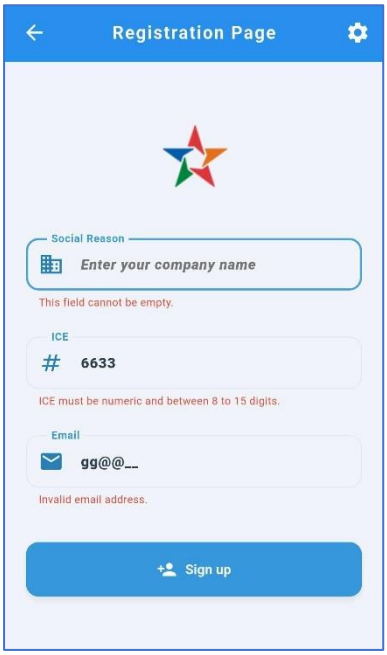

Figure 4 – Création d'un compte entreprise

Interface dédiée à l'enregistrement d'entreprises, avec validation locale des champs (raison sociale, ICE, e-mail) et envoi automatique d'un mot de passe temporaire à l'utilisateur.

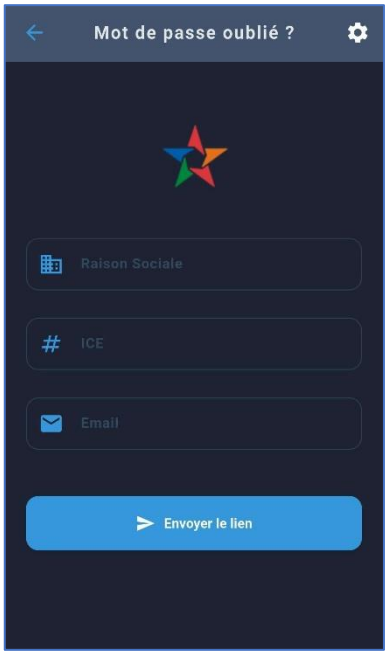

Figure 5 – Réinitialisation du mot de passe

En cas d'oubli, l'utilisateur peut générer un mot de passe temporaire après validation de ses informations d'entreprise. Un e-mail est automatiquement envoyé.

Figure 6 – Changement du mot de passe

Interface permettant à l'utilisateur de modifier son mot de passe en saisissant l'ancien, le nouveau et sa confirmation.

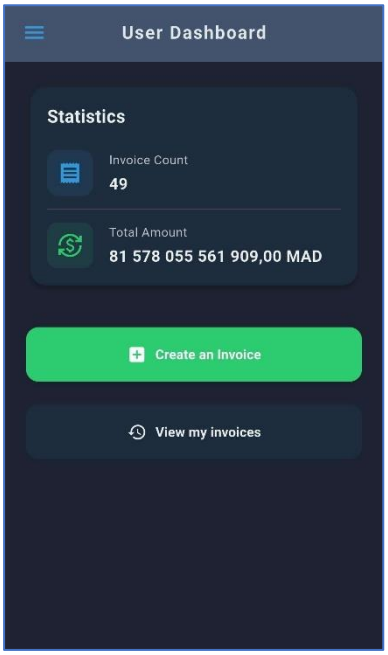

Figure 7 – Tableau de bord utilisateur

Présentation visuelle des factures soumises, incluant le nombre total et le montant cumulé. Des actions permettent de créer une nouvelle facture ou d'accéder à l'historique.

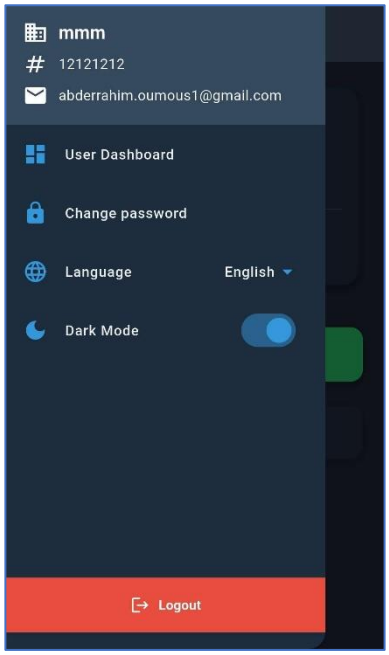

Figure 8 – Menu de navigation latéral

Le Drawer affiche les infos de l'utilisateur connecté, les liens de navigation et les paramètres (langue, thème), avec un contenu dynamique selon son rôle (utilisateur ou admin).

Figure 9 – Création d'une nouvelle facture

Formulaire pour soumettre une facture avec saisie du montant (min. 5M) et ajout d'un fichier PDF. Les validations locales assurent le respect des contraintes avant envoi.

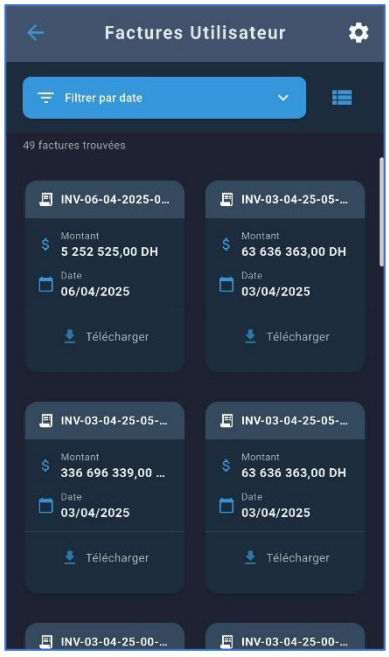

Figure 10 – Historique des factures

Liste interactive de toutes les factures de l'entreprise avec filtres par date, pagination, et téléchargement sécurisé de chaque document.

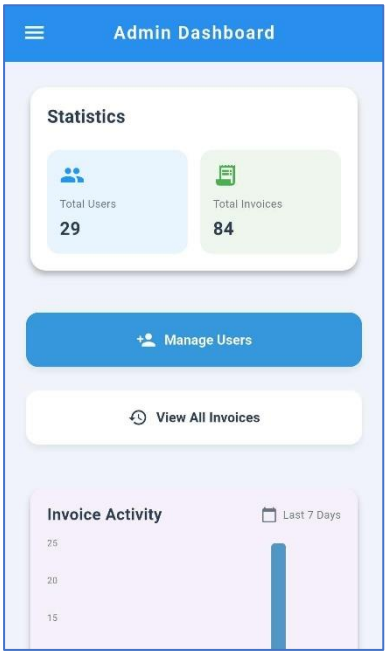

Figure 11 – Tableau de bord administrateur

Présente une vue d'ensemble de la plateforme : graphiques dynamiques sur l'activité (factures, adoption, statuts), statistiques générales (utilisateurs inscrits, total des factures), et accès rapide à la gestion des utilisateurs et à la liste complète des factures.

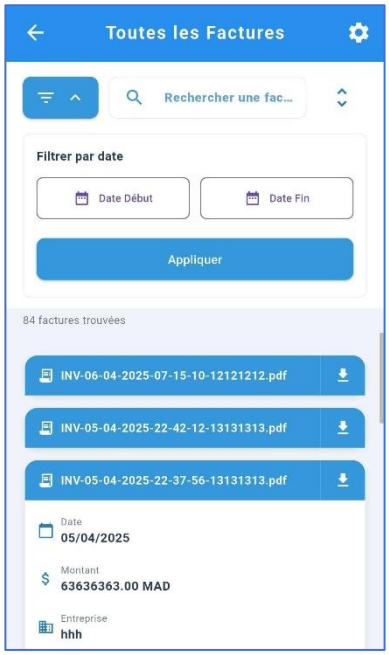

Figure 12 – Vue globale des factures (admin)

Permet à l'administrateur de visualiser toutes les factures, de les filtrer, les rechercher et les télécharger.

Figure 13 – Interface de gestion des utilisateurs

Liste les utilisateurs enregistrés avec options de recherche (par ICE, raison sociale ou email), filtrage actif/inactif, activation/désactivation, et lien vers leurs factures.

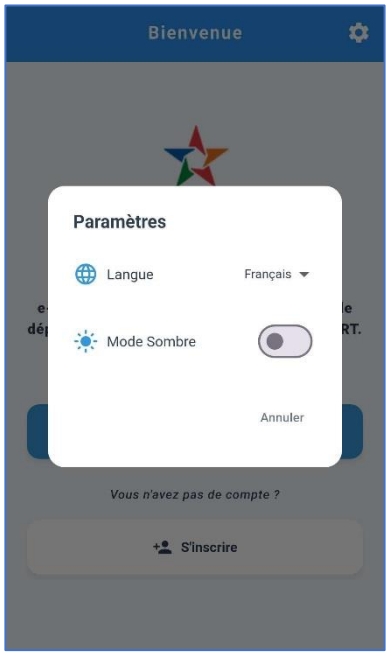

Figure 14 – Réglages rapides (thème et langue)

Permet à l'utilisateur de changer la langue de l'interface et d'activer le thème sombre ou clair via un menu accessible depuis la barre d'app.

Déploiement local et configuration de l'environnement

Pour exécuter l'application e-Facture dans un environnement de développement local, certaines étapes de configuration sont nécessaires, tant pour le backend que pour le frontend.

Prérequis techniques

Le bon fonctionnement du projet nécessite les outils suivants :

- **Node.js** (version 16 ou ultérieure)
- **Flutter SDK** (version stable avec Dart 3.7.0 ou plus récent)
- Un éditeur de code tel que **Visual Studio Code** ou **Android Studio**

Procédure d'installation

1. **Clonage du dépôt** Le code source du projet est disponible sur GitHub. Après clonage :

git clone https://github.com/Azanore/efac.git

cd efac

2. **Mise en place du backend (Node.js)** Dans le dossier e-facture-backend :

npm install

npm start

Le serveur Node.js sera accessible via http://localhost:5000.

3. **Lancement du frontend (Flutter)** Depuis le dossier e_facture :

flutter pub get

flutter run

L'application Flutter peut être exécutée sur un émulateur ou un appareil connecté.

Configuration des variables d'environnement

Deux fichiers .env doivent être créés manuellement :

- Dans le dossier backend (e-facture-backend) :

.env

MONGODB_URI=mongodb://localhost:27017/e_facture_db

JWT_SECRET=***

JWT_EXPIRATION=3600

PORT=5000

SMTP_USER=***

SMTP_PASS=***

- Dans le dossier frontend (e_facture) :

.env

API_URL=http://192.168.11.107:5000/api

AUTH_PATH=/user/auth

INVOICE_PATH=/user/invoices

Les dossiers générés automatiquement comme node_modules, .dart_tool ou build sont exclus du dépôt Git et régénérés automatiquement lors de l'installation.

Déroulement du stage

Semaine 1 : Cadrage, analyse et planification

La première semaine a été consacrée à la **prise de connaissance du projet**, à la lecture du **cahier des charges** et à l'identification des **besoins fonctionnels** pour les deux profils principaux : utilisateur standard et administrateur. J'ai procédé à une analyse structurée des besoins, permettant de dégager les **user stories**, les **ressources nécessaires**, ainsi qu'un **planning par sprints** pour organiser le travail.

Parallèlement, j'ai commencé à manipuler des outils comme **Postman** pour tester des API simples, et mis en place une base de communication entre le frontend Flutter et le backend Node.js via des requêtes de test (ajout, affichage, suppression). J'ai également élaboré un **diagramme de cas d'usage** et un **diagramme de classes** pour structurer les interactions à venir.

Difficultés rencontrées : changements fréquents dans les user stories, manque de maquettes abouties, incertitudes sur les technologies exactes à utiliser et quelques blocages techniques liés à l'environnement de développement et aux premières tentatives avec JWT.

Semaine 2 : Mise en place technique et début du développement

Cette semaine a marqué le **lancement effectif du développement**, avec la mise en place de l'architecture **MVVM** côté Flutter. J'ai intégré des outils comme **Provider** et **SharedPreferences** pour gérer l'état et la persistance des préférences (langue, thème…). Le **sprint 1** a débuté avec l'implémentation de la gestion de thème (sombre/clair), des langues, et de widgets personnalisés.

J'ai développé les premières pages fonctionnelles (connexion, inscription, mot de passe oublié), tout en améliorant la documentation et la cohérence du code. La gestion de l'authentification via **JWT**, le **stockage sécurisé**, et les premiers tests d'envoi de mail ont été mis en place.

Difficultés rencontrées : configuration complexe de certaines dépendances (ex. dio sur web, SDK 34), erreurs d'importations (dotenv), validation de l'envoi de fichiers PDF et ajustement progressif du système de traduction.

Semaine 3 : Fonctionnalités avancées côté utilisateur

Cette semaine a été consacrée à l'ajout de fonctionnalités plus avancées pour l'utilisateur standard. J'ai développé la page **"Mes factures"**, le **dashboard utilisateur** avec statistiques, l'affichage conditionnel, le filtrage par date, la mise en page des cartes et la gestion du **téléchargement sécurisé des factures** (avec progression, notification et ouverture automatique).

Une attention particulière a été portée à l'**optimisation du code**, à la mise en place du **lazy loading**, à l'amélioration du **responsive design**, ainsi qu'à l'expérience utilisateur (messages de fin de liste, splash screen, exécution asynchrone des tâches lourdes…).

Difficultés rencontrées : complexité croissante du code, lenteur de certaines pages, difficultés à faire comprendre certaines logiques à l'IA, et des ajustements techniques importants liés au téléchargement de fichiers et à la structure des pages.

Semaine 4 : Fonctions administratives et filtrage

À ce stade, j'ai développé les interfaces réservées à l'administrateur : consultation des **factures globales**, gestion des **comptes utilisateurs**, affichage conditionnel des **utilisateurs actifs/inactifs**, et passage dynamique des **paramètres entre pages**.

J'ai mis en place le **filtrage par mot-clé, par statut, et par date**, ainsi que la gestion de **l'exportation PDF côté admin**. Les **graphiques statistiques**, les ajustements de thème dynamique, et des corrections de bugs liés à la langue arabe et aux styles ont également été réalisés.

Difficultés rencontrées : résolution de bugs persistants, configuration de composants complexes comme le filtrage croisé, et gestion des conflits visuels liés au thème sombre.

Semaine 5 : Finalisation, nettoyage et amélioration de l'expérience

Durant la dernière semaine, j'ai concentré mes efforts sur le **refactoring** des pages d'authentification, la **gestion des erreurs multilingues**, et l'uniformisation des comportements à travers toute l'application. J'ai également renforcé la **logique métier**, assuré la cohérence des styles, et ajusté les derniers détails fonctionnels.

La phase finale a permis de livrer une version **fonctionnelle, propre et stable**, avec un **code optimisé** et une **expérience utilisateur fluide**, prête à être testée à plus grande échelle.

Glossaire

Terme / Acronyme	Définition
SNRT	Société Nationale de Radiodiffusion et de Télévision (Maroc)
ICE	Identifiant Commun de l'Entreprise : numéro unique attribué à chaque entité juridique au Maroc
JWT	JSON Web Token : jeton d'authentification sécurisé utilisé pour les requêtes protégées
API	Application Programming Interface : interface de communication entre systèmes
Flutter	Framework open-source développé par Google pour créer des applications mobiles multiplateformes
Node.js	Environnement d'exécution JavaScript côté serveur
Express.js	Framework minimaliste pour créer des API web avec Node.js
Provider	Package Flutter de gestion d'état basé sur le modèle ChangeNotifier
ViewModel	Partie de l'architecture MVVM contenant la logique métier de l'interface
MVVM	Model-View-ViewModel : architecture logicielle permettant de séparer les responsabilités
Dio	Bibliothèque Flutter permettant de faire des appels HTTP avancés
MongoDB	Base de données NoSQL utilisée pour stocker les utilisateurs et les factures
Mongoose	Bibliothèque Node.js pour interagir avec MongoDB via des modèles structurés
Middleware	Fonction intermédiaire interceptant les requêtes pour ajouter des vérifications ou traitements
Secure Storage	Mécanisme de stockage chiffré sur mobile (token, données sensibles)
Internationalisation	Capacité d'une application à supporter plusieurs langues

Conclusion

Ce stage au sein de la Société Nationale de Radiodiffusion et de Télévision (SNRT) a constitué une expérience à la fois **formatrice, stimulante et concrète**, me permettant d'appliquer les connaissances acquises en formation tout en découvrant les réalités du développement logiciel en milieu professionnel.

Le projet e-Facture m'a permis d'aborder toutes les phases du cycle de vie d'une application mobile moderne : de la compréhension du besoin métier jusqu'à l'implémentation technique, en passant par la conception modulaire, l'intégration de services, la sécurisation des échanges et la gestion d'état. Grâce à l'utilisation de technologies comme **Flutter** pour le frontend et **Node.js** pour le backend, j'ai pu mettre en œuvre une architecture complète, robuste et évolutive.

Sur le plan technique, ce stage m'a permis de renforcer mes compétences en **gestion d'authentification sécurisée**, en **traitement de fichiers**, en **communication client-serveur** via API REST, ainsi qu'en **structuration d'un code modulaire et maintenable**. Il m'a également sensibilisé à l'importance de l'UX, de la sécurité des données, et de la qualité de l'expérience utilisateur dans un environnement mobile.

D'un point de vue personnel, ce stage a été l'occasion de développer des **soft skills essentiels** : autonomie, rigueur, esprit d'analyse, gestion des priorités et communication professionnelle. J'ai pu m'adapter à un cadre de travail exigeant, tout en bénéficiant d'un encadrement bienveillant et motivant.

Enfin, ce projet représente pour moi bien plus qu'un simple exercice académique : c'est une première **expérience concrète de réalisation d'une solution métier**, dans un contexte réel, avec de vrais utilisateurs et de vraies contraintes. Il constitue une base solide pour la suite de mon parcours professionnel, que je souhaite orienter vers le développement d'applications mobiles modernes et à fort impact.