# Grimoire Complet du Projet Hedwige & CI/CD

Ce document est une archive complète et commentée de la création de l'application "Hedwige". Il contient les explications de chaque étape ainsi que le code final et fonctionnel correspondant, prêt à être versionné avec Git.

## I. Le Backend : "Hedwige-Express" (Python & Docker)

Notre backend est une petite application Python qui sert de cerveau à notre projet. Il est conçu pour être léger, testable et facilement déployable.

### Étape 1 : Création du Squelette de l'Application

**Objectif :** Mettre en place une base de code minimale mais robuste, avec des tests pour garantir son bon fonctionnement.

**Code hedwige-express/requirements.txt (version initiale) :**

Flask

pytest

**Code hedwige-express/app.py (version initiale) :**

from flask import Flask, jsonify

app = Flask(\_\_name\_\_)

@app.route("/")

def index():

return jsonify({"message": "Hedwige est prête !"})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**Code hedwige-express/tests/test\_app.py :**

import pytest

from app import app as flask\_app

@pytest.fixture

def app():

yield flask\_app

@pytest.fixture

def client(app):

return app.test\_client()

def test\_index(client):

"""Teste la route principale."""

response = client.get('/')

assert response.status\_code == 200

assert response.json == {"message": "Hedwige est prête !"}

### Étape 2 : La Dockerisation (le Portoloin)

**Objectif :** Rendre notre application portable et reproductible grâce à Docker, en la préparant pour un environnement de production.

**Code hedwige-express/requirements.txt (version mise à jour pour la production) :**

Flask

pytest

gunicorn

**Code hedwige-express/Dockerfile :**

# Étape 1 : L'image de base

FROM python:3.9-slim

# Étape 2 : Définir le répertoire de travail dans le conteneur

WORKDIR /app

# Étape 3 : Installer les dépendances

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

# Étape 4 : Copier le code de l'application

COPY . .

# Étape 5 : Exposer le port que Gunicorn utilisera

EXPOSE 5001

# Étape 6 : Commande pour lancer l'application

CMD ["gunicorn", "--bind", "0.0.0.0:5001", "app:app"]

### Étape 3 : Ajout de la Route d'Authentification

**Objectif :** Donner au backend la capacité de vérifier l'identité d'un utilisateur en validant un jeton Google.

**Code hedwige-express/requirements.txt (version finale) :**

Flask

pytest

gunicorn

google-auth

requests

flask-cors

**Code hedwige-express/app.py (version finale) :**

from flask import Flask, jsonify, request

from google.oauth2 import id\_token

from google.auth.transport import requests

from flask\_cors import CORS # Important pour autoriser les requêtes du frontend

app = Flask(\_\_name\_\_)

# Active CORS pour autoriser les requêtes depuis http://localhost:4200

CORS(app, resources={r"/api/\*": {"origins": "http://localhost:4200"}})

@app.route("/")

def index():

return jsonify({"message": "Hedwige est prête !"})

@app.route("/api/auth/google", methods=['POST'])

def google\_auth():

token\_data = request.get\_json()

token = token\_data.get('credential')

if not token:

return jsonify({"error": "Jeton manquant"}), 400

try:

# Remplacez par votre propre ID Client (celui du frontend)

CLIENT\_ID = "168327457573-ap4pe9peilrhpcdpnri7kvo4lb6q7eds.apps.googleusercontent.com"

id\_info = id\_token.verify\_oauth2\_token(token, requests.Request(), CLIENT\_ID)

user\_info = {

"email": id\_info['email'],

"name": id\_info['name'],

"picture": id\_info['picture'],

"given\_name": id\_info.get('given\_name', '')

}

return jsonify(user\_info), 200

except ValueError as e:

print(f"Erreur de vérification du jeton : {e}")

return jsonify({"error": "Jeton Google invalide"}), 401

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True, port=5001)

## II. Le Pipeline CI/CD (GitHub Actions)

**Objectif :** Automatiser les tests, la validation du code et la publication de notre image Docker à chaque modification du code.

### Étape 1 : Le Pacte des Secrets (Docker Hub & GitHub)

**Objectif :** Permettre à GitHub Actions de se connecter à Docker Hub de manière sécurisée, sans jamais exposer de mot de passe dans le code.

**Action A : Créer un Jeton d'Accès sur Docker Hub**

1. Se connecter à [hub.docker.com](https://hub.docker.com).
2. Cliquer sur son nom d'utilisateur en haut à droite, puis sur **"Account Settings"**.
3. Aller dans la section **"Security"**.
4. Cliquer sur **"New Access Token"**.
5. Donner un nom au jeton (ex: github-actions-hedwige) et cliquer sur **"Generate"**.
6. **Copier immédiatement le jeton généré**. C'est le seul moment où il sera visible.

**Action B : Stocker les Secrets dans GitHub**

1. Aller sur le dépôt GitHub du projet.
2. Cliquer sur l'onglet **"Settings"**, puis **"Secrets and variables" > "Actions"**.
3. Cliquer sur **"New repository secret"**.
   * **Name :** DOCKERHUB\_USERNAME
   * **Secret :** Votre nom d'utilisateur Docker Hub.
4. Cliquer à nouveau sur **"New repository secret"**.
   * **Name :** DOCKERHUB\_TOKEN
   * **Secret :** Le jeton d'accès copié depuis Docker Hub.

### Étape 2 : Création du Fichier de Workflow

**Code .github/workflows/main.yml :**

name: Pipeline CI/CD pour Hedwige-Express

# Déclencheur : s'exécute à chaque push sur la branche main

on:

push:

branches: [ "main" ]

jobs:

build-and-test:

runs-on: ubuntu-latest

steps:

# Étape 1 : Récupérer le code du dépôt

- name: Checkout code

uses: actions/checkout@v3

# Étape 2 : Configurer Python

- name: Set up Python

uses: actions/setup-python@v4

with:

python-version: '3.9'

# Étape 3 : Installer les dépendances

- name: Install dependencies

run: |

python -m pip install --upgrade pip

pip install flake8 pytest

pip install -r requirements.txt

# Étape 4 : Linter le code (vérification de la qualité)

- name: Lint with flake8

run: |

# stop the build if there are Python syntax errors or undefined names

flake8 . --count --select=E9,F63,F7,F82 --show-source --statistics

# exit-zero treats all errors as warnings. The GitHub editor is 127 chars wide

flake8 . --count --exit-zero --max-complexity=10 --max-line-length=127 --statistics

# Étape 5 : Lancer les tests unitaires

- name: Run Unit Tests

run: |

python3 -m pytest

# Étape 6 : Se connecter à Docker Hub

- name: Login to Docker Hub

uses: docker/login-action@v2

with:

username: ${{ secrets.DOCKERHUB\_USERNAME }}

password: ${{ secrets.DOCKERHUB\_TOKEN }}

# Étape 7 : Construire et pousser l'image Docker

- name: Build and push Docker image

uses: docker/build-push-action@v4

with:

context: .

push: true

tags: ${{ secrets.DOCKERHUB\_USERNAME }}/hedwige-express:latest

## III. Le Frontend : "Hedwige-Frontend" (Angular)

**Objectif :** Créer une interface utilisateur interactive qui permet de s'authentifier avec Google et de communiquer avec notre backend.

**Code hedwige-frontend/src/index.html :**

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>HedwigeFrontend</title>

<base href="/">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" type="image/x-icon" href="favicon.ico">

<!-- Ajout du script officiel de connexion Google -->

<script src="[https://accounts.google.com/gsi/client](https://accounts.google.com/gsi/client)" async defer></script>

</head>

<body>

<app-root></app-root>

</body>

</html>

**Code hedwige-frontend/src/app/app.config.ts (version finale et nettoyée) :**

import { ApplicationConfig } from '@angular/core';

import { provideRouter } from '@angular/router';

import { routes } from './app.routes';

import { provideHttpClient } from '@angular/common/http';

export const appConfig: ApplicationConfig = {

providers: [

provideRouter(routes),

provideHttpClient() // Activation du module de communication HTTP

]

};

**Code hedwige-frontend/src/app/app.component.ts (version finale et fonctionnelle) :**

import { Component, NgZone, AfterViewInit, inject } from '@angular/core';

import { CommonModule } from '@angular/common';

import { DomSanitizer, SafeHtml } from '@angular/platform-browser';

import { HttpClient } from '@angular/common/http';

import { jwtDecode } from 'jwt-decode';

declare const google: any;

@Component({

selector: 'app-root',

standalone: true,

imports: [CommonModule],

template: `

<div class="flex h-screen w-full bg-gray-900 text-gray-200 font-sans">

<aside class="w-64 flex-shrink-0 bg-gray-800/50 p-6">

<div class="flex items-center space-x-3 mb-10">

<svg xmlns="[http://www.w3.org/2000/svg](http://www.w3.org/2000/svg)" class="h-8 w-8 text-cyan-400" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"><path d="M22 12h-4l-3 9L9 3l-3 9H2"/></svg>

<span class="text-2xl font-bold tracking-wider">Hedwige</span>

</div>

<nav class="space-y-3">

<a \*ngFor="let folder of folders" href="#"

(click)="setActiveFolder(folder.name); $event.preventDefault()"

[class]="getFolderClasses(folder.name)">

<div class="h-6 w-6" [innerHTML]="sanitizeIcon(folder.icon)"></div>

<span class="font-medium">{{ folder.name }}</span>

</a>

</nav>

</aside>

<main class="flex-1 flex flex-col">

<header class="flex h-24 flex-shrink-0 items-center justify-between border-b border-gray-700/50 px-8">

<h1 class="text-3xl font-light text-gray-400">{{ activeFolder }}</h1>

<div \*ngIf="user" class="flex items-center space-x-4">

<span class="font-medium">{{ user.name }}</span>

<img [src]="user.picture" class="h-12 w-12 rounded-full" alt="Photo de profil" />

<button (click)="signOut()" class="rounded-full bg-gray-700 px-4 py-2 text-white">Déconnexion</button>

</div>

<div \*ngIf="!user" class="h-10 w-10 rounded-full bg-gradient-to-br from-cyan-400 to-purple-500"></div>

</header>

<div class="flex-1 flex items-center justify-center p-8">

<div \*ngIf="!user" class="w-full max-w-lg text-center">

<h2 class="text-4xl font-bold text-white mb-4">Bienvenue dans Hedwige Mail</h2>

<p class="text-lg text-gray-400 mb-8">

Connectez-vous pour envoyer et recevoir vos parchemins numériques.

</p>

<div id="google-btn"></div>

</div>

<div \*ngIf="user" class="w-full max-w-lg text-center">

<h2 class="text-4xl font-bold text-white mb-4">Bienvenue, {{ user.given\_name }} !</h2>

<p class="text-lg text-gray-400 mb-8">Vous êtes authentifié auprès de notre service.</p>

</div>

</div>

</main>

</div>

`

})

export class AppComponent implements AfterViewInit {

private sanitizer = inject(DomSanitizer);

private ngZone = inject(NgZone);

private http = inject(HttpClient);

user: any = null;

activeFolder = 'Boîte de réception';

backendUrl = 'http://localhost:5001';

folders = [

{ name: 'Boîte de réception', icon: '<svg xmlns="[http://www.w3.org/2000/svg](http://www.w3.org/2000/svg)" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"><path d="M20 13V6a2 2 0 00-2-2H6a2 2 0 00-2 2v7m16 0v5a2 2 0 01-2 2H6a2 2 0 01-2-2v-5m16 0h-2.586a1 1 0 00-.707.293l-2.414 2.414a1 1 0 01-.707.293h-3.172a1 1 0 01-.707-.293l-2.414-2.414A1 1 0 006.586 13H4" /></svg>' },

{ name: 'Envoyés', icon: '<svg xmlns="[http://www.w3.org/2000/svg](http://www.w3.org/2000/svg)" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"><path d="M12 19l9 2-9-18-9 18 9-2zm0 0v-8" /></svg>' },

{ name: 'Corbeille', icon: '<svg xmlns="[http://www.w3.org/2000/svg](http://www.w3.org/2000/svg)" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round"><path d="M19 7l-.867 12.142A2 2 0 0116.138 21H7.862a2 2 0 01-1.995-1.858L5 7m5 4v6m4-6v6m1-10V4a1 1 0 00-1-1h-4a1 1 0 00-1 1v3M4 7h16" /></svg>' }

];

ngAfterViewInit(): void {

google.accounts.id.initialize({

client\_id: '168327457573-ap4pe9peilrhpcdpnri7kvo4lb6q7eds.apps.googleusercontent.com',

callback: (response: any) => {

this.ngZone.run(() => {

this.handleCredentialResponse(response);

});

}

});

google.accounts.id.renderButton(document.getElementById('google-btn'), { theme: 'outline', size: 'large', type: 'standard', shape: 'pill' });

}

handleCredentialResponse(response: any) {

if (response.credential) {

this.http.post(`${this.backendUrl}/api/auth/google`, { credential: response.credential })

.subscribe({

next: (backendUser: any) => {

console.log("Authentification réussie côté backend !", backendUser);

// On utilise les infos vérifiées par le backend pour plus de sécurité

this.user = backendUser;

},

error: (err) => {

console.error("Erreur d'authentification backend:", err);

this.signOut();

}

});

}

}

signOut(): void {

google.accounts.id.disableAutoSelect();

this.user = null;

}

setActiveFolder(folderName: string) { this.activeFolder = folderName; }

sanitizeIcon(icon: string): SafeHtml { return this.sanitizer.bypassSecurityTrustHtml(icon); }

getFolderClasses(folderName: string): string {

const baseClasses = "flex items-center space-x-4 rounded-lg p-3 text-gray-400 transition-all duration-200 hover:bg-gray-700/50 hover:text-white";

if (folderName === this.activeFolder) { return `${baseClasses} bg-cyan-400/20 text-cyan-300`; }

return baseClasses;

}

}

**Présentation de l'Entreprise : AG2R La Mondiale**

****

AG2R La Mondiale est un groupe leader en France spécialisé dans la protection sociale et patrimoniale. Avec une histoire de plus de 115 ans, AG2R La Mondiale se distingue par son modèle de gouvernance paritaire et mutualiste, plaçant les assurés au centre de sa stratégie et de ses décisions. Le groupe accompagne plus de 15 millions de personnes et 500 000 entreprises, ce qui représente environ une entreprise sur quatre en France.

AG2R La Mondiale est un organisme français, à but non lucratif, de protection sociale et patrimoniale. Sa gouvernance repose sur le paritarisme et le mutualisme.

**Les Activités Principales :**

* **Épargne et Retraite** : AG2R La Mondiale offre des solutions pour préserver le niveau de vie à la retraite, avec des produits d'épargne et de retraite complémentaire et supplémentaire.
* **Santé** : Le groupe propose des assurances pour protéger la santé de ses assurés et de leurs proches, répondant aux besoins de couverture médicale.
* **Prévoyance** : AG2R La Mondiale aide à se prémunir contre les aléas de la vie tels que l'incapacité de travail, l'invalidité, et le décès.
* **Assurance Vie** : Le groupe sécurise les revenus et prépare l'avenir de ses clients grâce à des produits d'assurance vie adaptés.

résentation de l'Entreprise : AG2R La Mondiale

AG2R La Mondiale est le résultat de l'union entre deux grands acteurs de la protection sociale en France : AG2R et La Mondiale.

Historique :

La Mondiale a été fondée en 1905, spécialisée principalement dans les assurances de personnes, notamment la gestion de l'épargne et de la retraite.

AG2R (Association Générale de Retraite par Répartition), de son côté, a été créée en 1951. AG2R s'est concentré sur la protection sociale, couvrant des domaines comme la retraite complémentaire, la santé, et la prévoyance.

Fusion :

En 2008, AG2R et La Mondiale ont décidé de fusionner pour former AG2R La Mondiale, un groupe majeur de la protection sociale et patrimoniale. Cette fusion a permis de combiner les forces d'AG2R dans les régimes de retraite et de prévoyance avec l'expertise de La Mondiale dans l'assurance-vie et la gestion de patrimoine.

Le Groupe Aujourd'hui :

Depuis la fusion, AG2R La Mondiale est devenu un acteur incontournable en France, protégeant plus de 15 millions de personnes et accompagnant plus de 500 000 entreprises. Cette fusion a permis de renforcer la capacité du groupe à offrir une large gamme de solutions d’assurance pour les particuliers et les professionnels, tout en maintenant une gouvernance paritaire et mutualiste qui implique activement les assurés dans les décisions stratégiques.

### Contexte et Défis du Projet

**Besoin :** Le projet vise à simplifier et sécuriser le processus de demande de badge pour les nouveaux collaborateurs au sein d'AG2R La Mondiale. Actuellement, il est nécessaire de créer à la fois une identité et un compte de messagerie pour initier une demande de badge, ce qui complique et alourdit le processus et que les managers sont obliger d’envoyer un mail avec les informations nécessaires et la photo de bénéficiaire. Le but est donc de permettre la création de demandes de badge directement à partir de l'identité du bénéficiaire, sans nécessité de créer un compte de messagerie, tout en incluant la photo du bénéficiaire et en respectant les normes de sécurité et de protection des données.

**Cahier des Charges :**

Ø **Exigences Fonctionnelles :**

* **Identification Unique** : La demande de badge doit être exclusivement liée à l'identité du bénéficiaire, créée via l'application Mercury.
* **Inclusion de la Photo** : La photo du bénéficiaire doit être incluse dans la demande et stockée conformément aux normes définies dans un répertoire partagé DIMEX.
* **Site de Récupération** : Chaque demande doit inclure le site spécifique de récupération du badge, adapté aux multiples emplacements d'AG2R en France.
* **Habilitations Spécifiques** : Les autorisations spécifiques nécessaires selon le profil du bénéficiaire doivent être incluses dans la demande.
* **Validation Hiérarchique et Fonctionnelle** : Seuls les managers, après validation hiérarchique et fonctionnelle, sont autorisés à faire une demande de badge.

Ø **Exigences Non Fonctionnelles :**

* **Sécurité** : Toutes les données personnelles et photos doivent être sécurisées et conformes aux normes de protection des données de l'entreprise.
* **Accessibilité** : L'interface doit être accessible uniquement aux managers autorisés.

Ø **Contraintes Techniques :**

* **Technologies Utilisées** : Angular pour le front-end, ASP.NET pour le back-end, et SQL Server pour la base de données.
* **Formats de Photo** : Les photos doivent être au format JPEG, JPG, avec un fond blanc et ne doivent pas dépasser 5 mégaoctets.

Ø **Conditions d'Utilisation :**

* **Plages Horaires** : La demande doit préciser les plages horaires de présence du bénéficiaire (cadre: 6h-20h, standard: 7h-19h).
* **Commentaires** : Une section commentaires doit permettre aux managers de spécifier toute demande particulière ou contexte spécifique lié au besoin du badge.

Ø **Validation et Acceptation des Livrables :**

* Chaque fonctionnalité sera développée en sprints et validée par des revues de sprint pour s'assurer qu'elle répond aux besoins exprimés.
* Des tests d'intégration et de sécurité seront effectués pour garantir la fiabilité et la conformité du système.

**Livrables Attendues :**

1. **Fonctionnalité de Demande de Badge avec Photo :**
   * Développement d'une fonctionnalité permettant de créer une demande de badge en incluant la photo du bénéficiaire. Cette fonctionnalité doit être intégrée de manière fluide dans l'application Mercury360, permettant une gestion facile des demandes par les managers.
   * Les photos doivent être validées pour respecter les critères de format (JPEG, JPG) et de taille (maximum 5 mégaoctets).
2. **Intégration de l'Identité Physique :**
   * Implémentation de l'intégration avec le système existant pour utiliser uniquement l'identité physique du bénéficiaire créée via l'application Mercury. Cette intégration vise à éliminer la nécessité de créer un compte de messagerie, simplifiant ainsi le processus.
3. **Gestion des Sites de Récupération :**
   * Ajout d'une fonctionnalité pour spécifier le site de récupération du badge dans chaque demande. Cette fonctionnalité doit prendre en compte les différents emplacements d'AG2R La Mondiale à travers la France.
4. **Inclusion des Habilitations Spécifiques :**
   * Développement d'une option pour inclure les habilitations spécifiques nécessaires en fonction du rôle et des besoins du bénéficiaire. Cela inclut des autorisations d'accès particulières selon les sites de récupération.
5. **Interface Utilisateur Accessible et Sécurisée :**
   * Création d'une interface utilisateur intuitive et sécurisée, accessible uniquement aux managers autorisés, permettant de faire des demandes de badge après une validation hiérarchique et fonctionnelle.
   * L'interface doit comporter une section de commentaires pour que les managers puissent ajouter des notes spécifiques liées à la demande.
6. **Système de Notification :**
   * Développement d'un système de notification qui informe les managers de la création d'une identité, les invitant à compléter la demande de badge.
7. **Documentation Complète :**
   * Fourniture de la documentation technique et utilisateur détaillée, incluant les guides d'utilisation, les spécifications techniques, et les procédures de sécurité.

Ces livrables garantissent que le système mis en place répondra aux besoins spécifiques d'AG2R La Mondiale tout en maintenant des standards élevés de sécurité et d'efficacité opérationnelle.

### Gestion de Projet

#### Planning et Suivi

Dans le cadre de ce projet, la gestion de projet a été assurée selon la méthode Agile, et plus précisément avec le framework Scrum. Cette approche a permis une gestion itérative et incrémentale du projet, favorisant l'adaptation continue aux besoins évolutifs de l'entreprise et une livraison progressive des fonctionnalités. Voici comment le planning et le suivi ont été organisés :

1. **Sprints :**
   * Le projet a été découpé en sprints de trois semaines. Chaque sprint avait pour objectif de livrer une ou plusieurs fonctionnalités utilisables du projet. Ce découpage a permis de concentrer les efforts sur des lots de travail définis, facilitant ainsi la gestion des priorités et l'intégration des retours des parties prenantes.
2. **Réunions de Suivi :**
   * **Daily Stand-ups :** Chaque jour, l'équipe se réunissait pour un bref point de situation (15 minutes), permettant à chacun de partager ses avancées, de soulever les blocages éventuels et de synchroniser les actions à venir.
   * **Sprint Planning :** Au début de chaque sprint, une réunion de planification était organisée pour définir les tâches à accomplir. Les User Stories (histoires utilisateurs) étaient sélectionnées du backlog produit en fonction de leur priorité et de la capacité de l'équipe.
   * **Sprint Review :** À la fin de chaque sprint, une revue de sprint permettait de présenter le travail accompli aux parties prenantes, de recueillir leurs feedbacks et de valider les fonctionnalités développées.
   * **Rétrospective :** Après chaque sprint, une réunion de rétrospective était organisée pour analyser ce qui s'était bien passé et identifier les points d'amélioration pour les sprints futurs. Cette démarche d'amélioration continue est un pilier de la méthode Agile.
3. **Outils de Suivi :**
   * **Jira :** Un outil de gestion de projet comme Jira a été utilisé pour suivre l'avancement des tâches, gérer le backlog et faciliter la communication au sein de l'équipe.
   * **Tableaux Kanban :** Les tableaux Kanban ont été utilisés pour visualiser l'état des tâches (à faire, en cours, terminé), offrant une vue d'ensemble claire sur l'avancement du projet.
4. **Communication et Collaboration :**
   * La méthode Scrum favorise une communication régulière et transparente entre les membres de l'équipe, les Product Owners (PO), et les parties prenantes. Cette collaboration étroite a permis de répondre rapidement aux changements de besoins et d'assurer que le produit final correspondait aux attentes des utilisateurs finaux.

En adoptant cette approche Agile et en structurant le projet autour de cycles de travail courts et itératifs, l'équipe a pu fournir des livrables de qualité, respecter les délais, et répondre de manière proactive aux défis rencontrés tout au long du projet.

### Environnement Humain

Dans le cadre du projet de demande de badge avec photo, j'ai eu l'opportunité de travailler au sein de l'équipe dédiée aux habilitations. Cette équipe fait partie intégrante de l'organisation AG2R La Mondiale, qui comprend plusieurs autres sections spécialisées telles que la santé prévoyance et la data science. Voici une description de la composition et des rôles au sein de mon équipe :

* **Product Owner (PO) :** Le PO est responsable de définir la vision du produit et de prioriser les besoins en fonction des attentes des utilisateurs finaux et des objectifs d'entreprise. Le PO joue un rôle crucial dans la validation des fonctionnalités développées, s'assurant que chaque livrable répond aux besoins exprimés.
* **Proxy Product Owner (PPO) :** Le PPO assiste le PO en gérant les tâches quotidiennes et en faisant le lien avec l'équipe de développement. Le PPO aide également à clarifier les exigences et à s'assurer que l'équipe est alignée sur les objectifs du projet.
* **Business Analyst :** Le Business Analyst est chargé de recueillir, analyser et documenter les besoins des parties prenantes. Il joue un rôle clé dans la traduction des exigences métiers en spécifications fonctionnelles que l'équipe de développement peut utiliser.
* **Scrum Master :** Le Scrum Master facilite les processus Scrum au sein de l'équipe. Il organise les cérémonies Scrum (sprints planning, daily stand-ups, sprint review, rétrospectives), aide à résoudre les obstacles rencontrés par l'équipe, et assure que les principes Agile sont respectés tout au long du projet.
* **Développeurs (3) :** L'équipe comprend trois développeurs, y compris moi-même. Nous sommes responsables de la mise en œuvre des fonctionnalités définies, en suivant les bonnes pratiques de développement et en garantissant la qualité du code produit. Les développeurs collaborent étroitement avec le PO, le PPO, et les Business Analysts pour s'assurer que les solutions techniques répondent aux besoins fonctionnels.
* **Ops (DevOps) :** Le DevOps est en charge de la gestion de l'infrastructure, de l'intégration continue et du déploiement des applications. Il veille à ce que les environnements de développement, de test et de production soient stables, sécurisés et fonctionnels. Le DevOps joue également un rôle important dans l'automatisation des processus de déploiement pour améliorer l'efficacité de l'équipe.

Cette structure d'équipe permet une collaboration étroite entre les différentes fonctions, garantissant une compréhension commune des objectifs et des exigences du projet. En travaillant ensemble, nous assurons la livraison de solutions de qualité qui répondent aux besoins de l'entreprise tout en respectant les délais et les contraintes du projet. La diversité des compétences au sein de l'équipe permet de couvrir tous les aspects du développement, de l'analyse des besoins à la mise en production, assurant ainsi une approche complète et intégrée du projet.

### Objectifs de Qualité

**Objectifs de qualité** fait référence aux standards de performance et d'excellence que le projet doit atteindre pour être considéré comme réussi. Dans le contexte de votre projet de demande de badge avec photo, les objectifs de qualité peuvent inclure plusieurs aspects :

1. **Sécurité :**
   * Protéger les données personnelles des utilisateurs, notamment les photos des bénéficiaires et les informations d'identité. L'application doit respecter les normes de sécurité de l'entreprise et se conformer aux réglementations en matière de protection des données (RGPD par exemple).
   * Utilisation de techniques de chiffrement comme BCrypt pour protéger les mots de passe et de l'authentification via JWT pour sécuriser l'accès à l'application.
2. **Maintenabilité :**
   * Le code de l'application doit être bien structuré, documenté et modulaire, facilitant la maintenance et les futurs développements. Cela inclut l'adoption des meilleures pratiques de développement et des principes de conception.
   * Utilisation de tests automatisés pour détecter rapidement les erreurs et assurer la stabilité du système au fil du temps.
3. **Conformité :**
   * L'application doit se conformer aux normes internes de l'entreprise ainsi qu'aux exigences légales et réglementaires applicables. Cela comprend les normes de sécurité de l'information, de confidentialité des données, et de gestion des identités.

En résumant, les **objectifs de qualité** visent à garantir que le produit livré non seulement répond aux attentes fonctionnelles mais est aussi sécurisé, performant, facile à utiliser, et durable à long terme. Ces objectifs sont cruciaux pour assurer la satisfaction des utilisateurs finaux et la réussite globale du projet.

### Analyse du Besoin

L'analyse du besoin est une étape essentielle dans la conduite de votre projet, car elle permet de comprendre les exigences des utilisateurs finaux et de définir clairement les fonctionnalités nécessaires pour répondre à ces exigences. Voici comment structurer cette section :

#### 1. **Identification du Besoin**

Le besoin principal identifié pour ce projet est de permettre une gestion plus efficace et sécurisée des demandes de badge pour les nouveaux collaborateurs d'AG2R La Mondiale. Actuellement, la création d'un badge nécessite non seulement la création d'une identité dans le système Mercury, mais aussi la création d'un compte de messagerie, ce qui alourdit le processus. Le besoin est donc de simplifier cette procédure en permettant la création de demandes de badge uniquement avec l'identité du bénéficiaire, sans la nécessité d'un compte de messagerie.

#### 2. **Objectifs de la Solution**

* **Simplification du Processus :** Offrir une solution qui permet de créer une demande de badge en utilisant uniquement l'identité du bénéficiaire, créée dans l'application Mercury.
* **Inclusion de la Photo :** Ajouter la capacité de joindre une photo du bénéficiaire à la demande de badge pour faciliter l'identification et l'émission des badges.
* **Amélioration de la Sécurité :** Intégrer des mesures de sécurité robustes pour protéger les informations sensibles des bénéficiaires, notamment les données personnelles et les photos.
* **Facilitation de l'accès :** Permettre aux managers de soumettre des demandes de badge avec des autorisations spécifiques selon le profil du bénéficiaire, ainsi que la sélection du site de récupération du badge.

#### 3. **Analyse des Parties Prenantes**

* **Managers :** Ils ont besoin d'une interface simplifiée pour créer et soumettre des demandes de badge. Ils doivent pouvoir sélectionner facilement le bénéficiaire, ajouter des autorisations spécifiques, et joindre une photo.
* **Service des Ressources Humaines :** Ce service a besoin d'assurer que le processus de demande de badge est conforme aux politiques internes de sécurité et de gestion des accès.
* **Équipe de Sécurité :** Cette équipe s'assure que toutes les données personnelles sont sécurisées et conformes aux réglementations sur la protection des données. Elle est également concernée par l'attribution des habilitations spécifiques.
* **Bénéficiaires :** Les nouveaux collaborateurs qui ont besoin de leurs badges d'accès pour leurs premiers jours. Une procédure simplifiée et rapide est essentielle pour garantir une intégration fluide.

#### 4. **Contraintes Identifiées**

* **Contraintes Techniques :** Utilisation d'Angular pour le front-end, ASP.NET pour le back-end, et SQL Server pour la gestion des données. La solution doit s'intégrer sans heurts avec l'infrastructure existante de l'entreprise.
* **Contraintes de Sécurité :** Respecter les politiques de sécurité internes et les réglementations externes comme le RGPD pour le traitement des données personnelles.
* **Contraintes de Temps :** Le projet doit être réalisé dans un délai limité, en respectant les cycles de développement définis par la méthodologie Scrum, avec des sprints de trois semaines.

#### 5. **Solutions Proposées**

* **Développement d'une Interface Utilisateur Intuitive :** Créer une interface simple et facile à utiliser pour que les managers puissent soumettre des demandes de badge avec toutes les informations nécessaires.
* **Intégration des Photos :** Mettre en place une fonctionnalité pour uploader les photos des bénéficiaires directement lors de la demande de badge.
* **Automatisation des Notifications :** Développer un système de notifications pour informer les managers et les validateurs des étapes de validation et de finalisation de la demande.
* **Sécurité Renforcée :** Utiliser des technologies de sécurisation comme JWT pour l'authentification et le chiffrement des données sensibles, assurant ainsi la confidentialité et l'intégrité des données.

En résumé, l'analyse du besoin permet de clarifier les attentes et les exigences des différentes parties prenantes, de définir les objectifs de la solution, et d'identifier les contraintes techniques et de sécurité à respecter. Cela sert de base pour la conception et le développement d'une solution qui répond de manière efficace et sécurisée aux besoins identifiés.

### Conception et Codage

La phase de conception et de codage de votre projet de demande de badge avec photo a été structurée en plusieurs étapes clés pour garantir une mise en œuvre efficace et de haute qualité. Voici le détail des étapes suivies :

#### 1. **Brainstorming pour l'Interface Utilisateur**

La première étape a été de concevoir une interface utilisateur intuitive et conviviale pour permettre aux managers de créer facilement des demandes de badge. Voici comment cela a été réalisé :

* **Réunions de Brainstorming :** J'ai participé à plusieurs sessions de brainstorming avec le Product Owner (PO), le Proxy Product Owner (PPO), et les Business Analysts pour définir les besoins et les attentes en matière d'interface utilisateur. Ces sessions ont permis de recueillir des idées sur la disposition des éléments, les fonctionnalités nécessaires et les interactions utilisateur.
* **Outil de Maquettage - Figma :** Les idées recueillies lors des brainstorming ont été transformées en maquettes interactives utilisant Figma. Cet outil a permis de visualiser les écrans de l'application, y compris la sélection des bénéficiaires, le téléchargement des photos, la sélection des sites de récupération et l'ajout des habilitations spécifiques. Les maquettes ont été discutées et affinées jusqu'à ce qu'elles répondent aux exigences fonctionnelles et aux attentes des utilisateurs.
* **Feedback et Ajustements :** Les maquettes ont été présentées à l'équipe élargie pour recueillir des feedbacks. Les ajustements ont été faits en fonction des commentaires reçus, garantissant ainsi que l'interface soit intuitive et que toutes les fonctionnalités nécessaires soient couvertes.

#### 2. **Développement Front-End**

Une fois les maquettes validées, la phase de développement front-end a commencé :

* **Choix des Technologies :** Angular a été choisi comme framework principal pour le développement du front-end en raison de sa modularité, sa performance et sa facilité d'intégration avec d'autres services. Material Design a été utilisé via Angular Material pour assurer une interface visuelle cohérente et moderne.
* **Création des Composants :** Les composants essentiels de l'application ont été développés, notamment :
  + DemandeBadgeComponent : Composant principal pour gérer les demandes de badge.
  + BeneficiaireSelectorComponent : Pour sélectionner l'identité du bénéficiaire.
  + PhotoUploaderComponent : Pour permettre le téléchargement des photos des bénéficiaires.
  + HabilitationsSelectorComponent : Pour sélectionner les habilitations spécifiques et le site de récupération.
* **Data Binding et Communication entre Composants :** Une approche de partage de données entre composants parent et enfant a été utilisée pour faciliter la communication. L'utilisation de @Input() et @Output() a permis de lier les données et de déclencher des événements de manière efficace.
* **Tests Unitaires :** Des tests unitaires ont été écrits pour chaque composant pour s'assurer que les interactions utilisateur produisent les résultats attendus, et que les composants fonctionnent correctement dans différentes conditions.

#### 3. **Versionning avec Git**

Pour assurer un suivi rigoureux des modifications et permettre une collaboration efficace, un système de versioning a été mis en place :

* **Création d'une Branche Dédiée :** J'ai créé ma propre branche sur le dépôt Git de l'application pour y réaliser mes développements. Cela a permis de travailler de manière isolée sans impacter la branche principale avant que les fonctionnalités soient complètes et testées.
* **Commits Réguliers :** Des commits réguliers ont été effectués, accompagnés de messages descriptifs pour documenter les changements. Cela a facilité la revue de code et la traçabilité des modifications.
* **Pull Requests et Revues de Code :** Avant de fusionner les modifications dans la branche principale, des pull requests ont été créées. Les membres de l'équipe ont effectué des revues de code pour valider la qualité et la conformité des changements avant leur intégration.

#### 4. **Développement Back-End**

En parallèle du développement front-end, la partie back-end a été conçue et développée pour supporter les fonctionnalités de l'application :

* **Technologie Utilisée :** ASP.NET a été utilisé pour le développement du back-end. Cette technologie permet de créer des API robustes et sécurisées, facilitant la gestion des requêtes et des réponses.
* **Création des API :**
  + GetBadgeInfo : API principale pour récupérer les informations nécessaires à la création d'une demande de badge (sites de récupération, habilitations spécifiques, etc.).
  + Gestion des requêtes et des réponses pour assurer une communication efficace entre le front-end et le back-end.
* **Middleware de Sécurité :** Un middleware spécifique a été mis en place pour filtrer les requêtes et vérifier que seuls les utilisateurs ayant les bonnes autorisations peuvent accéder aux fonctionnalités de l'application. Cela inclut l'authentification via JWT et la vérification des rôles des utilisateurs.

#### 5. **Intégration et Tests**

* **Intégration Continue :** Une intégration continue a été mise en place pour déployer automatiquement les modifications validées dans un environnement de test. Cela a permis de détecter rapidement les bugs et les problèmes d'intégration.
* **Tests d'Intégration :** Des tests d'intégration ont été réalisés pour s'assurer que les composants front-end communiquent correctement avec les API back-end et que toutes les fonctionnalités fonctionnent comme prévu.
* **Validation des Livrables :** Chaque fonctionnalité a été validée lors des revues de sprint, en impliquant les parties prenantes pour garantir que le produit final répond aux attentes et exigences définies.

En suivant ces étapes de conception et de codage, le projet a pu avancer de manière structurée et efficace, en répondant aux besoins fonctionnels et techniques tout en garantissant une qualité élevée du code et des livrables. Cette méthodologie a permis de livrer une application performante et sécurisée, répondant aux besoins de l'entreprise et facilitant la gestion des demandes de badge pour les nouveaux collaborateurs.

b

### Slide : Gestion d'Erreur et Tests Fonctionnels pour la Demande de Badge

#### **Titre :** Gestion des Erreurs et Validation des Demandes de Badge

#### **Contexte et Objectifs :**

La fonctionnalité de demande de badge est essentielle pour le bon fonctionnement des processus d'accès. Pour garantir la qualité et la fiabilité de ce processus, une gestion rigoureuse des erreurs et des tests fonctionnels approfondis sont nécessaires.

#### **Gestion d'Erreur :**

1. **Validation des Champs Obligatoires :**
   * Les champs obligatoires incluent la sélection du bénéficiaire, la photo du bénéficiaire et le site de récupération.
   * Si l'un de ces champs n'est pas renseigné, un message d'erreur clair est affiché en bas de la page :
     + Exemple : "Veuillez choisir un bénéficiaire et un site de récupération et une photo de bénéficiaire."
   * Cette validation garantit que toutes les informations nécessaires sont fournies avant de pouvoir soumettre la demande de badge.
2. **Contrôle des Formats de Photo :**
   * Les photos doivent être au format JPEG, JPG ou BMP, avec une taille maximale de 5 mégaoctets et un fond blanc.
   * Si la photo ne répond pas à ces critères, un message d'erreur est affiché et la photo ne peut pas être téléchargée.
3. **Gestion des Sélections Multiples :**
   * Les habilitations spécifiques et les plages horaires peuvent inclure plusieurs options. Une validation est mise en place pour éviter les incohérences ou les erreurs dans la sélection.

#### **Tests Fonctionnels :**

1. **Scénarios de Tests :**
   * **Test de Validation de Champs :** Vérifier que les messages d'erreur s'affichent correctement lorsque des champs obligatoires sont manquants.
   * **Test de Format de Photo :** Valider que seules les photos répondant aux spécifications (taille, format, fond) sont acceptées.
   * **Test de Sélection des Options :** Assurer que les utilisateurs peuvent sélectionner les habilitations spécifiques et les plages horaires appropriées sans conflit.
2. **Exécution des Tests :**
   * Les tests ont été effectués en utilisant des scénarios réels de demande de badge, simulant différentes configurations et erreurs possibles.
   * Chaque étape du processus de demande a été testée pour s'assurer que le système répond correctement à toutes les situations, tant pour les entrées valides que pour les erreurs.

### Synthèse et Conclusion

Le projet de gestion de demande de badges au sein de l'application Mercury a été une réussite, démontrant ma capacité à mener à bien des initiatives de développement logiciel complexes. En commençant par l'analyse des besoins, le projet a progressé de manière structurée à travers des phases de conception, de développement, et de test, tout en respectant les exigences de qualité et de sécurité.

**Synthèse des Accomplissements** :

* J'ai su m'adapter à des technologies modernes telles qu'Angular pour le front-end et ASP.NET pour le back-end, garantissant ainsi une application performante et intuitive.
* La mise en place d'un système de communication efficace entre les composants du front-end et la logique métier du back-end a permis d'assurer la cohérence des données.
* J'ai développé une solution robuste de gestion des accès et des droits, en intégrant des mécanismes de sécurité tels que les JWT et les Guards en Angular, répondant aux exigences strictes de protection des données de l'organisation.
* Les tests fonctionnels et d'intégration ont validé la stabilité et la fiabilité du système, assurant que les fonctionnalités répondent aux besoins des utilisateurs finaux et aux attentes des parties prenantes.