Concurs de Cuina UOC — Documentació Marcel Prats Ustrell

1. Descarregar/clonar el repositori

Per obtenir el codi font del projecte, cal clonar el repositori de GitHub:

```
git clone https://github.com/marcelprats/pac3-marcel.git
cd pac3-marcel
```

2. Instal·lació de dependències

Aquest projecte utilitza Node.js. Un cop dins la carpeta, instal·la les dependències:

```
npm install
```

> **Nota:** Les dependències principals són Tailwind CSS, PostCSS, i Vite com a entorn de desenvolupament i build.

3. Entorn de desenvolupament local

Per treballar en local amb hot reload:

```
npm run dev
```

Això obre la web a `http://localhost:5173` (o port similar).

4. Estructura de carpetes

- `components/`: components reutilitzables (`_card.html`, `_cta.html`, `_testimonial.html`)
- `views/`: fitxers de seccions globals (ex: `footer.html`)
- `assets/styles/`: estils (només utilitzats per configurar Tailwind i personalitzar colors/fonts)
- `assets/images/`: imatges del projecte
- `assets/scripts/`: JS mínim
- `index.html`: pàgina principal

5. Compilació per a producció

Per generar la versió optimitzada i minificada per a desplegament:

```
npm run build
```

Els fitxers es generen a la carpeta `dist/`.

6. Publicació a Netlify

1. Puja el repositori a GitHub.
2. Ves a Netlify, crea un nou site connectant el teu repo.
3. Configura el build command (`npm run build`) i el directori de publicació (`dist`).
4. Publica i comprova el resultat a l'URL proporcionada.

> **Resultat:** La web està publicada i enllaçada amb GitHub; cada push actualitza la web automàticament.

7. Justificació de decisions de desenvolupament

- **Tailwind utility-first:**
  He aplicat la metodologia utility-first CSS a tot el projecte. Totes les classes d'estil (espaiats, colors, tipografies, efectes) estan posades directament als components HTML, sense codi duplicat i facilitant la mantenibilitat. Només he utilitzat SCSS per definir colors propis i la font UOC a la configuració de Tailwind.
- **Mobile first:**
  El disseny està pensat mobile first, aprofitant les utilitats `sm:`, `md:` etc. de Tailwind per adaptar-se a qualsevol pantalla.
- **Components reutilitzables:**
  Tots els blocs (targetes de recepta, CTA, testimonial) són components independents. Si cal afegir-ne més, només cal fer més includes amb les dades corresponents.
- **Colors i tipografia:**
  S'ha personalitzat la paleta de colors a Tailwind per adaptar-la a la identitat UOC. La font s'ha afegit globalment.
- **Efectes i interactivitat:**
  S'ha afegit transició i efecte "zoom" a les targetes amb Tailwind (`hover:scale-105` i `active:scale-95`) per millorar la UX sense JS.
- **Accessibilitat:**
  S'han afegit textos alternatius a les imatges, focus visible als enllaços/botons, i estructura semàntica clara. Contrast revisat.

8. Preguntes específiques sobre utility-first CSS

1. Avantatges d'usar utility-first CSS amb Tailwind?

- Permet prototipar ràpidament sense sortir de l'HTML.
- Elimina codi mort o duplicat: cada utilitat només s'aplica on cal.
- Facilita manteniment i refactorització: canvis d'estil sense buscar en grans fulls CSS.
- Fomenta la coherència visual en tot el projecte.
- El build optimitza el CSS i només manté les utilitats usades.

2. Com s'organitza el codi amb utility-first CSS?

- L'estil es posa directament als components, no en fulls CSS separats.

- Si una combinació d'utilitats es reutilitza molt, es pot fer una classe amb `@apply` (ex: `.btn-primari`).
- Els fitxers són més llegibles: veus l'estil directament amb l'HTML.

3. Problemes més habituals amb utility-first i solucions:

- **HTML llarg:** Es compensa amb la llegibilitat i l'evitació de duplicats.
- **Classes molt llargues:** Es pot refactoritzar amb components o mini-classes amb `@apply` si cal.
- **Aprenentatge inicial:** Un cop entès, és molt més àgil que CSS tradicional.

9. Anàlisi del codi generat per la IA i adaptació

Per a la pàgina de participants (`contestants.html`), s'ha generat el codi amb IA. Aquesta pàgina es va desenvolupar a partir del disseny de Figma proporcionat pel professorat, però la implementació concreta (estructura, classes, estilització responsive, etc.) es va generar automàticament amb IA.

- **Tecnologia utilitzada:** HTML i CSS tradicional (no Tailwind per aquesta pàgina).
- **Estructura:** La disposició de columnes, el layout i els breakpoints responsius es corresponen fidelment a Figma.
- **Adaptació:** S'han fet retocs mínims al codi generat (canvi de textos en els links, petits ajustos visuals) però la base i l'estil general provenen de la sortida de la IA (Copilot).
- **Valoració:** El codi generat per la IA és clar, net i fàcil de modificar, i s'adapta bé a requeriments de disseny reals.

> Les altres pàgines del projecte (inici, etc.) s'han maquetat manualment amb Tailwind CSS, seguint l'estil de uoc Boiletplate.

10. Relació entre Figma i la implementació

- **Participants (`contestants.html`):**
  - Basada en el disseny de Figma, però la implementació concreta (estructura, estilització i responsabilitat) s'ha fet a partir de codi generat amb IA.
  - El resultat manté el layout, colors i jerarquia visual del Figma, amb adaptacions puntuals per a l'accessibilitat, la llengua i la integració amb la resta del projecte.
  - Aquesta pàgina utilitza HTML i CSS escrit a mà (no Tailwind), seguint el patró clàssic de maquetació responsive i columnes.
- **Resta de pàgines:**
  - Maquetades manualment, utilitzant Tailwind CSS i la metodologia utility-first.
  - S'han seguit les pautes de Figma i la paleta corporativa, però sense cap ajuda d'IA en la generació del codi.
  - Els components (targetes, cta, testimonis, etc.) estan fets amb codi propi, aplicant les utilitats de Tailwind per aconseguir un resultat coherent i mantenible.

11. Resultats

- Projecte complet, responsiu, modular i accessible.
- Codi net, ben estructurat i documentat.
- Publicació i desplegament automatitzats.

12. Enllaços

- **Repo GitHub:** https://github.com/marcelprats/pac3-marcel
- **Web publicada a Netlify:** https://pac3-marcel.netlify.app/

13. Extracció de classes i components

- He extret dues classes reutilitzables a Tailwind via `@apply`:
  - `.btn-primari` (per als botons principals)
  - `.card-base` (per a la base de les targetes de recepta)
- També he extret dos components reutilitzables amb `posthtml-include`:
  - `_card.html` (targeta de recepta)
  - `_cta.html` (bloc de crida a l'acció)

14. Comparativa: CSS semàntic vs utility-first CSS

- A la PAC 2, utilitzava CSS semàntic amb classes específiques per cada component i un full d'estils a part. Això feia que el codi fos més llarg i amb més repetició d'estils comuns.
- Ara, amb utility-first CSS (Tailwind), l'estil es posa directament a l'HTML, cosa que m'ha permès detectar i reescriure patrons repetits, fer el codi més net i mantenible i detectar més ràpidament incoherències visuals.
- El procés de desenvolupament ha estat més àgil amb Tailwind, tot i que al principi el codi HTML sembla més carregat de classes.