Portail

Procédure — Structure, ajout d'application et déploiement

Développeur	COUR Cyrille
Projet	Portail — page d'accueil app launcher de cyrillecode.fr
Stack	HTML5 / CSS3 / JavaScript
Date	24/08/2026
Version	1.0

Sommaire

1. Contexte

Portail est la page d'accueil de cyrillecode.fr : une grille de tuiles, façon « app launcher » inspirée du portail applicatif d'Odoo, qui renvoie vers chacune des applications du portfolio (Portfolio, Gourmandex, Wanderbook, MyVAR...).

L'application est volontairement légère : HTML/CSS/JS vanilla, sans framework ni dépendance externe, sans backend ni base de données. Cette procédure documente sa structure, la façon d'ajouter une application à la grille, et le pipeline de déploiement mis en place.

Note — Cette procédure s'adresse à toute personne (moi-même en premier lieu) devant faire évoluer le portail : ajouter une app, modifier le thème visuel, ou intervenir sur le déploiement.

2. Environnement

2.1 Prérequis

- Un navigateur — aucun outil de build n'est requis.
- Node.js (optionnel) — uniquement pour servir le dossier en local via un petit serveur statique.
- Git, pour cloner le dépôt et pousser les modifications.

2.2 Installation locale

```
$ git clone <url-du-depot> Portail
$ cd Portail
```

Aucune installation de dépendances n'est nécessaire. Deux options pour visualiser le portail en local :

- Ouvrir directement index.html dans un navigateur.
- Servir le dossier avec un serveur statique local (recommandé pour éviter les restrictions CORS/favicon) :

```
$ npx serve .
```

3. Procédure

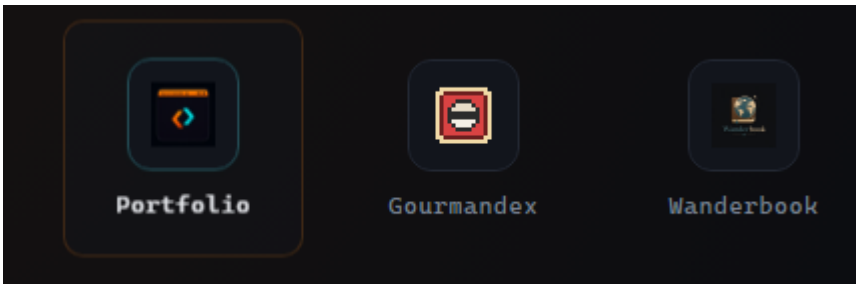
3.1 Étape 1 — Ajouter une application à la grille

Détail technique

La grille est entièrement pilotée par un tableau JavaScript apps défini en tête de scripts.js. Chaque entrée génère automatiquement une tuile au chargement de la page — aucune modification de index.html ou style.css n'est nécessaire pour ajouter une app.

```
const apps = [
  { name: "Portfolio", url: "https://cyrillecode.fr", icon: "Images/CyrilleCode.png" },
  { name: "Gourmandex", url: "https://gourmandex.cyrillecode.fr", icon: "Images/gourmandex.ico" },
  { name: "Wanderbook", url: "https://wanderbook.cyrillecode.fr", icon: "Images/wanderbook.ico" },
  { name: "MyVAR", url: "https://myvar.cyrillecode.fr", icon: "https://myvar.cyrillecode.fr/favicon.ico" },
];
```

Le champ icon accepte soit une URL distante (favicon de l'app cible), soit un chemin local dans Images/ pour les apps sans favicon exploitable (cas de Portfolio, qui utilise Images/CyrilleCode.png).



3.2 Étape 2 — Modifier le thème visuel

Les couleurs d'accent (--orange, --cyan) sont définies en variables CSS en tête de style.css. La police reste volontairement une police monospace système (aucun @font-face, aucune requête réseau au chargement) : ne pas charger de police externe sans modifier cette contrainte de conception en connaissance de cause.

3.3 Étape 3 — Vérifier avant de pousser

Avant tout push, vérifier localement :

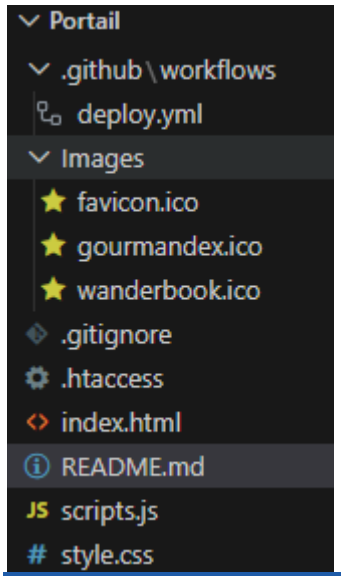
- La grille s'affiche correctement avec la nouvelle app (icône visible, lien correct).
- L'horloge du bandeau fonctionne toujours (aucune régression JS).
- Aucune erreur dans la console navigateur (404 favicon, etc.).

4. Structure / Données

Il n'y a pas de base de données : la seule « donnée » du projet est le tableau apps, codé en dur dans scripts.js.

Champ	Type	Rôle
name	string	Nom affiché sous la tuile
url	string	URL cible (sous-domaine cyrillecode.fr)
icon	string	URL du favicon distant, ou chemin local dans Images/

4.1 Arborescence du projet



5. Déploiement / CI-CD

Le déploiement repose sur un pipeline CI/CD volontairement minimal : à chaque push sur la branche principale, GitHub Actions envoie le contenu du dépôt sur le serveur, sans étape de build ni de tests — cohérent avec un projet 100 % statique qui n'a rien à compiler.

5.1 Déclenchement

- Trigger : push sur la branche principale (main).
- Aucune étape de build : les fichiers HTML/CSS/JS sont déployés tels quels.
- Aucune étape de test : la légèreté du projet ne justifie pas une suite de tests dédiée à ce stade.

5.2 Transfert vers le serveur

Le job GitHub Actions transfère les fichiers du dépôt vers le répertoire web du serveur (SCP/rsync selon le runner configuré), de façon équivalente au principe déjà en place pour Papie, mais sans les étapes de build/vendor qui n'ont pas de sens ici faute de dépendances.