Portfolio — Documentation unifiée

Généré le 2025-11-07 04:26

---

project-presentation.md

Présentation du projet

Objectifs

- Valoriser mon profil et les compétences que je développe en alternance.
- Donner un aperçu concret des projets que j'ai menés et des stacks que je maîtrise.
- Offrir un canal de contact professionnel avec suivi par email.
- Centraliser tous les documents utiles aux recruteurs et partenaires.

Public cible

Je m'adresse aux responsables techniques et recruteurs qui recherchent un alternant fullstack capable de livrer une application moderne (front + back).

Périmètre fonctionnel

1. **Page d'accueil** : hero, navigation rapide, accroche.
2. **À propos** : parcours, compétences, langues.
3. **Points forts** : carousel interactif.
4. **Projets** : fiches détaillées (stack, durée, rôle).
5. **Galerie** : modal plein écran.
6. **Documentation** : centre de ressources `/docs/*` (présentation, tests, analyses).
7. **Contact** : formulaire relié à un service externe compatible GitHub Pages.

Architecture technique

- Nuxt 3, Vue 3, Tailwind 4 (plugin Vite).
- API Nitro (`/api/contact`) activable sur un hébergement Node pour stocker les messages et déclencher Resend/SMTP.
- Intégration formulaire externe via `NUXT_PUBLIC_CONTACT_FORM_ENDPOINT` (Formspree, EmailJS…) lorsque j'héberge en statique.
- Déploiement cible : Vercel (Resend + `/tmp/portfolio-data`) ou Render/Railway pour l'API persistante.

Conformité aux attendus

- **Accroche & identité** : Hero avec ma photo, ma phrase d'introduction, ma localisation et un lien direct vers mon CV (`components/HeroSection.vue`).
- **Coordonnées & contact** : email, téléphone et formulaire connecté à Resend ou à `/api/contact` (`components/ContactForm.vue`).
- **Travaux regroupés** : projets détaillés (stack, rôle, résultats) + galerie modale (`components/ProjectsList.vue` , `components/ImageModal.vue`).
- **Suivi compétences RNCP** : matrice dédiée qui relie chaque bloc à mes réalisations (`components/CompetenciesSection.vue` , tableau ci-dessous).

Approche hybride

- **Présentation** : sections publiques Hero, À propos, Points forts, Projets et Galerie qui condensent mon meilleur profil pour les recruteurs.
- **Apprentissage** : section Documentation où je décris ma démarche, mes tests et mon suivi RNCP.
- **Mise à jour** : backlog Trello + revues hebdomadaires (cf. `docs/qa-monitoring.md` , `docs/improvement-proposals.md`).

Analyse des réalisations et progression

Bloc RNCP	Compétences clés travaillées	Réalisations & preuves	Progression / axes
Bloc 1 – Interfaces	Maquettage responsive, accessibilité, intégrations Vue/Nuxt	TopHardware (catalogue Next.js + Stripe) ; Propulse by CA (dashboard Nuxt + design system)	Poursuivre les tests UI automatisés (Playwright) et livrer une version EN du site.
Bloc 2 – Back-end	Modélisation SQL/NoSQL, API sécurisées, documentation	Propulse by CA (API Discourse + migrations AWS) ; TopHardware (API Symfony, OAuth)	Renforcer la couverture d'intégration et publier la documentation Swagger/Postman.
Bloc 3 – Ops	CI/CD, monitoring, plan de	Portfolio (plan QA, SQLite, sauvegarde messages) ; Propulse by CA (mise en prod	Industrialiser la pipeline GitHub Actions et

& QA	tests, runbooks	Discourse)	suivre les métriques GA4.
Bloc RNCP	**Compétences clés travaillées**	**Réalisations & preuves**	**Progression / axes**
Bloc 4 – Collaboration	Auto-évaluation, communication agile, partage connaissances	Portfolio (docs d'analyse, ICE backlog) ; Propulse by CA (rites Scrum, démos)	Documenter davantage les retours utilisateurs et préparer des rétrospectives trimestrielles.

Documentation & conformité

- **Conformité aux attendus** : l'ensemble des éléments demandés (accroche, contacts, projets, CV, analyse compétences) est présent sur la homepage et dans `/docs`.
- **Approche hybride** : la face présentation sert les recruteurs, la face apprentissage expose mes preuves, tests et plans d'évolution.
- **Analyse RNCP** : le tableau ci-dessus relie chaque bloc à des réalisations concrètes avec des axes de progression.

Maquette

Une maquette SVG est disponible dans `public/wireframe.svg` et accessible sur le site via `/docs/project-presentation`.

---

technical-documentation.md

Documentation technique

Installation

```
npm install
cp .env.example .env # puis renseigner les valeurs SMTP si l'API contact est utilisée
npm run dev
```

Scripts

- `npm run dev` : serveur de développement (http://localhost:3000).
- `npm run build` : build production (`.output/`).
- `npm run preview` : prévisualisation du build.
- `npm test` : suites unitaires (Vitest) + rapport de couverture.

Variables d'environnement

Variable	Description
`MAIL_HOST`	Hôte SMTP
`MAIL_PORT`	Port SMTP
`MAIL_USER`	Identifiant SMTP
`MAIL_PASS`	Mot de passe SMTP
`MAIL_RECIPIENT`	Destinataire des messages (par défaut `MAIL_USER`)
`RESEND_API_KEY`	Clé API Resend pour déclencher l'envoi via HTTPS
`RESEND_FROM`	Adresse d'expéditeur Resend (ex : `Portfolio <contact@exemple.com>`)
`RESEND_TO`	Destinataire lorsque Resend est utilisé (défaut : `MAIL_RECIPIENT`)
`NUXT_DATA_DIR`	(Optionnel) Répertoire cible pour `data/app.db` (défaut : `data/`, fallback `/tmp/portfolio-data` sur Vercel)

Base de données & services

- **SQLite** : initialisé automatiquement dans `data/app.db` via `better-sqlite3`.

    - Table `contact_messages` : `id`, `name`, `email`, `message`, `created_at`.
    - Accessible via `server/services/contactService.ts`.
- **Services** :
    - `contactService` : validation des inputs, persistance SQLite, envoi SMTP.

Tests

- Framework : Vitest (`npm test`).
- Périmètre actuel : validation serveur du formulaire de contact (TC01/02).
- Rapport HTML : `coverage/index.html`.

Points d'extension

- Ajouter un CMS headless : créer un plugin Nuxt et remplacer les données statiques des composants.
- CI/CD : ajouter un workflow `.github/workflows/deploy.yml` déclenchant le build.

Dépendances clés

- `@nuxt/image` : optimisation & responsive images.
- `@tailwindcss/vite` : génération des utilitaires Tailwind 4.
- `nodemailer` : envoi des emails de contact.
- `better-sqlite3` : persistance locale des messages.
- `vitest` / `@vitest/coverage-v8` : tests unitaires + couverture.
- `vue` / `nuxt` : framework principal.

---

specifications-techniques.md

Spécifications techniques

Stack

- Nuxt 3.16 (Vue 3, Nitro)
- Tailwind CSS 4 via `@tailwindcss/vite`
- Service de formulaires externe configurable (`NUXT_PUBLIC_CONTACT_FORM_ENDPOINT`) pour GitHub Pages
- (Optionnel) Nodemailer + Better SQLite 3 pour les déploiements Node
- TypeScript dans les composants complexes

Structure des composants

Composant	Rôle	Particularités
`HeroSection.vue`	Accroche, informations clés, CTA	Image statique optimisée (balise `<img>`)
`AboutSection.vue`	Parcours, compétences, langues	Mise en page responsive en grille
`Carousel.vue`	Mise en avant des points forts	Navigation prév/next, computed item
`ProjectsList.vue`	Catalogue des projets	Données typées, badges stack
`ImageModal.vue`	Galerie modale	Utilise Teleport + accessibilité ESC
`ContactForm.vue`	Formulaire de contact	Utilise `useContactForm` (fetch vers endpoint externe)

API

Formulaire de contact (GitHub Pages)

- **Envoi** : `fetch` JSON vers l'URL exposée par `NUXT_PUBLIC_CONTACT_FORM_ENDPOINT` (Formspree, EmailJS, etc.).
- **Validation** : trim, format email, longueur minimale (10 caractères) dans `useContactForm`.
- **Retour** : message de confirmation retourné par le service ou fallback configuré (`NUXT_PUBLIC_CONTACT_FORM_SUCCESS_MESSAGE`).

Option Nitro `POST /api/contact` (déploiement Node)

- **Entrée** : `{ name, email, message }`
- **Validation** : assurée côté serveur (`validateContactPayload`)
- **Persistance** : insertion dans `contact_messages` (SQLite)
- **Notification** : SMTP via Nodemailer (`MAIL_*`)
- **Sortie** : `{ success: true, message, contactId }` ou erreur 400/500

Sécurité

- Secrets en variables d'environnement (`MAIL_*` , `NUXT_PUBLIC_*`).
- Les messages envoyés via GitHub Pages transitent par un service tiers (Formspree, EmailJS) bénéficiant de leurs protections anti-spam.
- En déploiement Node, le module `contactService` valide et assainit les données avant insertion dans SQLite.
- Prévoir reCAPTCHA v3 ou un honeypot si la volumétrie augmente.

Performance

- Lazy loading implicite via Nuxt.
- Images locales servies par `@nuxt/image` (optimisation automatique).
- Build Nuxt optimisé (44 kB JS côté client hors dépendances).

---

user-guide.md

Guide utilisateur

1. Navigation rapide

1. Utilisez le menu fixe pour accéder directement aux ancres (`À propos` , `Projets` , `Points forts` , `Galerie` , `Contact`).
2. Le bouton « Découvrir mes projets » du Hero scrolle vers la liste des projets.
3. La galerie ouvre une modale plein écran pour examiner les visuels haute fidélité.

2. Contacter Ricardo

1. Rendez-vous dans la section « Me contacter ».
2. Renseignez votre nom, votre email professionnel et votre message (≥ 10 caractères).
3. Cliquez sur « Envoyer » : un message de confirmation ou d'erreur s'affiche. Les réponses sont données sous 48 h maximum.

> Pour un contact direct, utilisez `ricardo.chaves-rodrigues@epitech.eu` présent dans l'en-tête.

3. Accès à la documentation

- Ouvrez le lien « Docs » du menu ou rendez-vous sur `/docs` .
- Sélectionnez le document voulu (Présentation, Plan de tests, Guide, Améliorations).
- Copiez l'URL directe pour partager la ressource aux partenaires.

4. Support & feedback

- Pour signaler un problème ou proposer une amélioration, envoyez un email à `ricardo.chaves-rodrigues@epitech.eu` .
- Précisez le contexte (navigateur, appareil, date), les étapes avant le bug et joignez une capture d'écran si possible.

---

improvement-proposals.md

Propositions d'améliorations

1. Monitoring & retours utilisateurs

- Collecter systématiquement les messages entrants (stockage SQLite + Resend) pour identifier les points d'intérêt.
- Suivre les sessions via Google Analytics 4 (tableau récapitulatif prévu).
- Capitaliser les feedbacks recueillis lors des entretiens recruteurs et mettre à jour le backlog associé.

2. Ergonomie & accessibilité

- Contraste conforme WCAG AA (audits Lighthouse réguliers).
- Navigation clavier : focus visible, modale fermée via ESC.
- À prévoir : bouton « Aller au contenu » et traductions des textes alternatifs en anglais.

3. Pistes d'évolution produit

1. **Tests automatisés** : compléter Vitest avec Vue Testing Library puis Playwright pour les parcours critiques.
2. **CI/CD** : pipeline GitHub Actions déclenchant `npm run build` + déploiement Render/Vercel.
3. **CMS** : brancher un headless CMS (Contentful, Strapi) afin d'éditer les projets sans redeploiement.
4. **Internationalisation** : proposer l'interface en FR/EN/PT avec un switcher.
5. **Analytics** : activer GA4 ou Plausible pour suivre l'engagement et tracer les conversions.
6. **Accessibilité avancée** : mode clair/sombre, navigation au clavier dans le carousel via flèches.
7. **SEO** : enrichir les meta tags (Open Graph, Twitter Cards) et générer un sitemap.

4. Priorisation (ICE)

1. Mise en place du suivi analytique (impact fort, effort modéré).
2. Automatisation QA (réduit les régressions).
3. Refonte du module projets avec CMS.
4. Internationalisation complète.

---

test-plan.md

Plan de tests détaillé

Aperçu

- **Portée** : fonctions critiques (contact, navigation, carousel, galerie).
- **Responsable** : Ricardo Chaves Rodrigues.
- **Cycle** : à chaque itération ou livraison majeure.

Matrice de tests

ID	Fonction	Type	Attendu
TC01	Formulaire contact	Fonctionnel	Message de succès, envoi accepté par le service externe
TC02	Formulaire contact	Fonctionnel (erreur)	Message d'erreur si service indisponible ou endpoint absent
TC03	Formulaire contact	Validation	Empêcher email vide/format invalide
TC20	Navigation anchors	UI/UX	Scroll fluide vers la section
TC21	Carousel	UI	Items pivotent correctement
TC22	Modal galerie	Accessibilité	Fermeture via clic overlay et touches

Automatisation

- **Stack** : Vitest (tests unitaires sur `contactService` côté serveur).
- **Commandes** : `npm test` (exécute l'ensemble des suites).
- **Couverture actuelle** :
  - `TC01` / `TC02` : validation du formulaire (`contactService` pour l'option serveur, logique partagée avec `useContactForm`).
- **Rapports** : disponibles dans `coverage/` (générés via Vitest).

Stratégie d'exécution

1. Exécuter les tests manuels à chaque fin d'itération.
2. Documenter les résultats dans `docs/test-report.md` .
3. Étendre l'automatisation vers les scénarios UI (TC21, TC22) via des tests end-to-end (Playwright).

Outils

- Lighthouse pour l'accessibilité.
- Chrome DevTools (mobile viewport, throttling).
- Vitest (à intégrer) pour tester les composants.

---

test-report.md

Journal d'exécution des tests

Date	Version	Tests exécutés	Résultat	Notes
2025-02-__	v1.0	TC01, TC21	✅ / ⚠	...
2025-10-26	v1.1	TC01, TC02	✅	Suite automatisée Vitest (`npm test`).

> Renseigner le tableau après chaque session de tests.

---

deployment-guide.md

Guide de déploiement

Pré-requis

- Compte Render / Railway / VPS Node.js **ou** accès à un serveur Docker (VM, VPS, etc.)
- Node 20 LTS (si déploiement sans conteneur)
- Endpoint de formulaire tiers (Formspree, EmailJS, etc.) et variable `NUXT_PUBLIC_CONTACT_FORM_ENDPOINT`
- Variables d'environnement SMTP configurées (`MAIL_HOST`, `MAIL_PORT`, `MAIL_USER`, `MAIL_PASS`, `MAIL_RECIPIENT`) si l'API contact côté serveur est activée. Vous pouvez aussi utiliser Resend (`RESEND_API_KEY`, `RESEND_FROM`, `RESEND_TO`) pour un envoi HTTP compatible serverless.
- Optionnel : `NUXT_DATA_DIR` pour définir un dossier d'écriture (utiliser `/tmp/portfolio-data` sur Vercel ou plateformes read-only).

Déploiement GitHub Pages

1. Créer un formulaire sur Formspree (ou service équivalent) et récupérer l'URL d'endpoint.
2. Définir la variable `NUXT_PUBLIC_CONTACT_FORM_ENDPOINT` (Secrets GitHub ou `.env` local avant le build).
3. Facultatif : personnaliser `NUXT_PUBLIC_CONTACT_FORM_SUCCESS_MESSAGE` et `NUXT_PUBLIC_CONTACT_FORM_ERROR_MESSAGE`.
4. Lancer la GitHub Action “Deploy Nuxt site to Pages” ou exécuter `npm run generate` puis pousser le contenu de `.output/public`.
5. Vérifier le site : chargement des pages statiques et envoi du formulaire (consulter le tableau de bord du service externe).

Étapes Render (exemple)

1. Forker le dépôt sur GitHub.
2. Créer un nouveau service Web sur Render (Node.js).
3. Renseigner la commande build : `npm run build`.
4. Renseigner la commande start : `node .output/server/index.mjs`.
5. Ajouter les variables d'environnement (`MAIL_*`).
6. Déployer. Render installe les dépendances et lance la commande build.

Déploiement Docker (serveur ou local)

1. Copier et compléter la configuration : `cp .env.example .env`.
2. Construire l'image : `docker build -t portfolio-nuxt .`.
3. Lancer le conteneur :

```
docker run -d \
  --env-file .env \
  -v $(pwd)/data:/app/data \
  -p 3000:3000 \
  --name portfolio \
  portfolio-nuxt
```

> Le volume `$(pwd)/data` persiste la base SQLite (`contact_messages`).

4. Vérifier que l'application répond sur `http://localhost:3000`.

Vérifications post-déploiement

- Tester `/` (chargement de la homepage).
- Tester `/docs/project-presentation` (contenu statique accessible publiquement).
- Tester le formulaire de contact :
  - Sur GitHub Pages : vérifier la réception via le service tiers (Formspree, EmailJS, etc.).
  - Sur déploiement Node : s'assurer que le mail est envoyé et que la ligne est présente dans `contact_messages`.
- Sur hébergement Node, vérifier les logs (`docker logs portfolio` ou console Render) pour confirmer l'absence d'erreurs 500.

Dépannage

- **Erreur SMTP** : vérifier `MAIL_PORT` (465 => `secure: true`).
- **Build échoue** : s'assurer que la version Node >= 20 et que les dépendances sont installées.
- **Pages docs en 404** : vérifier que le dossier `pages/docs` est bien présent dans la branche déployée.
- **Base SQLite non créée** : vérifier que le dossier `data/` est accessible en écriture par le processus (volume manquant sur Docker ou droits insuffisants).
- **Formulaire inactif sur GitHub Pages** : contrôler `NUXT_PUBLIC_CONTACT_FORM_ENDPOINT` et les réglages du service externe (domaines autorisés, quotas).

---

deployment-report.md

Rapport de déploiement

Contexte

- **Date** : 2025-10-26
- **Version** : `v1.1`
- **Serveur visé** : Node 20 / Docker (image basée sur `node:20-alpine`)

Préparation

1. Installation des dépendances : `npm install`
2. Compilation : `npm run build`
   - Sortie générée dans `.output/`
   - Aucun blocage (quelques warnings Browserslist/Tailwind connus)
3. Image Docker disponible (`Dockerfile`) pour une livraison containerisée.

Validation

- **Tests automatisés** : `npm test` (Vitest + couverture V8)
  - Cas couverts : validation formulaire de contact (TC01/TC02).
  - Rapport HTML généré dans `coverage/index.html`.
- **Base SQLite** : le dossier `data/` est créé automatiquement ; volume recommandé en production pour persister les messages.

Actions post-déploiement

- Monter le volume : `-v /path/data:/app/data`
- Charger les variables d'environnement (`.env` ou secrets Render).
- Lancer le serveur : `node .output/server/index.mjs` (ou `docker run` selon la cible).

---

qa-monitoring.md

Monitoring & retours utilisateurs

Méthodologie

- Intégration d'un formulaire de contact relié à une boîte dédiée (suivi des demandes).
- Tableau de suivi des retours (Google Sheet) avec statut : Nouveau → Analyse → Résolu.
- Analyse des sessions (prévu) via Google Analytics 4.

Indicateurs suivis

Indicateur	Objectif	Fréquence
Nombre de messages entrants	≥ 3 / mois	Mensuelle
Délai de réponse	< 48h	Hebdomadaire
Taux d'erreur formulaire	< 5%	Mensuelle
Temps moyen sur la page projets	> 45s	Mensuelle

Boucle d'amélioration

1. Collecter le feedback (formulaire, entretiens).
2. Prioriser selon l'impact (Matrice Valeur / Effort).
3. Mettre à jour le backlog Trello.
4. Déployer et communiquer la mise à jour.

---

accessibility-report.md

Analyse accessibilité

Points conformes

- Contraste texte/fond vérifié (Lighthouse ≥ 4.5:1).
- Focus visible sur tous les liens et boutons.
- Navigation clavier possible (tabindex naturel, modal fermée avec ESC).
- Texte alternatif descriptif pour chaque image.

Améliorations planifiées

- Ajouter un bouton “Passer au contenu”.
- Implémenter la prise en charge de la navigation via la touche Enter sur le carousel.
- Vérifier la hiérarchie des titres (H1 unique, H2/H3 cohérents).
- Prévoir une version anglaise pour l'internationalisation.

---

document-argumentatif.md

Document argumentatif

Constat

Le portfolio répond aux objectifs de présentation, mais plusieurs améliorations peuvent booster l'engagement et la fiabilité.

Propositions clés

1. **Automatisation QA** :
   - Gain : réduction des régressions.
   - Coût : faible (mise en place Vitest + Playwright).
2. **Déploiement automatisé** :
   - Gain : livraison rapide, reproductible.
   - Coût : configuration GitHub Actions + Render.
3. **Internationalisation** :
   - Gain : toucher les recruteurs non francophones.
   - Coût : traduction FR → EN/ PT, structure i18n.
4. **Tracking analytique** :
   - Gain : décisions basées sur les données.
   - Coût : configuration GA4 ou Plausible.

Priorisation (ICE)

Proposition	Impact	Confiance	Effort	Score
Automatisation QA	8	7	4	13.5
Déploiement auto	9	6	5	12.0
Internationalisation	7	5	6	9.0
Tracking analytique	6	8	3	12.0

Conclusion

Les actions 1 et 2 sont à prioriser pour sécuriser les livraisons et gagner du temps sur la maintenance. L'internationalisation peut suivre une fois ces fondations posées.

---

working-environment.md

Environnement de travail & collaboration

Outils mis en place

- **Gestion de versions** : Git (workflow feature branch), dépôt GitHub privé.
- **Gestion de projet** : Trello avec colonnes Backlog → En cours → Revue → Terminé.
- **Communication** : Discord (synchrone) et email (suivi hebdomadaire).
- **CI/CD (prévu)** : GitHub Actions pour automatiser les tests et le déploiement.
- **Qualité & build** : Vitest pour les tests unitaires, Docker pour les livraisons reproductibles.

Architecture du dépôt

```
portfolio/
├─ app.vue                # Layout global
├─ components/            # Sections du site
├─ pages/                 # Routes (homepage + docs publiques)
├─ server/api/            # Endpoint contact
├─ server/services/       # Règles métier (contact)
├─ server/utils/          # Initialisation SQLite
├─ composables/           # Hooks Nuxt (contact, etc.)
├─ docs/                  # Documentation projet
├─ public/                # Maquette SVG et assets statiques
├─ data/                  # Base SQLite (contact_messages)
├─ Dockerfile             # Image de déploiement
└─ .env.example           # Variables d'environnement à configurer
```

Processus de contribution

1. Création d'une branche dédiée (`feature/section-hero`).
2. Développement et tests locaux (`npm run dev` ou `npm run build`).
3. Pull request avec description des changements et vérification du plan de tests.
4. Revue croisée + validation avant fusion dans `main` .

Suivi & reporting

- **Daily** : message Discord rapide (avancement, blocages, plan du jour).
- **Weekly** : point de 30 minutes pour prioriser les fonctionnalités.
- **Livrables** : chaque fonctionnalité s'accompagne d'une mise à jour des docs.

---

maquette.md

Maquette

La maquette du portfolio illustre un parcours vertical avec les sections suivantes :

1. Hero (photo, coordonnées, CTA).
2. À propos (parcours, compétences, langues).
3. Points forts (carousel).
4. Projets.
5. Galerie.
6. Documentation.
7. Contact.

Le fichier `public/wireframe.svg` reprend cette structure sous la forme d'un wireframe haute fidélité. Les couleurs principales sont :

- Fond : `#0a0a0a`
- Accent : `#ffdd00`
- Texte : `#ffffff`

Typographie : sans-serif moderne (interchangeable avec Inter / Poppins). Grid de 12 colonnes, marges 24 px.

---

figma-homepage.css

/* Figma reference stylesheet for the portfolio homepage layout. Purpose: reproduce the structure in Figma without touching the production styles. Each block mirrors a homepage section (Hero → Contact). */

:root { --bg-dark: #0a0a0a; --bg-panel: #151515; --accent: #ffdd00; --text-primary: #ffffff; --text-muted: #9ca3af; --section-width: 1100px; --radius-lg: 32px; }

body { font-family: 'Inter', sans-serif; background-color: var(--bg-dark); color: var(--text-primary); margin: 0; padding: 0; }

.page { display: flex; flex-direction: column; gap: 80px; padding: 80px 32px 120px; max-width: var(--section-width); margin: 0 auto; }

/* HERO */ .hero { display: grid; grid-template-columns: repeat(2, minmax(0, 1fr)); gap: 48px; padding: 64px; border-radius: var(--radius-lg); background: linear-gradient(180deg, #000000 0%, #121212 45%, #1a1a1a 100%); box-shadow: 0 40px 120px rgba(0, 0, 0, 0.45); }

.hero__infos { display: flex; flex-direction: column; gap: 24px; }

.hero__subtitle { text-transform: uppercase; letter-spacing: 0.3em; color: rgba(255, 221, 0, 0.7); font-size: 12px; }

.hero__title { font-size: 48px; color: var(--accent); margin: 0; }

.hero__paragraph { line-height: 1.7; color: #d1d5db; }

.hero__cta { display: flex; flex-wrap: wrap; gap: 16px; }

.hero__cta .primary { background: var(--accent); color: #000; padding: 14px 32px; border-radius: 999px; font-weight: 600; }

.hero__cta .secondary { border: 1px solid var(--accent); color: var(--accent); padding: 14px 32px; border-radius: 999px; font-weight: 600; }

.hero__portrait { border-radius: 40px; background: url('/images/hero-portrait.jpg') center/cover no-repeat; min-height: 420px; box-shadow: 0 0 60px rgba(255, 221, 0, 0.25); }

/* ABOUT */ .about { display: grid; grid-template-columns: 1.1fr 0.9fr; gap: 48px; }

.panel { border-radius: var(--radius-lg); background: var(--bg-panel); padding: 40px; border: 1px solid rgba(255, 221, 0, 0.15); }

.panel + .panel { margin-top: 24px; }

.panel h3 { margin-top: 0; margin-bottom: 16px; font-size: 20px; color: var(--accent); }

.chips { display: flex; flex-wrap: wrap; gap: 10px; }

.chips span { border-radius: 999px; border: 1px solid rgba(255, 221, 0, 0.3); padding: 8px 20px; font-size: 12px; letter-spacing: 0.1em; text-transform: uppercase; color: var(--accent); }

/* COMPETENCIES */ .competencies__grid { display: grid; gap: 32px; }

.competency-card { border: 1px solid rgba(255, 221, 0, 0.2); border-radius: var(--radius-lg); padding: 32px; background: rgba(21, 21, 21, 0.8); display: grid; grid-template-columns: repeat(2, minmax(0, 1fr)); gap: 24px; }

.competency-card h4 { text-transform: uppercase; font-size: 12px; letter-spacing: 0.3em; color: rgba(255, 221, 0, 0.6); margin-bottom: 12px; }

/* CAROUSEL */ .carousel { display: grid; grid-template-columns: 1fr 1fr; gap: 48px; align-items: center; }

.carousel__frame { border-radius: var(--radius-lg); border: 1px solid rgba(255, 221, 0, 0.2); overflow: hidden; min-height: 360px; background: url('/images/gallery-top-hardware.png') center/cover no-repeat; }

/* PROJECTS */ .projects__grid { display: grid; grid-template-columns: repeat(2, minmax(0, 1fr)); gap: 32px; }

.project-card { border-radius: var(--radius-lg); border: 1px solid rgba(255, 221, 0, 0.15); padding: 32px; background: var(--bg-panel); min-height: 280px; display: flex; flex-direction: column; justify-content: space-between; }

.project-card h3 { margin: 0; font-size: 22px; }

.project-card p { color: var(--text-muted); line-height: 1.6; }

/* GALLERY */ .gallery { display: grid; grid-template-columns: repeat(3, minmax(0, 1fr)); gap: 24px; }

.gallery__item { border-radius: 24px; min-height: 200px; border: 1px solid rgba(255, 221, 0, 0.2); background-size: cover; background-position: center; }

/* CONTACT */ .contact { border-radius: var(--radius-lg); border: 1px solid rgba(255, 221, 0, 0.2); padding: 40px; background: var(--bg-panel); display: grid; gap: 24px; }

.contact__form { display: grid; gap: 20px; }

.contact__form label { text-transform: uppercase; font-size: 12px; letter-spacing: 0.2em; color: rgba(255, 221, 0, 0.7); }

.contact__form input, .contact__form textarea { background: #1a1a1a; border: 1px solid #333; border-radius: 16px; padding: 16px 20px; color: var(--text-primary); }

.contact__form button { border: none; border-radius: 999px; background: var(--accent); color: #000; padding: 16px 32px; font-weight: 600; text-transform: uppercase; }

---

cahier-des-charges.md

Cahier des charges – Portfolio Ricardo Chaves Rodrigues

1. Contexte & objectifs

- **Sujet** : créer un portfolio hybride (apprentissage + présentation) démontrant toutes les compétences du titre RNCP.
- **Contrat** : page vitrine en ligne, responsive, flat design, mise à jour continue, documentation accessible à l'équipe pédagogique.
- **Utilisateurs cibles** :
  - Recruteurs / responsables techniques (face présentation).
  - Équipe pédagogique et proches (face apprentissage).

2. Exigences principales

Domaine	Attendus du sujet	État
Identité	Accroche, photo professionnelle, coordonnées complètes, CV PDF	✅ `HeroSection.vue` + lien `/CDI_CV_...`
Parcours & compétences	Sections À propos, Points forts, preuve RNCP par bloc	✅ `AboutSection.vue`, `CompetenciesSection.vue`
Travaux	Regrouper plusieurs projets + visuels	✅ `ProjectsList.vue`, `ImageModal.vue`, `Carousel.vue`
Contact	Email direct + formulaire relié à un service externe (Resend ou `/api/contact`)	✅ `HeroSection.vue` + `ContactForm.vue` / `server/api/contact`
Documentation	Accessible en ligne (présentation, tests, guide, améliorations)	✅ `/docs/*` pages + fichiers `docs/*.md` synchronisés
Mise à jour	Processus décrit (Trello, QA monitoring, improvement backlog)	✅ `docs/qa-monitoring.md`, `docs/improvement-proposals.md`
Accessibilité	Respect WCAG, navigation clavier, modal ESC	✅ `docs/accessibility-report.md`, `ImageModal.vue`

3. Couverture composant ↔ documentation

Section UI	Composant	Documentation associée
Hero & navigation	`HeroSection.vue`, `AppHeader.vue`	`docs/project-presentation.md`, `docs/user-guide.md`
À propos & compétences	`AboutSection.vue`, `CompetenciesSection.vue`	`docs/project-presentation.md` (analyse RNCP), `pages/docs/project-presentation.vue`
Points forts / Galerie	`Carousel.vue`, `ImageModal.vue`	`docs/maquette.md`, `figma-homepage.css`
Projets	`ProjectsList.vue`	`docs/project-presentation.md`, `docs/improvement-proposals.md` (CMS/filtre)
Contact	`ContactForm.vue`, `server/api/contact`, `server/services/contactService.ts`	`docs/technical-documentation.md`, `docs/test-plan.md`, `docs/deployment-guide.md`

Section UI publique	Composant `pages/docs/*`	Documentation associée Synchronisé avec `docs/*.md` (maj mai 2025)

4. Documents livrés

- **Figma / wireframe** : `public/wireframe.svg` + `docs/figma-homepage.css`.
- **Documentation technique** : `docs/technical-documentation.md`, `docs/specifications-techniques.md`.
- **Qualité** : `docs/test-plan.md`, `docs/test-report.md`, `docs/qa-monitoring.md`.
- **Déploiement** : `docs/deployment-guide.md`, `docs/deployment-report.md`.
- **Analyse** : `docs/project-presentation.md`, `docs/improvement-proposals.md`, `docs/document-argumentatif.md`.

5. Points vérifiés (mai 2025)

1. Pages `/docs` alignées avec les fichiers Markdown (user guide, test plan, improvements, project presentation).
2. Section RNCP mise à jour dans la page documentation.
3. Formulaire connecté à Resend (variables `RESEND_*` + fallback SMTP).
4. Base SQLite stockée dans `NUXT_DATA_DIR` (compatibilité Vercel).

6. Points d’attention / questions

1. **Internationalisation** : souhaites-tu prioriser une langue (EN/PT) ? Les textes ne sont pas encore traduits.
2. **CMS / données dynamiques** : valider le choix (Contentful vs Strapi) avant de structurer `ProjectsList`.
3. **Analytics** : préfères-tu GA4 ou Plausible pour respecter les contraintes RGPD ?
4. **Tests E2E** : Playwright est prévu ; faut-il intégrer une matrice détaillée pour TC21/22 avant la prochaine soutenance ?

Merci de confirmer ces choix ou de préciser d’autres priorités pour que la roadmap reste alignée avec les attentes pédagogiques.