Canada 2.0 — Plan de mise en œuvre (OpenG7)

\*\*Principe directeur :\*\* \*Public code, private state.\*

\*\*Portée :\*\* Inspirer et outiller les provinces canadiennes pour innover entre elles sur les intrants économiques (électricité, pétrole, services, main‑d’œuvre spécialisée, logistique), avec une chaîne logicielle 100 % ouverte et souveraine.

**──────────────────────────────────────────────────**

## 1) Gouvernance minimale (90 jours)

\*\*Trépied\*\*

* \*\*État (Fédéral + Provinces)\*\* : définit la stratégie, priorise, garde et opère les secrets (clés/tokens), valide la conformité.
* \*\*Communauté (Open Source)\*\* : conçoit, code, teste, documente et maintient.
* \*\*Industrie & Académique\*\* : intègre, forme, certifie, apporte l’expertise.

\*\*Org chart minimal\*\*

* \*\*Direction programme Canada 2.0 (État)\*\*
* \*\*Architecture & Sécurité\*\* (SecOps, KMS, conformité)
* \*\*Plateformes MVP\*\* (3 lead produit : API Registry, Talents, Flux Énergie)
* \*\*Interop & Données\*\* (schémas, catalogues, SBOM)
* \*\*Maintainers OSS (OpenG7)\*\*
* \*\*Review & Releases\*\* (CI/CD, GitHub Actions, attestations)
* \*\*DX & Docs\*\* (guides, templates, i18n, accessibilité)

\*\*Rituels\*\*

* \*\*RFC publiques\*\* (2 semaines), \*\*démonstrations bi‑hebdo\*\*, \*\*roadmap trimestrielle\*\*.
* \*\*Bug bounty\*\* (responsible disclosure) et \*\*pentests\*\* avant chaque mise en production majeure.

**──────────────────────────────────────────────────**

## 2) MVPs (0→90 jours)

### MVP A — Registre national d’APIs (interprovincial)

* Inventorie, normalise et publie les \*\*APIs de capacité/flux\*\* (énergie, transport, services).
* Mises à jour versionnées, schémas OpenAPI/AsyncAPI, disponibilité (SLA), gouvernance de changement.
* \*\*Objectif 90j :\*\* ≥ 200 endpoints actifs, SBOM public, latence et uptime monitorés.

### MVP B — Place de marché de \*\*main‑d’œuvre spécialisée\*\*

* Profils vérifiés (qualifications, sécurité, mobilité), contrats types, disponibilité en temps quasi réel.
* \*\*Objectif 90j :\*\* 5 secteurs prioritaires (énergie, santé, transport, construction, TI), 5 000 profils.

### MVP C — Tableau \*\*Flux Énergie\*\* (quasi temps réel)

* Vue agrégée des flux d’électricité/pétrole/gaz par province (capacités, indisponibilités planifiées, prix/tarifs).
* \*\*Objectif 90j :\*\* intégration de 2 opérateurs électriques + 1 pipeline, prévision J+1 simple.

\*\*Livrables transverses\*\*

* \*\*Charte “Public code, Private state”\*\* (ci‑dessous)
* \*\*Modèles de données\*\* et schémas partagés
* \*\*Kits de démarrage\*\* (squelettes Angular/NestJS, CI/CD, i18n, NgRx)

**──────────────────────────────────────────────────**

## 3) Charte « Public code, Private state » (extrait opérationnel)

1. \*\*Code & specs publics\*\* (Apache‑2.0). Toute exception est justifiée et publiée.
2. \*\*Secrets\*\* (clés, tokens, certificats, endpoints privés) \*\*jamais\*\* dans le code : stockage KMS, rotation auto, accès just‑in‑time (JIT).
3. \*\*Build & déploiements attestés\*\* : SBOM, signature d’artefacts, provenance (SLSA), journal d’audit immuable.
4. \*\*Données\*\* : par défaut ouvertes (licences compatibles); personnelles/sensibles → chiffrement, minimisation, journalisation d’accès.
5. \*\*Sécurité “shift‑left”\*\* : SAST/SCA/IaC scans en CI, revues obligatoires, tests d’intrusion périodiques.
6. \*\*Interop\*\* : identités fédérées, logs et corrélation interprovinciales, i18n FR/EN par défaut.

**──────────────────────────────────────────────────**

## 4) DevSecOps souverain (GitHub → build canadien)

* \*\*Branches\*\* : trunk‑based (feature branches + PR), releases taguées sémantiquement.
* \*\*CI\*\* : lint/format/test, SAST (CodeQL), SCA, tests E2E (Playwright), \*\*SBOM (Syft)\*\*, \*\*signature\*\* (Cosign).
* \*\*CD\*\* : GitOps (ArgoCD/FluxCD), environnements par province, approbations signées.
* \*\*Observabilité\*\* : OpenTelemetry, budgets de performance Web, SLO par appli.

**──────────────────────────────────────────────────**

## 5) Modèles de données (extraits TypeScript)

> Tous les chemins cités respectent la convention demandée : `src/app/...`

\*\*`src/app/shared/models/province.enum.ts`\*\*

export enum Province {  
 AB = 'AB', BC = 'BC', MB = 'MB', NB = 'NB', NL = 'NL', NS = 'NS', NT = 'NT',  
 NU = 'NU', ON = 'ON', PE = 'PE', QC = 'QC', SK = 'SK', YT = 'YT'  
}

\*\*`src/app/shared/models/sector-type.enum.ts`\*\*

export enum SectorType {  
 ENERGY = 'ENERGY', TRANSPORT = 'TRANSPORT', SERVICES = 'SERVICES',  
 MANUFACTURING = 'MANUFACTURING', CONSTRUCTION = 'CONSTRUCTION', IT = 'IT'  
}

\*\*`src/app/shared/models/api-endpoint.model.ts`\*\*

import { Province } from './province.enum';  
import { SectorType } from './sector-type.enum';  
  
export interface ApiEndpoint {  
 id: string; // UUID  
 name: string; // Affiché au public  
 baseUrl: string; // URL publique  
 schemaUrl?: string; // OpenAPI/AsyncAPI public  
 sector: SectorType;  
 province: Province;  
 status: 'UP' | 'DOWN' | 'DEGRADED';  
 lastUpdated: string; // ISO  
}

**──────────────────────────────────────────────────**

## 6) Frontend Angular — \*\*architecture signal‑first\*\* (exemples prêts à coller)

### 6.1 Routage

\*\*`src/app/app.routes.ts`\*\*

import { Routes } from '@angular/router';  
  
export const routes: Routes = [  
 { path: '', redirectTo: 'apis', pathMatch: 'full' },  
 { path: 'apis', loadComponent: () => import('./features/api-registry/api-registry.component').then(m => m.ApiRegistryComponent) },  
 { path: 'talents', loadComponent: () => import('./features/talent-marketplace/talent-marketplace.component').then(m => m.TalentMarketplaceComponent) },  
 { path: 'energy', loadComponent: () => import('./features/energy-flows/energy-flows.component').then(m => m.EnergyFlowsComponent) },  
];

### 6.2 Configuration application (NgRx + i18n)

\*\*`src/app/app.config.ts`\*\*

import { ApplicationConfig, importProvidersFrom } from '@angular/core';  
import { provideRouter } from '@angular/router';  
import { routes } from './app.routes';  
import { provideStore } from '@ngrx/store';  
import { provideState } from '@ngrx/store';  
import { provideEffects } from '@ngrx/effects';  
import { apiRegistryFeature } from './state/api-registry/api.reducer';  
import { ApiRegistryEffects } from './state/api-registry/api.effects';  
import { provideHttpClient, withInterceptorsFromDi } from '@angular/common/http';  
import { TranslateLoader, TranslateModule } from '@ngx-translate/core';  
import { HttpClient } from '@angular/common/http';  
import { TranslateHttpLoader } from '@ngx-translate/http-loader';  
  
export function httpLoaderFactory(http: HttpClient) {  
 return new TranslateHttpLoader(http, 'assets/i18n/', '.json');  
}  
  
export const appConfig: ApplicationConfig = {  
 providers: [  
 provideRouter(routes),  
 provideHttpClient(withInterceptorsFromDi()),  
 provideStore(),  
 provideState(apiRegistryFeature),  
 provideEffects([ApiRegistryEffects]),  
 importProvidersFrom(TranslateModule.forRoot({  
 defaultLanguage: 'fr',  
 loader: { provide: TranslateLoader, useFactory: httpLoaderFactory, deps: [HttpClient] },  
 })),  
 ],  
};

### 6.3 Composant — Registre d’APIs (signal‑first, formulaire typé)

\*\*`src/app/features/api-registry/api-registry.component.ts`\*\*

import { CommonModule } from '@angular/common';  
import { Component, computed, effect, inject, signal } from '@angular/core';  
import { ReactiveFormsModule, FormBuilder, Validators } from '@angular/forms';  
import { TranslateModule, TranslateService } from '@ngx-translate/core';  
import { Store } from '@ngrx/store';  
import { Province } from '../../shared/models/province.enum';  
import { SectorType } from '../../shared/models/sector-type.enum';  
import { ApiEndpoint } from '../../shared/models/api-endpoint.model';  
import { selectAllApis } from '../../state/api-registry/api.selectors';  
import { ApiRegistryActions } from '../../state/api-registry/api.actions';  
  
interface ApiForm {  
 name: string; baseUrl: string; schemaUrl?: string; province: Province; sector: SectorType;  
}  
  
@Component({  
 selector: 'app-api-registry',  
 standalone: true,  
 imports: [CommonModule, ReactiveFormsModule, TranslateModule],  
 template: `  
 <section class="p-4 md:p-6 space-y-4">  
 <header class="flex items-center gap-3">  
 <h1 class="text-2xl font-semibold">{{ 'apis.title' | translate }}</h1>  
 <input  
 type="search" placeholder="{{ 'common.search' | translate }}"  
 class="px-3 py-2 rounded-2xl border w-full md:w-80"  
 [value]="query()" (input)="query.set(($event.target as HTMLInputElement).value)" />  
 <button class="px-4 py-2 rounded-2xl border" (click)="showAdd.set(!showAdd())">  
 {{ showAdd() ? ('common.close' | translate) : ('apis.add' | translate) }}  
 </button>  
 </header>  
  
 <form \*ngIf="showAdd()" [formGroup]="form" (ngSubmit)="submit()" class="grid md:grid-cols-5 gap-3 items-end">  
 <input class="border rounded-xl px-3 py-2" placeholder="Nom" formControlName="name" />  
 <input class="border rounded-xl px-3 py-2" placeholder="https://…" formControlName="baseUrl" />  
 <input class="border rounded-xl px-3 py-2" placeholder="Schema URL (opt)" formControlName="schemaUrl" />  
 <select class="border rounded-xl px-3 py-2" formControlName="province">  
 <option \*ngFor="let p of provinces" [value]="p">{{ p }}</option>  
 </select>  
 <select class="border rounded-xl px-3 py-2" formControlName="sector">  
 <option \*ngFor="let s of sectors" [value]="s">{{ s }}</option>  
 </select>  
 <button class="mt-2 md:mt-0 px-4 py-2 rounded-xl border" type="submit" [disabled]="form.invalid">{{ 'common.save' | translate }}</button>  
 </form>  
  
 <div class="grid gap-2">  
 <article \*ngFor="let api of filtered()" class="p-3 rounded-2xl border flex items-center justify-between">  
 <div>  
 <h3 class="font-semibold">{{ api.name }}</h3>  
 <p class="text-sm opacity-70">{{ api.baseUrl }}</p>  
 </div>  
 <div class="text-sm">  
 <span class="px-2 py-1 rounded-full border mr-2">{{ api.province }}</span>  
 <span class="px-2 py-1 rounded-full border">{{ api.sector }}</span>  
 </div>  
 </article>  
 </div>  
 </section>  
 `,  
})  
export class ApiRegistryComponent {  
 private fb = inject(FormBuilder);  
 private store = inject(Store);  
 private i18n = inject(TranslateService);  
  
 provinces = Object.values(Province);  
 sectors = Object.values(SectorType);  
  
 showAdd = signal(false);  
 query = signal('');  
  
 form = this.fb.nonNullable.group<ApiForm>({  
 name: this.fb.nonNullable.control('', { validators: [Validators.required, Validators.minLength(3)] }),  
 baseUrl: this.fb.nonNullable.control('', { validators: [Validators.required] }),  
 schemaUrl: this.fb.control<string | undefined>(undefined),  
 province: this.fb.nonNullable.control(Province.QC),  
 sector: this.fb.nonNullable.control(SectorType.SERVICES),  
 });  
  
 all = this.store.selectSignal(selectAllApis);  
  
 filtered = computed(() => {  
 const q = this.query().toLowerCase().trim();  
 return this.all().filter(a => !q || a.name.toLowerCase().includes(q) || a.baseUrl.toLowerCase().includes(q));  
 });  
  
 submit() {  
 if (this.form.invalid) return;  
 const value = this.form.getRawValue();  
 const api: ApiEndpoint = {  
 id: crypto.randomUUID(),  
 name: value.name,  
 baseUrl: value.baseUrl,  
 schemaUrl: value.schemaUrl,  
 sector: value.sector,  
 province: value.province,  
 status: 'UP',  
 lastUpdated: new Date().toISOString(),  
 };  
 this.store.dispatch(ApiRegistryActions.addRequested({ api }));  
 this.form.reset({ name: '', baseUrl: '', schemaUrl: undefined, province: Province.QC, sector: SectorType.SERVICES });  
 this.showAdd.set(false);  
 }  
}

### 6.4 NgRx 19 — feature minimal (utilisé parcimonieusement)

\*\*`src/app/state/api-registry/api.actions.ts`\*\*

import { createActionGroup, emptyProps, props } from '@ngrx/store';  
import { ApiEndpoint } from '../../shared/models/api-endpoint.model';  
  
export const ApiRegistryActions = createActionGroup({  
 source: 'ApiRegistry',  
 events: {  
 'Load Requested': emptyProps(),  
 'Loaded Success': props<{ apis: ApiEndpoint[] }>(),  
 'Add Requested': props<{ api: ApiEndpoint }>(),  
 'Added Success': props<{ api: ApiEndpoint }>(),  
 'Failure': props<{ error: unknown }>(),  
 },  
});

\*\*`src/app/state/api-registry/api.reducer.ts`\*\*

import { createFeature, createReducer, on } from '@ngrx/store';  
import { ApiEndpoint } from '../../shared/models/api-endpoint.model';  
import { ApiRegistryActions } from './api.actions';  
  
export interface ApiRegistryState {  
 apis: ApiEndpoint[];  
 loading: boolean;  
 error?: string;  
}  
  
const initialState: ApiRegistryState = { apis: [], loading: false };  
  
const reducer = createReducer(  
 initialState,  
 on(ApiRegistryActions.loadRequested, (s) => ({ ...s, loading: true })),  
 on(ApiRegistryActions.loadedSuccess, (s, { apis }) => ({ ...s, loading: false, apis })),  
 on(ApiRegistryActions.addRequested, (s, { api }) => ({ ...s, apis: [api, ...s.apis] })),  
 on(ApiRegistryActions.failure, (s, { error }) => ({ ...s, loading: false, error: String(error) })),  
);  
  
export const apiRegistryFeature = createFeature({ name: 'apiRegistry', reducer });  
export const { name: apiRegistryFeatureKey, reducer: apiRegistryReducer, selectApiRegistryState } = apiRegistryFeature;

\*\*`src/app/state/api-registry/api.selectors.ts`\*\*

import { createSelector } from '@ngrx/store';  
import { selectApiRegistryState } from './api.reducer';  
  
export const selectAllApis = createSelector(selectApiRegistryState, (s) => s.apis);

> \*\*Signal‑first\*\* : chaque composant gère ses propres `signal()`/`computed()`/`effect()` pour l’UI (filtres, affichage). Le \*\*Store\*\* ne sert qu’aux états partagés multi‑composants et à la synchronisation réseau.

### 6.5 i18n (ngx‑translate) — fichiers de base

\*\*`src/assets/i18n/fr.json`\*\*

{  
 "common": { "search": "Rechercher…", "save": "Enregistrer", "close": "Fermer" },  
 "apis": { "title": "Registre d'APIs", "add": "Ajouter" }  
}

\*\*`src/assets/i18n/en.json`\*\*

{  
 "common": { "search": "Search…", "save": "Save", "close": "Close" },  
 "apis": { "title": "API Registry", "add": "Add" }  
}

### 6.6 Autres composants (squelettes)

\*\*`src/app/features/talent-marketplace/talent-marketplace.component.ts`\*\*

import { CommonModule } from '@angular/common';  
import { Component, computed, signal } from '@angular/core';  
import { TranslateModule } from '@ngx-translate/core';  
  
@Component({  
 selector: 'app-talent-marketplace',  
 standalone: true,  
 imports: [CommonModule, TranslateModule],  
 template: `  
 <section class="p-4 space-y-4">  
 <h1 class="text-2xl font-semibold">{{ 'talents.title' | translate }}</h1>  
 <!-- TODO: filtres signal-first, résultats virtualisés -->  
 <p class="opacity-70">Prototype à compléter (profils vérifiés, contrats types, dispo en temps quasi réel).</p>  
 </section>`  
})  
export class TalentMarketplaceComponent {  
 // TODO: implémenter filtres en signals, appels NgRx si partagés  
}

\*\*`src/app/features/energy-flows/energy-flows.component.ts`\*\*

import { CommonModule, isPlatformBrowser } from '@angular/common';  
import { Component, ElementRef, Inject, PLATFORM\_ID, ViewChild, effect, signal } from '@angular/core';  
import { TranslateModule } from '@ngx-translate/core';  
  
@Component({  
 selector: 'app-energy-flows',  
 standalone: true,  
 imports: [CommonModule, TranslateModule],  
 template: `  
 <section class="p-4 space-y-4">  
 <h1 class="text-2xl font-semibold">{{ 'energy.title' | translate }}</h1>  
 <div #map class="w-full h-[60vh] rounded-2xl border"></div>  
 <p class="text-sm opacity-70">Carte énergie : flux agrégés interprovinciaux (prototype).</p>  
 </section>`  
})  
export class EnergyFlowsComponent {  
 @ViewChild('map', { static: true }) mapEl!: ElementRef<HTMLDivElement>;  
 constructor(@Inject(PLATFORM\_ID) private pid: object) {}  
  
 mapReady = signal(false);  
  
 ngAfterViewInit() {  
 if (!isPlatformBrowser(this.pid)) return;  
 // TODO: intégrer Leaflet ou jsVectorMap; architecture signal-first  
 this.mapReady.set(true);  
 }  
}

**──────────────────────────────────────────────────**

## 7) Sécurité & secrets (opérationnel)

* \*\*KMS (État)\*\* : stockage des clés/tokens, rotation 90j, accès JIT, mTLS, HSM où requis.
* \*\*CI/CD\*\* : secrets montés à l’exécution (OIDC → KMS), jamais dans les variables de dépôt.
* \*\*Scopes minimes\*\* : secrets par environnement/province/service.
* \*\*Audit\*\* : tous les accès aux secrets sont journalisés (immuables, corrélés aux déploiements).

**──────────────────────────────────────────────────**

## 8) Données & interopérabilité

* \*\*Schémas\*\* : OpenAPI/AsyncAPI versionnés, validation systématique en CI.
* \*\*Catalogues\*\* : métadonnées (qualité, fraîcheur, provenance), licences ouvertes.
* \*\*Identités\*\* : fédération (ex : OIDC interprovincial), RBAC/ABAC unifié.

**──────────────────────────────────────────────────**

## 9) Accessibilité & i18n

* \*\*FR/EN\*\* by default, RTL prêt.
* \*\*WCAG 2.2 AA\*\* : focus visible, contrastes, clavier, lecteurs d’écran.
* \*\*Tests\*\* : axe-core en CI, Playwright pour parcours critiques.

**──────────────────────────────────────────────────**

## 10) Mesures de succès (12–18 mois)

* 80 % des nouveaux systèmes publics livrés \*\*open source\*\*.
* 100 % des secrets \*\*hors code\*\* et \*\*rotations auto\*\*.
* ≥ 1 000 « Artisans Canada 2.0 » certifiés.
* 3 plateformes interprovinciales en prod (énergie, talents, APIs).
* TTM < 1 semaine pour les changements de politique mineurs.

**──────────────────────────────────────────────────**

## 11) Backlog initial (trié par impact)

1. \*\*Charte\*\* validée (État + OSS) et publiée.
2. \*\*Squelette Angular\*\* (routes + i18n + NgRx + Tailwind) et \*\*squelette NestJS\*\* (OpenAPI + auth) publiés.
3. \*\*CI/CD\*\* avec SBOM + signature + scans activés.
4. \*\*Registre d’APIs\*\* : CRUD minimal + import CSV/JSON.
5. \*\*Talents\*\* : modèle profil + filtres signal‑first.
6. \*\*Énergie\*\* : carte statique + premiers flux simulés.
7. \*\*Observabilité\*\* : budgets perf et traces de bout en bout.

**──────────────────────────────────────────────────**

## 12) Étapes suivantes (exécutables cette semaine)

* \*\*Publier\*\* ce plan dans le dépôt `openg7-platform` avec les squelettes

(`src/app/...` comme montré).

* \*\*Créer\*\* les workflows CI (build, tests, SBOM, signature).
* \*\*Ouvrir\*\* 12 issues « good first issue » + bounties symboliques.
* \*\*Planifier\*\* la première démo « Registre d’APIs » (formulaire + liste + i18n).

**──────────────────────────────────────────────────**

> Besoin d’un \*\*zip de démarrage\*\* ? On peut générer un squelette Angular conforme (signal-first, NgRx, i18n, Tailwind) avec l’arborescence ci‑dessus et des tests de base.