Bidra till Global Governance Frameworks webbplats

Välkommen till den tekniska bidragsguiden för Global Governance Frameworks webbplats! Den här guiden hjälper dig att förstå vår kodbas, utvecklingsarbetsflöde och hur du gör meningsfulla bidrag till projektet.

🎯 Projektöversikt

Global Governance Framework webbplats är byggd med moderna webteknologier och fungerar som en plattform för att dela styrningsramverk, implementeringsverktyg och främja globalt samarbete. Vårt uppdrag är att skapa verktyg, mönster och ramverk som möjliggör för olika styrsystem att interagera och utvecklas tillsammans.

Kärnteknologier

- **Frontend**: SvelteKit (senaste)
- **Styling**: Tailwind CSS med anpassat designsystem
- **Internationalisering**: Anpassad i18n-implementation som stöder engelska och svenska
- **Formulärhantering**: Formspree-integration
- **Innehållshantering**: Markdown-baserat innehåll med dynamisk routing
- **Byggverktyg**: Vite
- **Pakethanterare**: npm

🚀 Komma igång

Förutsättningar

- Node.js (v16 eller högre)
- npm (kommer med Node.js)
- Git
- En kodredigerare (VS Code rekommenderas)

Snabb installation

1. **Klona repository:**

```
git clone git@github.com:GlobalGovernanceFramework/governance-framework-site.git
cd governance-framework-site
```

2. **Installera beroenden:**

```
npm install
```

3. **Starta utvecklingsservern:**

```
npm run dev -- --open
```

4. **Visa webbplatsen:** Öppna din webbläsare till `http://localhost:5173`

⚠️ Viktigt: Tailwind CSS konfigurationsproblem

Nuvarande status: Vår Tailwind CSS-installation verkar ha konfigurationskonflikter och kanske inte fungerar optimalt. Vi har två konfliktande konfigurationsfiler (`tailwind.config.js` och `tailwind.config.ts`) och kodbasen visar stort beroende av inline-stilar istället för Tailwind-verktyg.

Bevis på problem:

- Dubbletter av Tailwind-konfigurationsfiler med olika tillvägagångssätt
- Stor användning av inline `style` -attribut istället för Tailwind-klasser
- Inkonsekvent färgsystemimplementation
- Saknar korrekt Tailwind v4-konfiguration

Om du vill hjälpa till att fixa detta: Detta skulle vara ett **högt påverkansbidrag**! Den ideala lösningen skulle vara:

1. **Konsolidera till en enda konfigurationsfil** (helst TypeScript)
2. **Implementera Tailwind v4 korrekt** med vårt designsystem
3. **Konvertera inline-stilar till Tailwind-verktyg** genom hela kodbasen
4. **Konfigurera korrekta CSS-anpassade egenskaper** för vår färgpalett
5. **Säkerställa att alla Tailwind-plugins fungerar korrekt**

Nuvarande workaround: Om du stöter på Tailwind-problem under utveckling:

```
# Prova denna fix först
npm uninstall tailwindcss postcss autoprefixer @tailwindcss/typography @tailwindcss/t
npm install -D tailwindcss postcss autoprefixer @tailwindcss/typography @tailwindcss/
npx tailwindcss init -p
```

För nya bidragsgivare:

- Du kan fortfarande bidra effektivt med inline-stilar (som ses i befintlig kod)
- Följ färgpaletten som definieras i konfigurationerna
- Vi kommer att migrera till korrekta Tailwind-verktyg när konfigurationen är fixad

Vill du ta dig an denna utmaning? Om du har erfarenhet av Tailwind CSS v4-konfiguration och skulle vilja hjälpa till att modernisera vårt stilsätt, vänligen:

- Gå med i vår **#dev-design** kanal på Discord (under 🤝 WORKSPACES kategori) för att diskutera tillvägagångssättet
- Öppna ett GitHub-ärende för att koordinera med andra bidragsgivare
- Dela dina idéer och få feedback innan du börjar

Detta skulle avsevärt förbättra vår utvecklingsupplevelse!

📁 Projektstruktur

Att förstå vår projektstruktur är avgörande för effektiva bidrag:

```
src/
├── lib/
│   ├── components/          # Återanvändbara Svelte-komponenter
│   │   ├── Header.svelte   # Huvudnavigation
│   │   ├── Footer.svelte   # Webbplatssidfot
│   │   └── ...             # Andra UI-komponenter
│   ├── content/            # Markdown-innehållsfiler
│   │   ├── frameworks/     # Ramverksdokumentation
│   │   ├── ggf-os/        # GGF Operating System docs
│   │   └── translations/   # Översättningsguider
│   ├── i18n/               # Internationalisering
│   │   ├── en/             # Engelska översättningar
│   │   ├── sv/             # Svenska översättningar
│   │   └── index.js        # i18n-systemkärna
│   ├── posts/              # Blogginlägg
│   └── utils/              # Verktysfunktioner
├── routes/                 # SvelteKit-sidor
│   ├── frameworks/         # Ramverkssidor
│   ├── blog/              # Bloggfunktionalitet
│   └── ...                # Andra sidor
└── app.html               # HTML-mall
```

Nyckelkataloger förklarade

- `src/lib/components/` : Återanvändbara UI-komponenter som används på hela webbplatsen
- `src/lib/content/` : Allt markdown-innehåll, organiserat efter typ och språk
- `src/lib/i18n/` : Översättningsfiler och internationaliseringslogik
- `src/routes/` : SvelteKits filbaserade routingsystem
- `static/` : Statiska tillgångar som bilder, PDF:er och nedladdningar

🛠️ Utvecklingsarbetsflöde

1. Konfigurera din utvecklingsmiljö

Skapa en ny gren för din funktion:

```
git checkout -b feature/your-feature-name
```

2. Göra ändringar

Lägga till nya sidor

För att lägga till en ny sida, skapa en `+page.svelte`-fil i lämplig `src/routes/`-katalog:

```
<!-- src/routes/your-new-page/+page.svelte -->
<script>
  import { t } from '$lib/i18n';
  // Din sidlogik
</script>

<svelte:head>
  <title>Din sidtitel</title>
</svelte:head>

<div class="container mx-auto px-4 py-8">
  <h1 class="text-3xl font-bold mb-6">{$t('yourPage.title')}</h1>
  <!-- Ditt innehåll -->
</div>
```

Lägga till komponenter

Skapa återanvändbara komponenter i `src/lib/components/`:

```
<!-- src/lib/components/YourComponent.svelte -->
<script>
  export let title = '';
  export let description = '';
</script>

<div class="component-container">
  <h2 class="text-xl font-semibold">{title}</h2>
  <p class="text-gray-600">{description}</p>
</div>

<style>
  .component-container {
    /* Dina stilar med Tailwind-klasser */
    background-color: white;
    border-radius: 0.5rem;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    padding: 1.5rem;
  }
</style>
```

Arbeta med markdown-innehåll

Lägg till nytt ramverksinnehåll i `src/lib/content/frameworks/`:

```
<!-- src/lib/content/frameworks/sv/your-framework.md -->
# Din ramverkstitel

## Introduktion

Din ramverksintroduktion...

## Grundprinciper

1. **Princip ett**: Beskrivning
2. **Princip två**: Beskrivning

## Implementation
```

```
Implementeringsriktlinjer...
```

3. Internationalisering

Lägga till översättningar

Lägg till översättningsnycklar till lämpliga JSON-filer:

```
// src/lib/i18n/en/common.json
{
  "header": {
    "yourNewItem": "Your New Item"
  }
}
```

```
// src/lib/i18n/sv/common.json
{
  "header": {
    "yourNewItem": "Ditt nya objekt"
  }
}
```

Använda översättningar i komponenter

```
<script>
  import { t } from '$lib/i18n';
</script>

<h1>{$t('common.header.yourNewItem')}</h1>
```

4. Stilriktlinjer

⚠️ **Nuvarande tillstånd**: Vårt stilsystem har vissa konfigurationsproblem som presenterar både utmaningar och möjligheter:

Nuvarande tillvägagångssätt

På grund av Tailwind-konfigurationskonflikter använder mycket av vår nuvarande kodbas inline-stilar. Även om det inte är idealt fungerar detta tillvägagångssätt och bibehåller konsekvens.

Färgpalett (använd dessa värden i stilar)

```
:root {
  --primary-blue: #2B4B8C;
  --secondary-purple: #6B5CA5;
  --earthy-green: #2D5F2D;
  --warm-gold: #DAA520;
  --dark-gold: #B8860B;
}
```

Nuvarande komponentstilmönster

```
<!-- Nuvarande tillvägagångssätt som används genom hela kodbasen -->
<div style="background-color: #2B4B8C; color: white; padding: 1rem; border-radius: 0
  Innehåll
</div>

<!-- Mål: Konvertera till Tailwind-verktyg (hjälp behövs!) -->
<div style="background-color: #1e293b; color: white; padding: 1rem; border-radius: 0
  Innehåll
</div>
```

Stilriktlinjer för bidragsgivare

För snabba bidrag:

- Följ befintliga inline-stilmönster
- Använd vår färgpalett konsekvent
- Behåll responsiv design med CSS media queries

För Tailwind-entusiaster:

- Hjälp oss att fixa Tailwind-konfigurationen (högt påverkansbidrag!)
- Konvertera inline-stilar till korrekta Tailwind-verktyg
- Implementera vårt designsystem som Tailwind-anpassade klasser

Responsivt designmönster

```
<div style="
  display: grid;
  grid-template-columns: repeat(1, minmax(0, 1fr));
  gap: 2rem;
">
  <!-- Mobil: en kolumn -->
</div>

<style>
  @media (min-width: 768px) {
    div {
      grid-template-columns: repeat(2, minmax(0, 1fr));
    }
  }
</style>
```

5. Testa dina ändringar

Lokal testning

```
# Kör utvecklingsserver
npm run dev

# Bygg för produktion (test)
npm run build

# Förhandsgranska produktionsbygge
npm run preview
```

Innehållsvalidering

- Säkerställ att alla länkar fungerar korrekt
- Verifiera att översättningar är kompletta
- Testa på flera skärmstorlekar
- Kontrollera tillgänglighet med skärmläsare

🎨 Designsystem

Typografiskala

- **Rubriker**: Använd semantiska rubriktaggar med lämplig stil
- **Brödtext**: Använd läsbara fontstorlekar för huvudinnehåll
- **Liten text**: Använd mindre storlekar för bildtexter och metadata

Avståndsriktlinjer

- **Sektioner**: Använd konsekvent vertikalt avstånd mellan sektioner
- **Komponenter**: Använd konsekvent intern utfyllnad
- **Element**: Använd konsekvent vertikal rytm

Layoutmönster

- **Container**: Maximal bredd med auto-marginaler och utfyllnad
- **Grid**: Använd CSS Grid eller flexbox för layouter

- **Kort**: Konsekvent skugga och border-radius-mönster

📝 Innehållsriktlinjer

Markdown bästa praxis

- Använd semantisk rubrikhierarki (H1 → H2 → H3)
- Inkludera tydliga sektionsavbrott
- Lägg till beskrivande alt-text för bilder
- Använd konsekvent formatering för kodblock

Ramverksdokumentationsstruktur

```
# Ramverkstitel

## Sammanfattning
Kort översikt och viktiga fördelar

## Grundprinciper
Grundläggande principer med förklaringar

## Implementeringsriktlinjer
Steg-för-steg implementeringsprocess

## Verktyg och resurser
Länkar till relevanta verktyg och material

## Fallstudier
Verkliga exempel och framgångshistorier

## Bilagor
Ytterligare tekniska detaljer och referenser
```

🔧 Avancerad utveckling

Lägga till nya ramverksvisualiseringar

1. Skapa SVG-tillgångar i `static/frameworks/your-framework/`
2. Lägg till interaktiva komponenter i `src/routes/frameworks/visuals/your-framework/`
3. Använd D3.js eller andra visualiseringsbibliotek efter behov

Anpassade komponenter med interaktivitet

```
<!-- Exempel på interaktiv komponent -->
<script>
  import { onMount } from 'svelte';
  import { browser } from '$app/environment';

  let data = [];
  let loading = true;

  onMount(async () => {
    if (browser) {
      // Ladda data och initiera komponent
      loading = false;
    }
  });
</script>

{#if loading}
  <div class="loading-spinner">Laddar...</div>
{:else}
  <!-- Ditt interaktiva innehåll -->
{/if}
```

Arbeta med statiska tillgångar

- **Bilder**: Placera i `static/images/`
- **Nedladdningar**: Placera i `static/downloads/`
- **Ramverksverktyg**: Placera i `static/frameworks/tools/`

🤝 Bidragsprocess

1. Planera ditt bidrag

Innan du börjar:

- Kontrollera befintliga ärenden och diskussioner
- Föreslå nya funktioner i GitHub-ärenden
- Koordinera med underhållare för stora ändringar

2. Utvecklingschecklista

- ☐ Kod följer projektkonventioner
- ☐ Alla översättningar är kompletta
- ☐ Responsiv design fungerar på alla enheter
- ☐ Tillgänglighetsstandarder uppfylls
- ☐ Innehåll är välstrukturerat och tydligt
- ☐ Länkar och navigation fungerar korrekt
- ☐ Inga konsolfel eller varningar

3. Skicka ditt bidrag

```
# Säkerställ att din gren är uppdaterad
git checkout main
git pull origin main
git checkout your-branch
git merge main

# Pusha dina ändringar
git push origin your-branch

# Skapa en pull request på GitHub
```

Pull request-riktlinjer

- **Titel**: Tydlig, beskrivande titel
- **Beskrivning**: Förklara vilka ändringar som gjordes och varför
- **Skärmdumpar**: Inkludera före/efter-skärmdumpar för UI-ändringar
- **Testning**: Beskriv hur du testade dina ändringar

📚 Resurser

Dokumentation

- SvelteKit-dokumentation
- Tailwind CSS-dokumentation
- Vår stilguide

Utvecklingsverktyg

- Svelte-tillägg för VS Code
- Tailwind CSS IntelliSense

Gemenskap

- GitHub-diskussioner
- Discord-gemenskap - Gå med i **#dev-design** för tekniska diskussioner

🔐 Säkerhet och bästa praxis

Säkerhetsriktlinjer

- Commita aldrig känslig data (API-nycklar, lösenord)
- Validera alla användarinmatningar
- Följ OWASP säkerhetsriktlinjer
- Använd miljövariabler för konfiguration

Prestanda bästa praxis

- Optimera bilder innan du lägger till dem
- Använd lazy loading för tunga komponenter
- Minimera bundle-storlek med tree-shaking
- Följ SvelteKits prestandarekommendationer

🔧 Integrera nya ramverk i webbplatsen

När ett ramverks innehåll (alla markdown-filer och tillgångar) har godkänts och integrerats måste utvecklare utföra följande steg för att göra det synligt och tillgängligt på webbplatsen.

1. Skapa ramverkssidor

För varje sektion av det nya ramverket, skapa motsvarande sidor i SvelteKit-projektet på:

```
/src/routes/frameworks/docs/implementation/[ramverk-namn]/[sektion-namn]
```

Följ det befintliga mönstret från ramverk som `aging` eller `treaty-for-our-only-home` för sidstruktur och load-funktioner.

2. Definiera unikt färgtema

Tilldela ett unikt färgtema för det nya ramverket för att särskilja det visuellt. Detta innebär att uppdatera webbplatsens temakonfiguration för att säkerställa att ramverket har sin egen visuella identitet samtidigt som konsistensen med vårt designsystem bibehålls.

3. Uppdatera delade resurssidor

Verktygsintegration: Om ramverket inkluderar nya verktyg, lägg till dem på huvudverktygssidan på `/src/routes/frameworks/tools`. Säkerställ korrekt kategorisering och beskrivning.

Visualiseringsintegration: Lägg till nya ramverksvisualiseringar på huvudvisualiseringssidan med lämplig kategorisering och navigation.

Nedladdningsintegration: Lägg till den fullständiga ramverkets PDF-nedladdningslänk på `/src/routes/downloads`-sidan och säkerställ korrekt filnamngivning och organisation.

4. Uppdatera informationssidor

Ordlisteuppdateringar: Om ramverket introducerar ny terminologi, lägg till termer och definitioner på ordlistsidan på `/src/routes/frameworks/docs/glossary`. Bibehåll alfabetisk organisation och korsreferenser.

Fallstudier: När ramverket implementeras i verkliga sammanhang, lägg till fallstudier på `/src/routes/frameworks/docs/case-studies`. Detta är en pågående uppgift som växer över tid.

Navigationsuppdateringar: Uppdatera webbplatsnavigationen för att inkludera det nya ramverket i lämpliga menyer och indexsidor.

5. Teknisk implementeringschecklista

- [ ] Ramverkssidor skapade med korrekt routing-struktur
- [ ] Load-funktioner implementerade enligt befintliga mönster
- [ ] Färgtema definierat och tillämpat konsekvent
- [ ] Verktyg integrerade på verktygssidan
- [ ] Visualiseringar integrerade på visualiseringssidan
- [ ] PDF-nedladdningar tillagda på nedladdningssidan

- [ ] Ny terminologi tillagd i ordlistan
- [ ] Navigation uppdaterad för att inkludera nytt ramverk
- [ ] Alla länkar testade och fungerar korrekt
- [ ] Responsiv design verifierad på alla enheter

🌱 Vad att bidra med

Områden med hög påverkan

1. 🎨 **Tailwind CSS konfigurationsfix** ⭐ **MEST BEHÖVD** ⭐
   - Lös konfigurationskonflikter mellan `.js` och `.ts`-filer
   - Implementera Tailwind v4 korrekt med vårt designsystem
   - Konvertera inline-stilar till Tailwind-verktyg genom hela kodbasen
   - Konfigurera korrekt CSS anpassade egenskaper-integration
2. **Ramverksintegration**: Hjälp till att integrera godkända ramverk i webbplatsinfrastrukturen
3. **Internationalisering**: Lägg till stöd för nya språk
4. **Interaktiva verktyg**: Skapa visualiserings- och simuleringsverktyg
5. **Tillgänglighet**: Förbättra webbplatstillgänglighet och användbarhet
6. **Prestanda**: Optimera laddningstider och användarupplevelse

Bra första ärenden

- **Fixa Tailwind CSS-konfiguration** (hög påverkan för erfarna utvecklare)
- Konvertera inline-stilar till Tailwind-verktyg
- Fixa trasiga länkar eller stavfel
- Lägg till saknade översättningar
- Förbättra mobil-responsivitet
- Lägg till alt-text till bilder
- Uppdatera utdaterade beroenden

Pågående behov

- **Tailwind CSS-modernisering** (brådskande teknisk skuld)
- Innehållsgranskning och redigering
- Nya ramverksimplementationer
- Gemenskapsverktyg och funktioner
- Dokumentationsförbättringar
- Testning och kvalitetssäkring

💡 Få hjälp

Om du behöver hjälp:

1. **Kontrollera dokumentation**: Granska denna guide och länkade resurser
2. **Sök ärenden**: Leta efter liknande frågor eller problem
3. **Gå med i vår Discord**: Anslut med utvecklargemenskapen
4. **Skapa GitHub-ärenden**: För buggar, funktionsförfrågningar eller frågor
5. **Parprogrammering**: Kontakta för gemensamma kodningssessioner

👥 Utvecklargemenskap

Discord-server: Gå med i vår utvecklargemenskap på https://discord.gg/Zx4hMJf4JU

- **#dev-design kanal** (under 🤝 WORKSPACES kategori) för tekniska diskussioner, kodgranskningar och arkitekturbeslut
- Realtidschatt med andra utvecklare och designers
- Röstkanaler för parprogrammeringssessioner och tekniska möten
- Snabb hjälp för installationsproblem och utvecklingsfrågor

GitHub-diskussioner: Använd repository-diskussioner för:

- Tekniska frågor och lösningar

- Funktionsförslag och arkitekturdiskussioner
- Pull request-koordination
- Dokumentationsfeedback

När att använda vilken kanal

- **Discord #dev-design**: Snabba frågor, realtidssamarbete, brainstorming
- **GitHub-ärenden**: Buggrapporter, funktionsförfrågningar, formella förslag
- **GitHub-diskussioner**: Fördjupade tekniska diskussioner, dokumentationsfrågor
- **Pull requests**: Kodgranskningar, implementationsdiskussioner

🎉 Erkännande

Vi värdesätter alla bidrag till Global Governance Framework-projektet. Bidragsgivare kommer att:

- Listas i vår bidragsgivarsektion
- Erkännas i releaseanteckningar
- Bjudas in till bidragsgivearevenemang och diskussioner
- Ges prioritetsstöd för sina egna projekt

Tack för att du bidrar till en mer sammankopplad och samarbetsinriktad värld genom globala styrningsramverk!

Behöver omedelbar hjälp? Gå med i vår Discord-gemenskap i **#dev-design** kanalen eller skapa ett ärende på GitHub.