



RÉPUBLIQUE FRANÇAISE

MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Nom de naissance

- HAUGER

Nom d'usage

- HAUGER

Prénom

- Félix

Adresse

- 24 rue Jeanne Jugan, 13004 MARSEILLE

Titre professionnel visé

Concepteur Développeur d'Applications

MODALITÉ D'ACCÈS :

- Parcours de formation
- Validation des Acquis de l'Expérience (VAE)

DOSSIER PROFESSIONNEL (DP)

Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel. **Ce titre est délivré par le Ministère chargé de l'emploi.**

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen.**

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.

Il est consulté par le jury au moment de la session d'examen.

Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du **Dossier Professionnel (DP)** dans lequel le candidat a consigné les preuves de sa pratique professionnelle
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]

Ce dossier comporte :

- pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- une déclaration sur l'honneur à compléter et à signer ;
- des documents illustrant la pratique professionnelle du candidat (facultatif)
- des annexes, si nécessaire.

DOSSIER PROFESSIONNEL^(DP)

Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.



<http://travail-emploi.gouv.fr/titres-professionnels>

DOSSIER PROFESSIONNEL^(DP)

Sommaire

Exemples de pratique professionnelle

1 - Développer une application sécurisée	p. 6
---	-------------

▸ Haptify	p. p. 6
▸ VSION	p. p. 16
▸ QuickPath	p p. 25

2 - Concevoir et développer une application sécurisée organisée en couches	p. 33
---	--------------

▸ Haptify	p. p. 33
▸ VSION	p. p. 37

3 - Préparer le déploiement d'une application sécurisée	p. 41
--	--------------

▸ Haptify	p. p. 44
▸ VSION	p. p. 49
▸ QuickPath	p p.

Titres, diplômes, CQP, attestations de formation (facultatif) p.

Déclaration sur l'honneur p.

Documents illustrant la pratique professionnelle (facultatif) p.

Annexes (Si le RC le prévoit) p.

DOSSIER PROFESSIONNEL ^(DP)

EXEMPLES DE PRATIQUE PROFESSIONNELLE

DOSSIER PROFESSIONNEL (DP)

Activité-type 1 Développer une application sécurisée

Exemple n°1 - Haptify

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Projet d'entreprise effectué dans le cadre de l'alternance. Le projet s'est découpé sur une période de plusieurs mois.

Ce projet collaboratif portait sur la conception et le développement d'une application web permettant la création, la personnalisation et la génération d'effets haptiques.

L'objectif principal est de rendre plus accessible la création d'effets haptiques pour les développeurs et les designers.

Nous nous sommes organisés de manière agile : des réunions courtes quotidiennes le matin pour énumérer nos tâches, nos estimations et les difficultés attendues, et une réunion en fin de semaine pour faire le point sur le travail accompli. Nous avons géré le projet grâce à notion, qui nous a servi à gérer nos tâches et suivre nos réunions.

En premier lieu, j'ai créé un formulaire d'inscription à la Beta, pour récupérer les informations d'utilisateurs potentiels, et les tenir au courant des avancées et sorties du produit.

Join the Beta !

By taking part in the Haptify Beta, you'll benefit from its features free of charge during the launch phase, and you'll be helping to improve its performance, so don't hesitate!

Complete it

First Name	Last Name
Email	Company Name
Password	Confirm Password

I agree that my data will be used to improve software development
 I accept Haptify's terms of use

Join The Beta

Le backend est géré par AWS Amplify, qui lui-même écrit dans une base de données DynamoDB grâce au langage de requêtes GraphQL.

J'ai aussi ajouté un QRcode pour faciliter le partage du formulaire.

DOSSIER PROFESSIONNEL (DP)

J'ai créé un dépôt git et l'ai push sur GitHub pour permettre à l'équipe de contribuer au projet.

La première version de Haptify porte sur la génération d'effets pour les plateformes dites "2d". Sont concernés principalement les smartphones, en premier lieu sur le Web puis sur iOS et Android.

Sécurité : Authentification

Dans le cadre du développement sécurisé de l'application, j'ai mis en place un système d'authentification complet pour les utilisateurs. Ce système repose sur l'intégration d'AWS Amplify, qui s'appuie sur Amazon Cognito pour la gestion des identités, l'inscription, la connexion, la récupération de mot de passe et la gestion sécurisée des sessions.

J'ai veillé à configurer les différents flux d'authentification en respectant les bonnes pratiques de sécurité recommandées par AWS et l'ANSSI, en particulier concernant la validation des entrées utilisateur, la protection contre les attaques XSS ou CSRF, et la gestion sécurisée des jetons d'accès.

Les données sensibles, comme les identifiants d'accès et tokens JWT, sont protégées par des politiques de confidentialité strictes et stockées côté client selon les préconisations de sécurité de Cognito.

Côté frontend, j'ai utilisé le composant `withAuthenticator`, fourni par la bibliothèque Amplify UI, pour protéger les pages nécessitant une authentification. Ce composant encapsule la logique de vérification de session et permet d'appliquer des restrictions d'accès par rôle ou statut utilisateur, tout en garantissant une expérience utilisateur fluide.

Cette intégration s'inscrit dans une architecture logique en couches, où la couche de présentation reste découpée de la logique métier et des services d'authentification.

Ce travail a également nécessité l'analyse du besoin en sécurisation de l'accès aux ressources, la configuration des règles d'accès sur le backend, et l'adaptation de l'interface utilisateur selon les différents cas d'usage (utilisateur connecté, déconnecté, non autorisé). Les entrées sont systématiquement validées et les erreurs sont gérées de manière défensive, conformément aux principes de développement sécurisé.

Utilisation du component `withAuthenticator`

```
export default withAuthenticator(Start, {
  components: components
});
```

Gestion des paiements avec Stripe

Dans le cadre de mon application web proposant des plans payants, j'ai fait le choix d'utiliser Stripe pour gérer les paiements et abonnements récurrents.

DOSSIER PROFESSIONNEL (DP)

Ce choix s'explique par plusieurs critères techniques, sécuritaires et réglementaires :

1. Sécurité des données bancaires

L'une des principales raisons est que Stripe prend en charge directement la gestion des données bancaires, ce qui m'évite d'avoir à manipuler ou stocker des informations sensibles comme :

- les numéros de cartes bancaires
- les CVV
- les données de transaction brutes

Stripe est certifié PCI DSS niveau 1, le plus haut niveau de certification en matière de sécurité des données de paiement. Cela garantit :

- un chiffrement fort des données bancaires
- un stockage sur des infrastructures dédiées et surveillées
- la protection contre les fraudes

Ainsi, l'application reste conforme aux exigences de sécurité sans avoir à les implémenter manuellement, ce qui est fortement recommandé pour les projets non bancaires.

2. Conformité RGPD et simplicité légale

Stripe est également conforme au RGPD, et agit comme sous-traitant du traitement. Il fournit :

- une documentation claire sur la gestion des données personnelles
- des outils de suppression/anonymisation à la demande
- une politique de confidentialité détaillée et accessible (<https://stripe.com/fr/privacy>)

Cela me permet d'éviter de collecter des données bancaires sensibles, et de limiter les risques juridiques liés à une mauvaise gestion des données personnelles.

Comportement métier : gestion des abonnements Stripe

L'application propose plusieurs plans d'abonnement payants, gérés via la plateforme de paiement Stripe. Lorsqu'un utilisateur souscrit à un plan, la logique métier met en œuvre un processus sécurisé d'abonnement et de synchronisation avec le backend.

J'ai mis en place un moyen de paiement grâce à l'intégration Stripe au sein même de l'application.

Le comportement métier comprend plusieurs étapes clés :

DOSSIER PROFESSIONNEL (DP)

1. Crédation de la session de paiement

Logique du Component de Checkout

```
const CheckoutButton = ({ priceId }) => {
  const router = useRouter();

  const handleClick = async () => {
    try {
      const redirectPath = `/payment/checkout/${priceId}`;
      console.log("redirectPath", redirectPath);
      router.push(redirectPath);
    } catch (err) {
      console.error("Error while redirecting to checkout:", err);
    }
  }
}
```

Pages de paiement avec le routage dynamique de Next.js

```
payment
  checkout
    [...priceld].tsx
  return
    [...priceld].tsx
```

Lorsqu'un utilisateur clique sur « S'abonner », l'application Next.js envoie une requête à une API (/api/checkout) qui :

- crée une session Stripe Checkout
- initialise les paramètres du plan choisi
- redirige l'utilisateur vers l'interface de paiement Stripe intégrée

DOSSIER PROFESSIONNEL (DP)

Intégration Stripe dans Next.js

S'abonner à Premium Subscription
29,99 € par mois

Payer avec link

Ou

E-mail

email@example.com

Moyen de paiement

Informations de la carte

1234 1234 1234 1234



MM / AA

CVC

Nom du titulaire de la carte

Nom complet

Pays ou région

France

Enregistrer mes informations pour régler plus rapidement

Régalez vos achats sur ce site plus rapidement ainsi que partout où Link est accepté.

Payer et s'abonner

En confirmant votre abonnement, vous autorisez à débiter votre compte pour les paiements à venir, conformément à ses conditions. Vous pouvez annuler votre abonnement à tout moment.

Propulsé par **stripe** | Conditions d'utilisation Confidentialité

2. Paiement et redirection

Une fois le paiement validé, Stripe redirige l'utilisateur vers une URL de succès (`success_url`) ou d'échec (`cancel_url`), définies dans la session.

L'utilisateur revient sur le site avec un message de confirmation d'abonnement ou d'annulation.

3. Traitement des webhooks Stripe (backend Node.js/Next.js)

Le backend expose une route `POST /api/webhooks` qui reçoit les événements Stripe en temps réel (ex. `checkout.session.completed`, `invoice.paid`, `customer.subscription.deleted`, etc.).

DOSSIER PROFESSIONNEL (DP)

Les secrets pour accéder à Stripe sont stockés dans des variables d'environnement qui ne sont jamais loggées ni accessibles en public.

```
const stripe = require('stripe')(process.env.STRIPE_SECRET_KEY);
const webhookSecret = process.env.STRIPE_WEBHOOK_SECRET;
```

Chaque webhook est vérifié avec une signature secrète (Stripe-Signature) pour éviter toute requête frauduleuse.

```
export default async function POST(req: NextApiRequest, res: NextApiResponse) {
    const body = await buffer(req);

    const signature = req.headers['stripe-signature'];

    let data;
    let eventType;
    let event;

    // Verify the webhook signature and construct the event
    try {
        event = stripe.webhooks.constructEvent(body, signature, webhookSecret);
    } catch (err: any) {
        console.error(`Webhook signature verification failed. ${err.message}`);
        return res.json({ error: "Webhook signature verification failed.", status: 400 });
    }

    data = event.data;
    eventType = event.type;
```

On teste ensuite le type d'évènement pour mettre le statut de l'abonnement de l'utilisateur à jour (active, canceled, trialing, etc.).

DOSSIER PROFESSIONNEL (DP)

```
try {
  switch (eventType) {
    case 'checkout.session.completed': {
      // First payment is successful and a subscription is created (if mode was set to "subscription" in ButtonCheckout)
      // ✓ Grant access to the product
      let user;

      const session = await stripe.checkout.sessions.retrieve(
        data.object.id,
        {
          expand: ['line_items']
        }
      );

      const customerId = session?.customer;
      const customer = await stripe.customers.retrieve(customerId);
      const priceId = session?.line_items?.data[0]?.price.id;

      if (customer.email) {
        // 1. Check if user exists
        const email = customer.email;

        const queryListUserByEmail = gql`query ListUsers($filter: ModelUserFilterInput) {
          listUsers(filter: $filter) {
            items {
              id
              email
              createdAt
            }
          }
        }`;
        const { data } = await client.query({
          query: queryListUserByEmail,
          variables: { email }
        });
        user = data[0];
      }
    }
  }
}
```

On teste grâce à une requête GraphQL si l'utilisateur existe déjà, grâce à son adresse entrée précédemment sur le formulaire de paiement.

Si l'utilisateur n'est pas inscrit sur l'application, on le crée avec une mutation GraphQL.

```
// 1.1 If no user found, create it
if (!user) {
  console.log("no user found, creating user...");
  // Create user here
  const mutationCreateUser = gql`mutation CreateUser($input: CreateUserInput!) {
    createUser(input: $input) {
      id
      email
      customerId
      tokenLeft
    }
  }`;

  const { data } = await client.mutate({
    mutation: mutationCreateUser,
    variables: {
      input: {
        email: email,
        customerId: customerId,
        tokenLeft: 0
      }
    }
  });

  res.status(201).json({ success: true, user: data.createUser });
  user = data.createUser;
}
```

Dans les deux cas, je récupère ses données pour la suite.

DOSSIER PROFESSIONNEL (DP)

Ensuite, je teste si le plan existe bien en base de données grâce au planId, et si c'est le cas je donne accès à l'utilisateur aux features associées.

```
// 2. Update user data by giving them access to selected plan/features
(async () => {
  const queryListPlansByPriceId = gql`query ListPlans($filter: ModelPlanFilterInput) {
    listPlans(filter: $filter) {
      items {
        id
        stripePriceId
      }
    }
  }`;
  const userData = await client.query({
    query: queryListPlansByPriceId,
    variables: { priceId }
  });

  const plan = userData.data.getPlanByPriceId;

  if (!plan) {
    console.error('No matching plan found');
    throw new Error('No matching plan found for the price ID');
  }

  const mutationUpdateUser = gql`mutation UpdateUser($input: UpdateUserInput!) {
    updateUser(input: $input) {
      id
      stripePriceId
      planUsersId
      hasAccess
    }
  }`;
  const { data } = await client.mutate({
    mutation: mutationUpdateUser,
    variables: {
      input: {
        id: user.id,
        stripePriceId: priceId,
        planUsersId: plan.id,
        hasAccess: true
      }
    }
  });

  res.status(200).json({ success: true, user: data.updateUser });
})();
```

DOSSIER PROFESSIONNEL (DP)

Des emails peuvent être déclenchés via un service tiers si nécessaire.

2. Précisez les moyens utilisés :

- **AWS Amplify** pour gérer le back-end
- **VSCode** comme IDE / éditeur de code
- **Amplify CLI** pour gérer le développement et le déploiement du back-end en local
- **git** comme gestionnaire de contrôle de version
- **GitHub** pour sauvegarder le projet et rendre son code source accessible à l'équipe
- **React** comme librairie JavaScript
- **NextJS** comme Framework Web
- **Stripe** pour les paiements / abonnements
- **figma** pour le wireframe et le design des interfaces
- **notion** pour la gestion du projet
- **google agenda** pour l'emploi du temps

3. Avec qui avez-vous travaillé ?

Avec

- un développeur logiciel alternant
- une designer
- une docteure en informatique
- une doctorante spécialiste en haptique
- une scrum master
- le chef d'entreprise

DOSSIER PROFESSIONNEL (DP)

4. Contexte

Nom de l'entreprise, organisme ou association - V.RTU

Chantier, atelier, service - R&D

Période d'exercice - Du 01/2024 au 02/2025

5. Informations complémentaires (*facultatif*)

Haptify est une application qui permet à ses utilisateurs de générer et personnaliser des effets haptiques.

Les effets haptiques sont des effets qui permettent aux utilisateurs d'avoir des retours de sensations par le sens du toucher.

Les utilisateurs connectés ont un nombre de tokens limité, selon leur abonnement, pour générer des effets haptiques. En tapant un prompt, on appelle l'API qui utilise des IA génératives pour créer des effets spécifiques.

DOSSIER PROFESSIONNEL (DP)

Activité-type 1 Développer une application sécurisée

Exemple n°2 - VSION

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Projet d'entreprise effectué dans le cadre de l'alternance. Le projet s'est découpé sur une période de plusieurs mois.

VSION est un système de motion capture servant à entraîner des usagers à effectuer des tâches spécifiques.

VSION calcule et enregistre les positions des humains dans l'espace.

J'ai repris un projet existant. Une partie du système était déjà en place en langage python. Je devais le rendre livrable avec une UI et une UX acceptables.

La gestion des tâches s'est faite sur notion. De plus, des réunions courtes et régulières ont été mises en place pour suivre l'avancement du projet.

Comme choix de technologie, j'ai opté pour Flask, un framework Web Python avec Flask-SocketIO, une librairie associée au framework permettant la communication avec des websockets dans les applications.

J'ai créé le dépôt git, et je l'ai push sur GitHub.

J'ai créé un environnement virtuel venv pour verrouiller les versions de Python et de ses packages pour mon projet. Avec en plus un fichier requirements.txt, cela m'a permis de pouvoir travailler avec les mêmes dépendances peu importe l'environnement de travail.

J'ai également documenté l'installation dans le fichier README, au cas où quelqu'un d'autre devait travailler sur le projet ou bien si je devais moi-même le réinstaller.

Instructions d'installation :

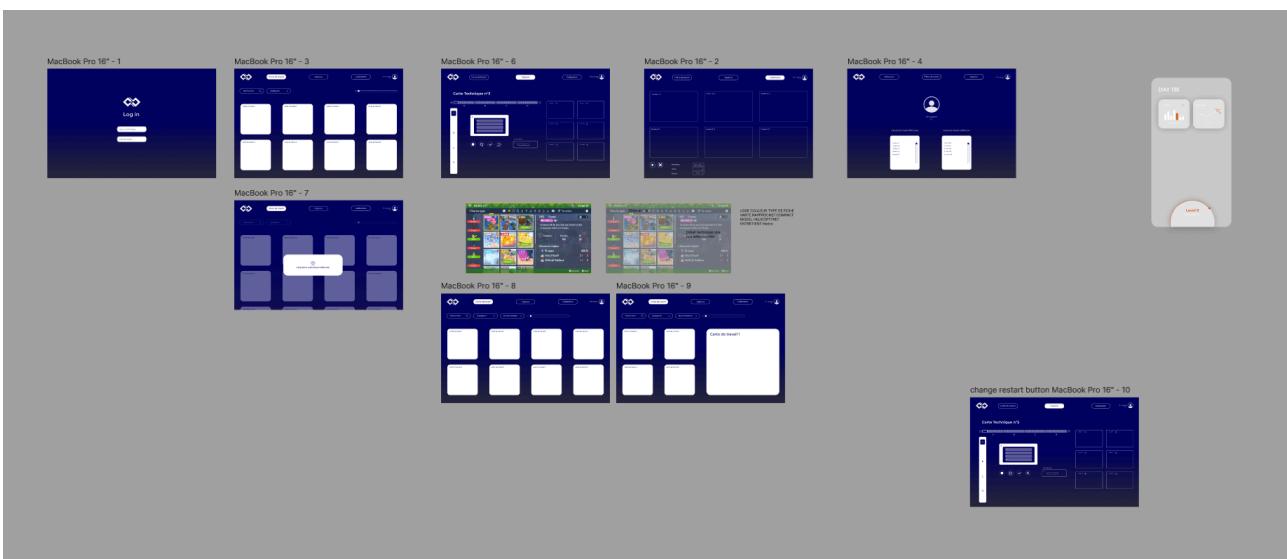
```
## Installation
1. Clone the repository
2. Install the virtual environment: `python3.10 -m venv .venv`
3. Activate the virtual environment: `.\venv\Scripts\activate`
4. Install the dependencies: `pip install -r requirements.txt`
5. Run the app: `python app.py`
   - If python can't find the modules in the virtual environment, check if the pip version is the one inside the virtual environment
   - For a temporary fix you can try `sys.path.append(r".\venv\Lib\site-packages")` before importing the modules
6. Open your browser and go to `localhost:5000`
```

requirements.txt :

DOSSIER PROFESSIONNEL (DP)

```
Flask == 3.0.0
Flask-SocketIO == 5.3.6
python-socketio == 5.10.0
mediapipe == 0.10.0
pandas == 1.5.2
pyyaml == 6.0
open3d == 0.16.0
opencv-python == 4.8.1.78
cython == 3.0.6
pyinstaller == 6.3.0
flask-sqlalchemy == 3.1.1
flask-migrate == 4.0.7
```

J'ai travaillé de concert avec la designer sur figma pour créer les interfaces, tels que des boutons d'enregistrement et des fiches de travail.



J'ai ensuite développé ces interfaces et le reste de l'application avec le framework web Python Flask, en utilisant Flask-SocketIO pour les communications en socket, comme par exemple des interactions exploitant le principe des flags.

En effet, la diffusion des flux vidéos sur une application client-serveur a demandé au départ de la réflexion sur comment afficher et traiter l'image en direct. C'est une feature obligatoire pour que le projet soit mené à bien.

Fort heureusement, des documentations et exemples sont présentes pour ce type d'applications, notamment pour des applications de sécurité et surveillance vidéo, ce qui a permis de résoudre cette problématique et de commencer le développement du projet.

DOSSIER PROFESSIONNEL (DP)

Lecture des frames avec opencv avec gestion de leur traitement et de l'enregistrement dans des fichiers vidéo. Chaque fonction tourne dans un thread.

```
def get_frames(self, index):
    while True:
        ret, frame = self.cameras[index].cap.read()
        if ret:
            self.frames[index] = frame
            if not self.processing.is_set():
                self.texture_raw_feeds[index] = self.frames[index]
            if self.record.is_set():
                self.writers[index].write(self.texture_raw_feeds[index])
                if self.params[index] is not None and self.points_files[index] is not None:
                    self.points_files[index].write(str(self.params[index]))
```

La route pour envoyer le flux d'image est pourvue d'une fonction générateur. Au lieu d'un statement `return`, on utilise à la place `yield`.

Contrairement à une fonction classique avec `return`, une fonction générateur ne va pas garder une liste de valeurs dans la RAM, mais va produire la valeur au moment demandé.

Grâce à cela, chaque appel de la fonction générateur dans une boucle permet de produire une image encodée JPEG à la fois de façon asynchrone/streamée.

Route dynamique pour envoyer les frames générées sous la forme d'un flux d'images continu :

```
@app.route('/video_feed/<int:index>')
def video_feed(index):
    if 0 <= index < len(c.RTSP_URLS):
        def gen_frames(index):
            """
            The function `gen_frames` generates frames from a frame queue and encodes them as JPEG images.

            :param index: The `index` parameter is used to specify the index of the frame queue from which
            frames will be retrieved. It is used to access the `frame_queues` list and retrieve the frame from
            the specified queue
            """

            while True:
                frame = video_processor.get_texture_feed(index)
                if frame is not None:
                    _, buffer = cv.imencode('.jpg', frame) # _ ignore the ret value
                    frame = buffer.tobytes()
                    yield (b'--frame\r\n'
                           b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

        return Response(gen_frames(index),
                        mimetype='multipart/x-mixed-replace; boundary=frame')
    else:
        return "Invalid stream index.", 404
```

Au final, j'affiche ces flux d'images sur ma page grâce à de simples balises HTML `` qui pointent sur les routes dynamiques des flux d'images.

DOSSIER PROFESSIONNEL (DP)

```
<section id="cameras">
    <div class="cameras-container">
        

        
    </div>
</section>
```

Pour les interactions qui nécessitent une communication directe et bidirectionnelle entre le client et le serveur, j'utilise des WebSockets.

Cela permet d'avoir des réponses sur l'état de services, sans avoir à relancer de requête.

Dans le cas d'enregistrements de fichiers vidéo, à partir du flux d'images :

- L'utilisateur déclenche l'enregistrement à distance
- Le serveur notifie l'utilisateur de l'état d'enregistrement (exemples : start / stop)
- Échange de commandes ou de données (exemples : timestamp, nom de fichier, frame)

Front-end :

Établissement d'une connexion WebSocket persistante entre le client et le serveur Flask-SocketIO :

```
let socket = io.connect('http://' + document.domain + ':' + location.port);
```

Envoi d'événements WebSocket au serveur pour l'enregistrement :

```
function startRecord() {
    console.log("start record");
    socket.emit('start recording', {data: 'start record!'});

}

function stopRecord() {
    console.log("stop record");
    socket.emit('stop recording', {data: 'stop record!'});
}
```

DOSSIER PROFESSIONNEL (DP)

Back-end :

Écoute des événements correspondants et lancement des enregistrements

```
@socketio.on('start recording')
def start_recording(msg):
    print('start recording')
    print(msg)
    emit('start recording', broadcast=True)

    timestamp = str(time.time())
    folder = f"./data/{timestamp}"
    os.mkdir(folder)

    video_processor.start_recording(folder)

@socketio.on('stop recording')
def stop_recording(msg):
    print('stop recording')
    print(msg)
    emit('stop recording', broadcast=True)

    video_processor.stop_recording()
```

Je gère la plupart des interactions vidéo, tels que les enregistrements, dans la classe `VideoProcessor`.

DOSSIER PROFESSIONNEL (DP)

Classe `VideoProcessor` avec une partie de ses méthodes :

```
class VideoProcessor:  
    def __init__(self):  
  
        def get_frames(self, index):  
  
        def process_frames(self, index):  
  
        def get_texture_feed(self, index):  
  
        def start_recording(self, folder):  
  
        def stop_recording(self):  
  
        def start_mediapipe_body(self):  
  
        def init_data_support_body(self, pipes, disps):
```

Dans ma classe, j'utilise `threading.Event`, un outil de la bibliothèque standard Python pour synchroniser des threads.

Je génère un “flag” pour gérer mon processus d’enregistrement vidéo sans avoir à bloquer l’application Flask.

Je peux activer ces flags, les désactiver ou attendre qu’ils changent d’état pour déclencher d’autres actions.

Utiliser les flags à la place de simples variables booléennes est indispensables quand on crée un programme avec du multithreading, pour coder de manière “thread-safe”.

DOSSIER PROFESSIONNEL (DP)

Cela permet d'accéder à mes variables sans “race condition” : le risque est que plusieurs threads accèdent en lecture et écriture simultanément à une même ressource (comme un simple booléen), sans coordination.

Flag d'événement - record :

```
self.record = threading.Event()
```

Quand j'appelle la fonction `start_recording`, elle prépare les writers d'OpenCV à écrire des images dans des fichiers vidéos, situés dans des dossiers spécifiques.

J'active ensuite le flag `record` qui indique que l'enregistrement des vidéos est censé commencer à ce moment précis.

```
def start_recording(self, folder):
    for i, camera in enumerate(self.cameras):
        self.writers[i] = cv.VideoWriter(f"{folder}/camera{i}.mp4", cv.VideoWriter_fourcc(*'avc1'), 30.0, (camera.frame_width,
                                                                                                         camera.frame_height))
    self.record.set()
```

Quand le flag est actif, les frames sont écrites dans les fichiers vidéo.

Fonction get_frame

```
def get_frames(self, index):
    while True:
        ret, frame = self.cameras[index].cap.read()
        if ret:
            self.frames[index] = frame
            if not self.processing.is_set():
                self.texture_raw_feeds[index] = self.frames[index]
            if self.record.is_set() or self.record_calib.is_set():
                self.writers[index].write(self.texture_raw_feeds[index])
```

Finalement, de la même manière que la fonction `start_recording` démarre l'enregistrement en activant les writers d'OpenCV, `stop_recording` ferme les writers et les remet à zéro avec le flag `record`.

DOSSIER PROFESSIONNEL (DP)

```
def stop_recording(self):
    self.record.clear()
    for i in range(len(self.cameras)):
        self.writers[i].release()
        self.writers[i] = None
```

Toutes ces interactions nécessitent juste une action de l'utilisateur, tel qu'un clic.

2. Précisez les moyens utilisés :

- **Python** comme langage de programmation back-end
- **Flask** comme framework
- **Flask-SocketIO** pour la communication en socket
- **venv** pour isoler un environnement virtuel pour mon projet, avec des versions spécifiques pour Python et ses packages
- **Jinja** comme moteur de template pour le front-end
- **JavaScript** pour gérer les WebSockets et interactions front-end
- **VSCode** comme IDE / éditeur de code
- **git** comme gestionnaire de contrôle de version
- **GitHub** pour sauvegarder le projet et rendre son code source accessible
- **figma** pour le wireframe et le design des interfaces
- **notion** pour la gestion du projet
- **google agenda** pour l'emploi du temps

DOSSIER PROFESSIONNEL^(DP)

3. Avec qui avez-vous travaillé ?

Avec

- une designer
- une scrum master
- le chef d'entreprise

4. Contexte

Nom de l'entreprise, organisme ou association - V.RTU

Chantier, atelier, service - R&D

Période d'exercice - Du 10/2023 au 11/2024

5. Informations complémentaires (*facultatif*)

V.SION est un système de motion capture servant à entraîner des usagers à effectuer des tâches spécifiques.

V.SION enregistre les positions des humains dans l'espace, sans besoin d'utiliser des combinaisons ou des marqueurs.

DOSSIER PROFESSIONNEL (DP)

Activité-type 1 Développer une application sécurisée

Exemple n°3 - QuickPath

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Projet personnel réalisé en tant qu'étudiant.

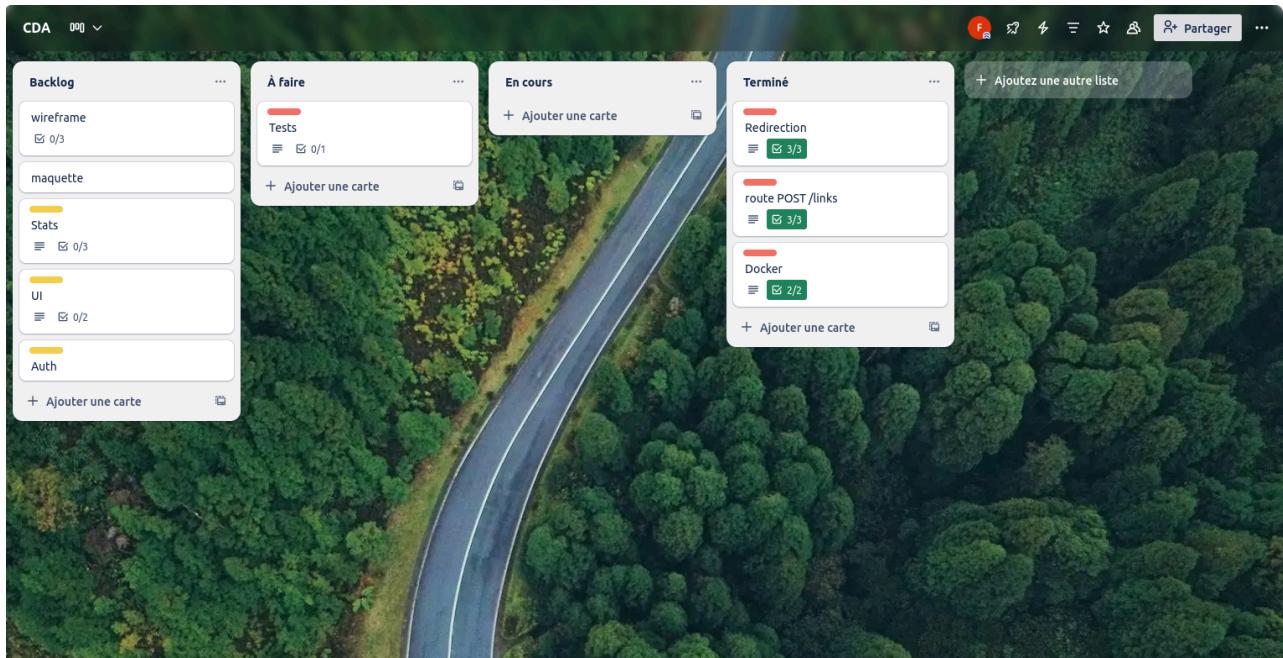
QuickPath est un raccourcisseur d'URL, qui permet à un utilisateur de POST une URL afin d'en générer une plus courte à partir d'un slug unique.

Comme choix de technologie, j'ai opté pour FastAPI, un framework Web python qui se targue, entre autres :

- d'avoir de hautes performances, similaires à node.js et go.
- d'être simple, intuitif et rapide à coder.
- d'avoir moins de bug lors du développement.

La gestion des tâches s'est faite avec Trello, un gestionnaire de projet en ligne.

Tableau kanban sur trello :



DOSSIER PROFESSIONNEL (DP)

Exemple de tâche trello :

En cours ▾

○ route POST /links

+ Ajouter

⌚ Dates

☑ Checklist

& Membres

📎 Pièce jointe

Étiquettes

MUST

+

☰ Description

Modifier

1. takes an url as a parameter
2. link and persist a short url with the origin url
3. return the short url with a 201 code

☑ todo

Masquer les tâches cochées

Supprimer

67%

- Create a short URL/slug from an origin URL
- if origin URL is invalid, get an error message with code 400/422
- ensure slug unicity

Ajouter un élément

L'importance et la priorité des tâches sont définies en se basant sur la **RFC 2119**, qui établit une terminologie normative afin de spécifier le niveau d'exigence de chaque fonctionnalité.

DOSSIER PROFESSIONNEL (DP)

- **MUST** indique une obligation absolue
- **SHOULD** indique une recommandation forte
- **COULD** indique une possibilité facultative

J'ai créé le dépôt git, et je l'ai push sur GitHub.

Pour mes commits, je me base sur <https://www.conventionalcommits.org/en/v1.0.0/>.

Aux prémisses du développement, j'ai installé un environnement virtuel venv, pour télécharger une version de Python récente propre au projet et ses dépendances, le tout isolé du reste de l'ordinateur.

```
python3.12 -m venv .venv
```

```
. .venv/bin/activate
```

Ensuite, j'installe mes dépendances :

```
/CDA/quick-path$ pip install fastapi uvicorn[standard] sqlmodel sqlalchemy pydantic email-validator python-dotenv
```

Et mes dépendances dev :

```
/CDA/quick-path$ pip install pytest pytest-asyncio httpx ruff
```

Pour mettre le tout sur un fichier requirements.txt, qui sert à pip les dépendances à installer quand on utilise pip install.

```
/CDA/quick-path$ pip freeze > requirements.txt
```

J'ai créé des composants métier pour pouvoir gérer la génération de slugs et la création de liens à entrer dans ma base de données.

DOSSIER PROFESSIONNEL (DP)

```
import secrets, string, datetime
from sqlmodel import select, Session
from .models import Link

ALPHABET = string.ascii_letters + string.digits

def generate_slug(session: Session, length: int = 6) -> str:
    while True:
        slug = ''.join(secrets.choice(ALPHABET) for _ in range(length))
        if not session.exec(select(Link).where(Link.slug == slug)).first():
            return slug

def create_link(session: Session, original_url: str, expires_at=None) -> Link:
    slug = generate_slug(session)
    link = Link(slug=slug, original_url=original_url, expires_at=expires_at)
    session.add(link)
    session.commit()
    session.refresh(link)
    return link
```

La fonction `generate_slug` permet de générer d'un identifiant unique court (slug) pour une URL.

Si le slug qu'elle essaie de générer existe déjà en base de données, elle effectue une boucle de génération d'unicité, jusqu'à générer un slug unique.

Cela permet de créer des liens raccourcis uniques, essentiels au fonctionnement de l'application QuickPath.

Comportement métier : redirection d'un lien court

Dans le fichier `main.py` qui gère les routes de FastAPI, j'ai créé plusieurs routes, dont celle de la redirection du slug vers le lien d'origine.

DOSSIER PROFESSIONNEL (DP)

main.py - route GET /{slug}

```
@app.get("/{slug}", name="redirect")
def redirect(slug: str, session: Session = Depends(get_session)):
    link: Link | None = session.exec(select(Link).where(Link.slug == slug)).first()
    if not link:
        raise HTTPException(status_code=404, detail="Lien introuvable")
    if link.expires_at and link.expires_at < datetime.datetime.utcnow():
        raise HTTPException(status_code=410, detail="Lien expiré")
    link.clicks += 1
    link.last_accessed = datetime.datetime.utcnow()
    session.add(link)
    session.commit()
    return RedirectResponse(link.original_url, status_code=301)
```

Elle récupère le lien dans la base de données correspondant au paramètre slug.

Si le lien n'existe pas, le code **404 Not Found** est renvoyé, qui “indique qu'un serveur ne peut pas trouver la ressource demandée” (source : MDN).

Si le lien existe mais qu'il est expiré (la date expires_at est antérieure à la date actuelle), le code **410 Gone** est renvoyé, qui “indique que l'accès à la ressource visée n'est plus disponible sur le serveur d'origine et que cet état est susceptible d'être définitif”. (source : MDN)

Si le lien est valide, l'utilisateur est redirigé avec un code 301 vers l'URL d'origine.

Simultanément, le compteur de clics (`clicks`) est incrémenté et la date du dernier accès (`last_accessed`) est mise à jour, ce qui permet d'enregistrer des statistiques de consultation du lien court.

Ce traitement applique les principes du style défensif, attendus dans le cadre du titre CDA :

- il valide la présence du lien
- il vérifie les conditions d'expiration
- il retourne des erreurs explicites et sécurisées selon les cas
- il assure la traçabilité des accès via la mise à jour atomique des champs de statistiques

Comportement métier : création d'un lien court

Dans le fichier main.py, j'ai défini une route POST /links qui permet à un visiteur de raccourcir une URL longue. L'API retourne un slug unique, associé à cette URL, ainsi qu'un lien court exploitable immédiatement.

DOSSIER PROFESSIONNEL (DP)

main.py - route POST /links

```
@app.post("/links", response_model=LinkOut, status_code=status.HTTP_201_CREATED)
def shorten_link(payload: LinkCreate, session: Session = Depends(get_session)):
    link = create_link(session, str(payload.url), payload.expires_at)
    return {
        "slug": link.slug,
        "short_url": f"{app.url_path_for('redirect', slug=link.slug)}"
    }
```

Lorsqu'un utilisateur soumet une URL, le traitement métier effectué est le suivant :

- La requête attend un objet JSON contenant une URL valide et éventuellement une date d'expiration
- La structure de l'entrée est validée par le modèle `LinkCreate`, qui repose sur le champ `url` de type `HttpUrl` (fourni par `pydantic`), empêchant toute soumission invalide ou malveillante
- Le traitement métier est ensuite délégué à la fonction `create_link()` (dans `crud.py`), qui :
 - génère un slug aléatoire unique (6 à 8 caractères alphanumériques)
 - vérifie qu'il n'est pas déjà utilisé (gestion de collision)
 - crée un objet `Link` et l'enregistre en base avec la date de création, l'URL, le slug et l'éventuelle date d'expiration

Une fois l'objet enregistré, une réponse JSON est renvoyée au client, contenant le slug et l'URL courte complète, avec un statut HTTP 201 Created.

Ce traitement respecte les principes du style défensif, requis dans le cadre du titre CDA :

- Le contrôle strict des données en entrée grâce à `pydantic`, empêchant les URLs mal formées, vides ou dangereuses
- La génération du slug avec gestion des collisions : aucune hypothèse n'est faite sur l'unicité du premier slug généré, une vérification en base est effectuée avant validation
- La logique métier est encapsulée dans une fonction dédiée (`create_link`) pour une meilleure séparation des responsabilités

Toute insertion est effectuée dans une transaction et la session est commitée proprement, évitant les écritures incomplètes.

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

- **Python** comme langage de programmation back-end
- **FastAPI** comme framework
- **SQLite** pour la persistance des données
- **Pydantic** pour la validation des données
- **venv** pour isoler un environnement virtuel pour mon projet en local, au lancement du projet
- **VSCodium** comme IDE / éditeur de code
- **git** comme gestionnaire de contrôle de version
- **GitHub** pour sauvegarder le projet et rendre son code source accessible
- **figma** pour le wireframe et le design des interfaces
- **Trello** pour la gestion du projet

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet.

4. Contexte

Nom de l'entreprise, organisme ou association ➔ **Projet personnel**

Chantier, atelier, service ➔ **Projet personnel**

Période d'exercice ➔ Du 05/2025 au 07/2025

DOSSIER PROFESSIONNEL^(DP)

5. Informations complémentaires (*facultatif*)

QuickPath, projet de raccourcisseur d'URL

DOSSIER PROFESSIONNEL (DP)

Activité-type 2

Concevoir et développer une application sécurisée organisée en couches

Exemple n°1 - Haptify

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Projet d'entreprise effectué dans le cadre de l'alternance. Le projet s'est découpé sur une période de plusieurs mois.

Ce projet collaboratif portait sur la conception et le développement d'une application web permettant la création, la personnalisation et la génération d'effets haptiques.

L'objectif principal est de rendre plus accessible la création d'effets haptiques pour les développeurs et les designers.

Nous avons effectué des sessions de brainstorming pour définir la direction de la conception et du développement de l'application. Le produit doit être :

- accessible (pour le public le plus large)
- générique (et non taillé sur mesure)

Conception de la base de données

J'ai participé à la conception du modèle de données à l'aide d'Amplify Studio, un outil visuel de modélisation et de génération de schémas GraphQL.

DOSSIER PROFESSIONNEL (DP)

Interface d'Amplify Studio

The screenshot shows the Amplify Studio interface for data modeling. On the left, a sidebar lists various services: Home, Manage, Content, User management, File browser, Design, UI Library, Set up (Data, Authentication, Storage, Functions, GraphQL API, REST API, Analytics, Predictions, Interactions, Notifications), Documentation, Support, and Feedback. The main area is titled "Data modeling" and shows a "Deployed" status. It contains several model definitions:

- Plan**: Fields: id (ID), name (String), description (String), tokenLimit (Int). Relationships: User (One to One).
- User**: Fields: id (ID), username (String), email (String), tokenLeft (Int), owner (String).
- FontData**: Fields: family (String), size (Float), weight (String), style (String).
- GradientData**: Fields: colors (String), angle (Float).
- PaddingData**: Fields: top (Float), leading (Float), bottom (Float), trailing (Float).
- AnimationData**: Fields: type (String), duration (Float), delay (Float).
- IconData**: Fields: name (String), color (String), size (Float), position (String).
- ShadowData**: Fields: color (String), radius (Float), x (Float), y (Float).
- ButtonStateData**: Fields: id (ID), backgroundColor (String), gradient (GradientData), textColor (String), font (FontData), borderColor (String), borderWidth (Float).
- AndroidFile**: Fields: id (ID), S3Bucket (String), name (String), prompt (String), updatedAt (AWSDateTime).
- ButtonDesign**: Fields: id (ID), name (String), normalState (ButtonStateData), pressedState (ButtonStateData), disabledState (ButtonStateData), createdAt (AWSDateTime).

At the top right, there are buttons for "Visual editor", "GraphQL schema", "Save and Deploy", and "Local setup instructions". An "Inspector panel" on the right allows selecting a model or relationship to configure properties and authorization rules.

Processus :

- Analyse des besoins fonctionnels issus des user stories pour identifier les entités métier (ex. Plan, User, Stats).
- Modélisation du schéma conceptuel directement dans l'interface visuelle d'Amplify Studio, sous forme d'objets typés avec relations (ex. : Plan avec clé primaire,).
- Définition des liens entre modèles (ex. : relation One to One entre User et Plan) et des règles de sécurité des champs (auth, ownership, read/write).
- Génération automatique de la base de données sous-jacente (Amazon DynamoDB) et de l'API GraphQL compatible avec AppSync.

Accès aux données

L'accès aux données s'effectue via l'API GraphQL générée par Amplify. J'ai utilisé Apollo Client côté front-end dans l'application Next.js pour interagir avec l'API, dans une logique de séparation des responsabilités : l'accès aux données est centralisé dans des hooks.

DOSSIER PROFESSIONNEL (DP)

Processus :

- Génération automatique des queries/mutations/subscriptions GraphQL à partir du modèle Amplify (`amplify codegen`).
- Crédit de composants réutilisables (`useCreateLink`, `useGetLinkBySlug`, `useStats`) avec Apollo Client, encapsulant les appels à l'API.
- Validation des entrées côté client et serveur :
 - Contrôle des formats (email, createdAt, priceId).
 - Nettoyage des entrées pour prévenir les injections.
 - Vérification des formats (ex. : pressedState).
 - Nettoyage des entrées pour prévenir les injections.

Sécurité et robustesse :

- Authentification basée sur Amazon Cognito et contrôle d'accès via les directives `@auth` définies dans le schéma.
- Les données critiques (ex : `User.email`) sont protégées selon les permissions définies.

Subscriptions — Suivre les changements en temps réel

En plus des classiques mutations et queries, GraphQL apporte un autre type d'opération : les subscriptions.

Les subscriptions permettent de réagir en temps réel aux changements en base. Par exemple :

- Afficher en live un nouvel utilisateur inscrit (`onCreateUser`)
- Réagir à une modification d'un plan (`onUpdatePlan`)
- Être notifié d'une suppression (`onDeletePlan`)

Exemple de subscription GraphQL :

```
export const onUpdatePlan = /* GraphQL */ `subscription OnUpdatePlan($filter: ModelSubscriptionPlanFilterInput) {  onUpdatePlan(filter: $filter) {    id    name    description    tokenLimit    createdAt    updatedAt    __typename  }  }`;
```

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

- **AWS Amplify** et **Amplify Studio** pour gérer le back-end
- **GraphQL** pour la gestion des requêtes et des données
- **VSCode** comme IDE / éditeur de code
- **Amplify CLI** pour gérer le développement et le déploiement du back-end en local
- **React** comme librairie JavaScript
- **NextJS** comme Framework Web
- **Stripe** pour les paiements / abonnements
- **figma** pour le wireframe et le design des interfaces
- **notion** pour la gestion du projet
- **google agenda** pour l'emploi du temps

3. Avec qui avez-vous travaillé ?

Avec

- un développeur logiciel alternant
- une designer
- une docteure en informatique
- une doctorante spécialiste en haptique
- un scrum master
- le chef d'entreprise

4. Contexte

Nom de l'entreprise, organisme ou association - V.RTU

Chantier, atelier, service - R&D

Période d'exercice - Du 01/2024 au 02/2025

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL (DP)

Haptify est une application qui permet à ses utilisateurs de générer et personnaliser des effets haptiques.

Les effets haptiques sont des effets qui permettent aux utilisateurs d'avoir des retours de sensations par le sens du toucher.

Les utilisateurs connectés ont un nombre de tokens limité, selon leur abonnement, pour générer des effets haptiques. En tapant un prompt, on appelle l'API qui utilise des IA génératives pour créer des effets spécifiques.

Activité-type 2

Concevoir et développer une application sécurisée organisée en couches

Exemple n°2 - VSION

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Projet d'entreprise effectué dans le cadre de l'alternance. Le projet s'est découpé sur une période de plusieurs mois.

J'ai travaillé de concert avec la designer sur figma pour créer les interfaces, tels que des boutons d'enregistrement et des fiches de travail.

Comme choix de technologie, j'ai opté pour Flask, un framework Web Python avec Flask-SocketIO, une librairie associée au framework permettant la communication avec des websockets dans les applications.

Ayant déjà des compétences et une certaine expérience dans le web, le développement d'interface m'a semblé plus simple et adaptable que l'existant en Qt.

De plus, pour le développement futur, une architecture client serveur m'a paru pertinente, pour que la responsabilité de calcul ne soit pas cantonnée sur l'appareil du technicien, et qu'il puisse être éventuellement utilisé par plusieurs techniciens à la fois.

J'ai participé au maquettage de l'application conjointement avec la designer de l'entreprise.

Maquette design de calibration avec onboarding tutorial :

DOSSIER PROFESSIONNEL (DP)

Fiche de travail

Capture

Calibration

Mr Laurent

Caméra 1

Caméra 2

Caméra 3

Caméra 4

Caméra 5

Caméra 6

Après avoir vérifié la visibilité du damier, cliquer sur "Vérifier le damier"

< Suivant >

Vérifier le damier

Lancer la calibration

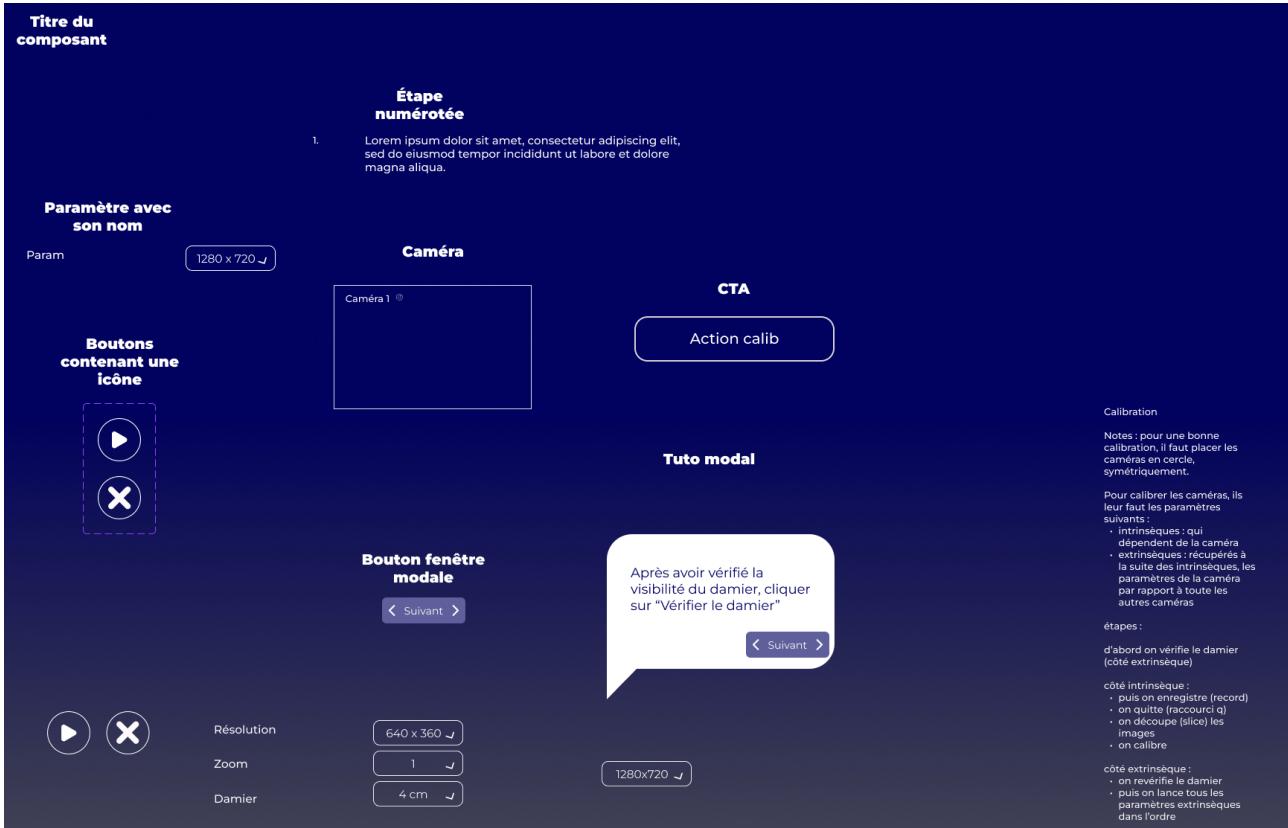
Résolution: 1280 x 720

Côté d'une dalle de damier (cm): 11.5

Nombre colonnes x Nombre lignes: 9 x 6

DOSSIER PROFESSIONNEL (DP)

Différents composants pour la calibration :



User story : l'utilisateur calibre les caméras.

Pour pouvoir utiliser le programme correctement, l'utilisateur doit calibrer 6 caméras positionnées de manière symétrique autour du sujet.

Pour ce faire, il montre un damier à chaque caméra et enregistre le tout avant de lancer les étapes de calibration.

Des paramètres peuvent être extraits à partir des images capturées pour régler les caméras.

User story : l'utilisateur enregistre ses mouvements en effectuant une tâche spécifique.

L'utilisateur, un employé expérimenté, montre la bonne manière d'effectuer une tâche pour permettre l'entraînement d'autres employés.

DOSSIER PROFESSIONNEL (DP)

2. Précisez les moyens utilisés :

- **Python** comme langage de programmation back-end
- **Flask** comme framework
- **Flask-SocketIO** pour la communication en socket
- **venv** pour isoler un environnement virtuel pour mon projet, avec des versions spécifiques pour Python et ses packages
- **Jinja** comme moteur de template pour le front-end
- **JavaScript** pour gérer les WebSockets et interactions front-end
- **VSCode** comme IDE / éditeur de code
- **git** comme gestionnaire de contrôle de version
- **GitHub** pour sauvegarder le projet et rendre son code source accessible
- **figma** pour le wireframe et le design des interfaces
- **notion** pour la gestion du projet
- **google agenda** pour l'emploi du temps

3. Avec qui avez-vous travaillé ?

J'ai travaillé avec :

- une designer
- une scrum master
- le chef d'entreprise

4. Contexte

Nom de l'entreprise, organisme ou association - V.RTU

Chantier, atelier, service - R&D
Période d'exercice - Du 10/2023 au 11/2024

DOSSIER PROFESSIONNEL (DP)

5. Informations complémentaires (facultatif)

VSION est un système de motion capture servant à entraîner des usagers à effectuer des tâches spécifiques. VSION enregistre les positions des humains dans l'espace, sans besoin d'utiliser des combinaisons ou des marqueurs.

Activité-type 3 Préparer le déploiement d'une application sécurisée

Exemple n°1 - Haptify

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Projet d'entreprise effectué dans le cadre de l'alternance. Le projet s'est découpé sur une période de plusieurs mois.

Ce projet collaboratif portait sur la conception et le développement d'une application web permettant la création, la personnalisation et la génération d'effets haptiques.

L'objectif principal est de démocratiser la création d'effets haptiques pour les développeurs et les designers, grâce à une UI et UX intuitives.

J'ai testé des routes d'API avec Hoppscotch, un écosystème pour développer et tester des API.

Hoppscotch est une alternative open source à d'autres outils de développement d'API comme Postman et Insomnia.

Interface de Hoppscotch

The screenshot shows the Hoppscotch interface with the following details:

- Header Bar:** Shows "HOPPSSCOTCH" on the left, search bar with "Chercher" and "Ctrl + K" shortcut, and a user icon on the right.
- Request List:** A horizontal list of requests:
 - GET Untitled
 - POST axum
 - GET Actix
 - POST Quick-Path
 - GET QuickPath links/...
- Current Request:** A detailed view of a GET request to "http://localhost:3000/api/checkout_sessions/price_IQQV3DC7zh3bV3wj663dCPQ?session_id=cs_test_a1Xy0g8Qq8xGUsjVYZccdZJfb4wKy".
 - Method:** GET
 - URL:** http://localhost:3000/api/checkout_sessions/price_IQQV3DC7zh3bV3wj663dCPQ?session_id=cs_test_a1Xy0g8Qq8xGUsjVYZccdZJfb4wKy
 - Buttons:** Envoyer (Send) and Sauvegarder (Save).
- Request Details:** Tabbed view showing "Paramètres", "Corps", "En-têtes", "Autorisation", "Script de pré-requête", and "Tests". The "En-têtes" tab is selected, showing a table with one row: "En-tête 1" with value "Valeur 1".
- Right Sidebar:** Includes environment selection, file management, and other tool icons.

DOSSIER PROFESSIONNEL (DP)

Le déploiement de l'application s'est fait avec amplify. Avec des noms de domaines haptify achetés, j'ai paramétré la redirection vers haptify.io.

Afin d'assurer la qualité du code avant chaque déploiement, j'ai mis en place une phase d'analyse statique du code avec ESLint.

Cette étape, intégrée au processus d'intégration continue, permettait de détecter et corriger automatiquement les erreurs de style ou de syntaxe dès la phase de développement. Le linter était exécuté systématiquement via une GitHub Action, déclenchée à chaque déploiement.

Le déploiement de l'application a été automatisé avec AWS Amplify.

Grâce au fichier amplify.yml, qui décrit les différentes étapes de build, de test et de déploiement, nous avons pu regrouper

- l'installation des dépendances
- l'exécution d'ESLint
- la commande de build propre à notre stack

Une fois le build validé, l'application était automatiquement publiée sur les environnements configurés.

En complément, j'ai configuré le nom de domaine personnalisé en pointant le DNS vers le domaine principal haptify.io, et activé le chiffrement HTTPS pour sécuriser les échanges.

2. Précisez les moyens utilisés :

- **godaddy** pour la gestion de domaine
- **Amazon Amplify Hosting** pour l'hébergement et la CI intégrée
- fichier **amplify.yml** pour la gestion du build et de la CI
- **Amazon Route 53** pour la gestion des domaines personnalisés
- **git** comme logiciel de contrôle de version
- **GitHub** pour l'hébergement du code source
- **TypeScript** pour la détection des erreurs, la robustesse et la maintenabilité du code
- **ESLint** pour la détection des erreurs et les bonnes pratiques de code
- fichier **.env** pour la gestion des variables d'environnement

3. Avec qui avez-vous travaillé ?

DOSSIER PROFESSIONNEL (DP)

Avec

- un développeur logiciel alternant
- une designer
- une docteure en informatique
- une doctorante spécialiste en haptique
- un scrum master
- le chef d'entreprise

4. Contexte

Nom de l'entreprise, organisme ou association - V.RTU

Chantier, atelier, service - R&D

Période d'exercice - Du 01/2024 au 02/2025

5. Informations complémentaires (facultatif)

Haptify est une application qui permet à ses utilisateurs de générer et personnaliser des effets haptiques.

Les effets haptiques sont des effets qui permettent aux utilisateurs d'avoir des retours de sensations par le sens du toucher.

Les utilisateurs connectés ont un nombre de tokens limité, selon leur abonnement, pour générer des effets haptiques. En tapant un prompt, on appelle l'API qui utilise des IA génératives pour créer des effets spécifiques.

DOSSIER PROFESSIONNEL (DP)

Activité-type 3 Préparer le déploiement d'une application sécurisée

Exemple n°2 - VSION

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Projet d'entreprise effectué dans le cadre de l'alternance. Le projet s'est découpé sur une période de plusieurs mois.

Pour le déploiement, plutôt que de déployer sur un serveur distant, nous avons opté pour un modèle de distribution via un exécutable compilé sur un poste spécifique.

J'ai compilé le tout avec Cython et PyInstaller. Cela permet de simplifier la distribution à des utilisateurs non développeurs et d'assurer une meilleure protection du code source métier.

Pendant le développement, j'ai mis de simples instructions de compilation dans le fichier README.

1. If you added a python file, add them as extensions in `setup.py`
2. If any file is added or updated, compile the modules in cython code: `python setup.py build_ext --inplace`
3. Compile the app: `pyinstaller app.spec`

Analysons un peu :

Pour commencer, tout nouveau fichier python doit être ajouté en tant qu'extension à convertir pour cython, comme on peut le voir dans l'image plus bas.

Ensuite, la ligne de commande dans l'étape 2 appelle le fichier `setup.py` avec `python` pour transformer le code en librairie/extension native, dans le même dossier source.

Le fichier `setup.py` est un fichier que j'ai créé pour gérer le build du programme, en utilisant cython.

DOSSIER PROFESSIONNEL (DP)

setup.py

```
from distutils.core import setup
from distutils.extension import Extension
from Cython.Build import cythonize

ext_modules = cythonize(
    [
        Extension("constants", ["constants.py"]),
        Extension("config", ["config.py"]),
        Extension("models", ["models.py"]),
        Extension("utils_db", ["utils_db.py"]),
        Extension("utils_app_debugger", ["utils_app_debugger.py"]),
        Extension("utils_3d_reconstruct", ["utils_3d_reconstruct.py"]),
        Extension("utils_display", ["utils_display.py"]),
        Extension("utils_mediapipe", ["utils_mediapipe.py"]),
        Extension("utils_work_card", ["utils_work_card.py"]),
        Extension("calibration_intrinsic", ["calibration_intrinsic.py"]),
        Extension("calibration_extrinsic", ["calibration_extrinsic.py"]),
        Extension("CameraHandler", ["CameraHandler.py"]),
        Extension("VideoProcessor", ["VideoProcessor.py"]),
    ],
    build_dir="build_cythonize",
    compiler_directives=dict(
        always_allow_keywords=True, # Allow keyword arguments in pure Python functions
        language_level=3, # Source code is always Python 3
    )
)

setup(
    name = 'f-vision',
    ext_modules = ext_modules
)
```

La fonction `cythonize()` de cython convertit les fichiers .py en .c.

Quant à `setup()`, elle compile ces fichiers en extensions natives (.pyd sur Windows, .so sur Linux/macOS).

Vient ensuite la troisième et dernière étape, la compilation en un exécutable.

Pour cela, la ligne `pyinstaller app.spec` utilise au préalable un fichier .spec généré à l'aide de `pyi-makespec`, de la suite d'outils de pyinstaller.

La commande d'origine était `pyi-makespec --onefile app.py`.

Cela m'a permis de générer un fichier .spec qui indique à pyinstaller de créer un répertoire de distribution qui contient l'exécutable et éventuellement les librairies/extensions dynamiques.

DOSSIER PROFESSIONNEL (DP)

Dossier de build :

```
└─ build_cythonize
    └─ CameraHandler.c
    └─ CameraHandler.cp310-win_amd64.pyd
    └─ constants.c
    └─ constants.cp310-win_amd64.pyd
    └─ utils_3d_reconstruct.c
    └─ utils_3d_reconstruct.cp310-win_amd64.pyd
    └─ utils_app_debugger.c
    └─ utils_app_debugger.cp310-win_amd64.pyd
    └─ utils_display.c
    └─ utils_display.cp310-win_amd64.pyd
    └─ utils_mediapipe.c
    └─ utils_mediapipe.cp310-win_amd64.pyd
    └─ VideoProcessor.c
    └─ VideoProcessor.cp310-win_amd64.pyd
    └─ work_card_utils.c
    └─ work_card_utils.cp310-win_amd64.pyd
```

Fichier app.spec :

```
# -*- mode: python ; coding: utf-8 -*-

import glob
from PyInstaller.utils.hooks import collect_data_files

pyd_files = [(f, '.') for f in glob.glob('build_cythonize/*.pyd')]
mediapipe_data = collect_data_files('mediapipe')

flask_data_files = [
    ('.\\"templates"', 'templates'),
    ('.\\"static"', 'static'),
    ('.\\"data\\work_cards"', 'data/work_cards')
]

all_data_files = mediapipe_data + flask_data_files

a = Analysis(
    ['app.py'],
    pathex=[],
    binaries=pyd_files,
    datas=all_data_files,
    hiddenimports=[],
    hookspath=[],
    hooksconfig={},
    runtime_hooks=[],
    excludes=[],
    noarchive=False,
)
pyz = PYZ(a.pure)

exe = EXE(
    pyz,
    a.scripts,
    a.binaries,
    a.datas,
    [],
    name='app',
    debug=False,
    bootloader_ignore_signals=False,
    strip=False,
    upx=True,
    upx_exclude=[],
    runtime_tmpdir=None,
    console=True,
    disable_windowed_traceback=False,
    argv_emulation=False,
    target_arch=None,
    codesign_identity=None,
    entitlements_file=None,
)
```

DOSSIER PROFESSIONNEL (DP)

Tout cela permet d'avoir un exécutable fonctionnel et livrable.

2. Précisez les moyens utilisés :

- **Cython** pour optimiser les performances et obfuscuer le code
- **PyInstaller** pour créer un exécutable livrable et obfuscuer le code
- **pyi-makespec** pour créer un fichier de spécification pour la compilation
- **Python** comme langage de programmation back-end
- **Flask** comme framework
- **Flask-SocketIO** pour la communication en socket
- **venv** pour isoler un environnement virtuel pour mon projet, avec des versions spécifiques pour Python et ses packages
- fichier **README.md** pour les instructions d'installation et de compilation

3. Avec qui avez-vous travaillé ?

J'ai réalisé seul cette partie.

4. Contexte

Nom de l'entreprise, organisme ou association - V.RTU

Chantier, atelier, service - R&D

Période d'exercice - Du 10/2023 au 11/2024

DOSSIER PROFESSIONNEL^(DP)

5. Informations complémentaires (*facultatif*)

VSION est un système de motion capture servant à entraîner des usagers à effectuer des tâches spécifiques. VSION enregistre les positions des humains dans l'espace, sans besoin d'utiliser des combinaisons ou des marqueurs.

DOSSIER PROFESSIONNEL (DP)

Activité-type 3 Préparer le déploiement d'une application sécurisée

Exemple n°3 - QuickPath

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Projet personnel réalisé en tant qu'étudiant.

QuickPath est un raccourcisseur d'URL, réalisé en Python avec le framework FastAPI.

J'ai mis en place la conteneurisation avec Docker, avec pour objectifs :

- garantir la portabilité de l'application quel que soit l'environnement d'exécution
- faciliter les étapes de test, build et déploiement dans une chaîne CI/CD
- permettre une mise en production plus fiable et reproductible.

J'ai rédigé un Dockerfile avec une approche multi-stage, séparant la phase d'installation des dépendances de celle de lancement. Ce procédé optimise la taille de l'image et renforce la sécurité.

Dockerfile :

```
# Stage 1 : build deps
FROM python:3.12-slim AS builder
WORKDIR /app
COPY requirements.txt .
RUN pip install --user -r requirements.txt

# Stage 2 : runtime
FROM python:3.12-slim
WORKDIR /app
ENV PYTHONPATH=/app
COPY --from=builder /root/.local /root/.local
COPY ./app ./app
EXPOSE 8000
CMD ["python", "-m", "uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Pour automatiser le lancement de l'ensemble des composants du projet, j'ai créé un fichier docker-compose.yml.

J'ai mis en place des tests pour une couverture >80%.

Le plan de tests couvre l'ensemble des fonctionnalités clés du service :

- création de liens valides et gestion des erreurs de validation
- unicité des slugs même en cas de collision
- redirections HTTP (301 ou 410 selon les cas)
- traitement des slugs inexistants (404)
- gestion du compteur de clics et de la date du dernier accès

DOSSIER PROFESSIONNEL (DP)

Chaque fonctionnalité fait l'objet d'un test automatisé, exécuté en environnement isolé grâce à une base de données temporaire, initialisée pour chaque session de test.

J'ai utilisé pytest-asyncio pour gérer les appels asynchrones de FastAPI, et ASGITransport pour simuler les requêtes HTTP sans nécessiter de serveur externe.

J'ai également créé des fixtures pour préparer et fournir des ressources nécessaire à mes tests (e.g. création de tuples dans la base de données).

Une fixture client instancie dynamiquement un client HTTP couplé à une session de base de données injectée via la dépendance `get_session`.

Une autre fixture session assure l'utilisation d'une base SQLModel en mémoire pour des tests reproductibles et rapides. Enfin, un hook de session `@pytest.fixture(scope="session", autouse=True)` crée et supprime les tables entre les sessions.

Exemple de test - création de lien

```
@pytest.mark.asyncio
async def test_create_link_success(client: AsyncClient):
    resp = await client.post("/links", json={"url": "https://example.com"})
    assert resp.status_code == 201
    body = resp.json()
    assert body["slug"].isalnum() and 5 < len(body["slug"]) <= 8
    assert body["short_url"].endswith(body["slug"])
```

Exemple de fixture - création et destruction des tables de la base de données

```
@pytest.fixture(scope="session", autouse=True)
def _prepare_database():
    """Create all tables once per test session, drop them afterwards."""
    SQLModel.metadata.create_all(bind=engine)
    yield
    SQLModel.metadata.drop_all(bind=engine)
```

DOSSIER PROFESSIONNEL^(DP)

2. Précisez les moyens utilisés :

- **python** comme langage de programmation
- **FastAPI** comme framework
- **pytest** pour les tests de l'application
- **httpx et AsyncClient** pour run les tests
- **Docker** pour conteneuriser l'application

3. Avec qui avez-vous travaillé ?

J'ai réalisé ce projet seul.

4. Contexte

Nom de l'entreprise, organisme ou association ▶ **Projet personnel**

Chantier, atelier, service ▶ **Projet personnel**

Période d'exercice ▶ **Du 04/25 au 07/25**

DOSSIER PROFESSIONNEL^(DP)

5. Informations complémentaires (*facultatif*)

QuickPath, projet de raccourcisseur d'URL

DOSSIER PROFESSIONNEL (DP)

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
Développeur web et web mobile	La Plateforme	08/2023
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

DOSSIER PROFESSIONNEL ^(DP)

DOSSIER PROFESSIONNEL (DP)

Déclaration sur l'honneur

Je soussigné

Félix HAUGER

,

déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis l'auteur des réalisations jointes.

Fait à *MARSEILLE*

le *15 juillet 2025*

pour faire valoir ce que de droit.

Signature :



DOSSIER PROFESSIONNEL (DP)

DOSSIER PROFESSIONNEL (DP)

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé

DOSSIER PROFESSIONNEL^(DP)

ANNEXES

(Si le RC le prévoit)

