

Convencions DS 2023

Oficina Tècnica UX/UI

Document de convencions de desenvolupament del Design System 2023

1. Git Flow

Git Flow és un model alternatiu de creació de branques a Git en el qual s'utilitzen branques de funció i diverses branques principals. En realitat, Gitflow és una mena d'idea abstracta d'un flux de treball de Git. Això vol dir que ordena quin tipus de branques s'han de configurar i com fusionar-les.

En lloc d'una única branca mestra, aquest flux de treball fa servir dues branques per registrar l'historial del projecte. La branca mestra o principal emmagatzema l'historial de publicació oficial i la branca develop o de desenvolupament serveix com a branca d'integració per a les funcions.

1.1 Branques

La creació i estructura del sistema de branques es fa en base a la Git Branch Naming Convention. Les branques es classifiquen en 2 categories:

- **Branques regulars:** són aquelles branques que estan permanentment disponibles en el repositori.
- **Branques temporals:** són les branques que es creen i es deixen anar segons el requisit, normalment quan es mergea amb una branca regular. No són presents permanentment en el repositori.

1.1.1 Branques regulars

Les nostres branques regulars són:

- La branca màster és la branca predeterminada en el repositori de Git i conté la versió que es desplega per a producció. Ha de ser estable en tot moment i abans de qualsevol merge, qualsevol codi de la branca develop a la branca màster ha de sotmetre's a proves i verificacions.

- La branca `develop` és la que s'utilitza per desenvolupar qualsevol funció nova i conté l'historial complet del projecte. Les noves branques que es creen a partir de la branca `develop`, i aquestes solen ser branques temporals.

1.1.2 Branques temporals

Les branques temporals de Git són les branques que es creen i es deixen anar segons el requisit. No són presents permanentment en el repositori. Es classifiquen en 5 blocs, però en el nostre cas només tindrem en compte les Bug Fix i les Feature Branches.

- **Feature:** són branques que serveixen per afegir, refactoritzar o eliminar una funcionalitat.
- **Bug Fix:** són branques que serveixen per arreglar un error, generalment d'una funcionalitat ja creada.

Algunes de les convencions que apliquem en la creació de les branques, són:

- Iniciar el nom de la branca amb la paraula del grup de classificació, en el nostre cas `feat` o `fix`.
- Utilitzar barra inclinada (/) com a separador entre el nom de grup i el nom de la funcionalitat o error.
- Utilitzar noms únics per a la funcionalitat o l'error. Si aquest nom està compost per diverses paraules, les separarem amb un guió normal (-).

Després del nom del grup de classificació de la branca, ha d'haver-hi una "/" seguit del nom que descriu la funcionalitat o l'error que es vol dur a terme. Si aquest nom està compost per diferents paraules, es separen amb un guió normal "-". Si no hi ha referència, simplement s'afegeix `no-ref`.

1.3 Com crear commits

Els commits són la instantània dels canvis dels arxius per a una branca i moment del projecte. Els missatges de commit són una forma de documentar els canvis que es realitzen en un repositori de codi. Aquests missatges són molt útils per comprendre quins canvis es van realitzar en un projecte i per què es van fer. Com és molt important entendre quins canvis s'han fet, utilitzem també una convenció, concretament la Conventional Commits.

Hi ha diferents tipus de commits, però nosaltres, com que només tindrem 2 tipus de branques temporals, només utilitzarem aquells commits de tipus feat i fix.

L'estructura comença amb el tipus de commit seguit del nom de la funcionalitat o de l'error entre parèntesis. Per introduir el missatge descriptiu, introduïm ":".

Els missatges dels commits sempre els introduirem en anglès.

- **feat(nom-funcionalitat)**: missatge descriptiu en anglès
- **fix(nom-arreglament)**: missatge descriptiu en anglès

2. Estructura i bones pràctiques

2.1 Llibreries utilitzades en aquest projecte

- **LIT ELEMENT**: és un marc base per a la creació de components, el que coneixem generalment per Web Components. Està basada en Javascript estàndard i funciona recolzant-se en característiques natives dels navegadors. A més, com que és Javascript natiu, es poden crear components que es podran utilitzar en aplicacions com Angular, React, VueJS, etc.

La major avantatge d'utilitzar aquest marc és la interoperabilitat dels components web a més que és simple per al desenvolupament de components i lleuger, creant paquets petits i temps de càrrega curts.

- **SASS**: és un pre-processador de CSS. Ens permet generar fulls d'estil de forma automàtica, afegint-los característiques que no té CSS, i que són pròpies dels llenguatges de programació, com poden ser variables, funcions, selectors aniuats, herència, etc.
- **STORYBOOK**: és una eina per generar documentació de components UI, i ofereix la possibilitat tant de desenvolupament com el testing de components. Permet la visualització dels components d'una manera organitzada, interactuar amb ells de forma dinàmica i testar-los com si es trobessin desplegats en una aplicació real.
- **VISUAL STUDIO CODE**: és un editor de codi font desenvolupat per Microsoft. És programari lliure i multiplataforma, està disponible per a Windows, GNU/Linux i macOS. El VS Code té una bona integració amb Git, compta amb suport per a depuració de codi, i disposa d'un gran nombre d'extensions, que bàsicament et donen la possibilitat d'escriure i executar codi en qualsevol llenguatge de programació.

- **NODE JS:** és un entorn de codi obert i multiplataforma per executar JavaScript fora del navegador. Es fa servir per desenvolupar aplicacions escalables del costat del servidor i de xarxa. En aquest projecte és necessari fer servir la versió V.16.16.0 per arrencar el projecte i evitar conflictes.

3. Convencions

Les convencions són bàsicament un conjunt de directrius que recomanen un estil, pràctiques i mètodes de programació per a cada aspecte del projecte. Per això, hi ha diferents àrees a tenir en compte.

3.1 Folders

Per crear una carpeta, sempre s'escriurà en minúscula i en cas de ser necessari, separades per guions normals “-”.

Per classificar els elements que conformen el projecte, hem utilitzat el sistema d'Atomic Design. És una metodologia de disseny basada en compondre i dissenyar interfícies digitals a partir d'elements replicables i modulars.

En l'Atomic Design existeixen 5 nivells, però nosaltres només utilitzarem els 3 primers ja que el nostre projecte no en necessita més:

1. Àtoms
2. Molècules
3. Organismes

Basant-nos en aquests nivells, estructurarem els fitxers.

3.1.1 SASS

Tenim la següent estructura:

- **Tokens:** on definim les variables de tipografies, icones, colors, etc.
- **Àtoms:** aquells elements més bàsics com pot ser un títol.
- **Molècules:** és la unió de diversos àtoms, per exemple, un input amb un label.
- **Organismes:** serien una selecció més complexa en la qual podem trobar molècules definides i àtoms, com per exemple un header o un footer.

3.1.2 Històries

En aquesta carpeta emmagatzemem els arxius necessaris per crear la documentació dels elements a Storybook i segueixen l'estructura d'Atomic Design:

- **Àtoms:** aquells elements més bàsics com pot ser un títol.
- **Molècules:** és la unió de diversos àtoms, per exemple, un input amb un label.
- **Organismes:** serien una selecció més complexa en la qual podem trobar molècules definides i àtoms, com per exemple un header o un footer.

3.1.3 Actius

Aquest és el lloc on emmagatzemem imatges, icones, tipografies, etc. Els actius sempre s'escriuen en minúscula i separats per guionet “-”, així com la nomenclatura de les seves carpetes.

3.2 SASS

Els arxius d'estils estan a la carpeta scss i s'estructuren en base a Atomic Design. Els arxius es creen amb:

nom-del-component.scss

Tots els arxius s'importen a styles.scss, tot i que pot donar-se el cas que una molècula o organisme importi directament en el seu propi arxiu.

Les classes es structuren a través de la convenció BEM. BEM és un estàndard o convenció que es va crear per a nomenar les classes CSS dels elements d'un lloc web. L'objectiu d'utilitzar BEM és facilitar la lectura i comprensió de l'ús que té cada classe que s'hi afegeix a un element, d'aquesta manera s'eviten duplicats en les propietats de dues classes o escriure codi de més.

BEM són les sigles de Bloc-Element-Modificador, que es resumeixen de la següent manera:

- **Bloc** és el contenidor principal del component que s'està creant. Modal, card, header, footer, menú, dropdown...
- **Element** són les parts del bloc. Dins d'un bloc hi pot haver molts elements que conformen l'estructura del mateix. Icona, text, element, imatge, botó, radio, entrada, enllaç...

- **Modificador** són les diferents variants que pot tenir un bloc o element. Actiu, ocult, blau, gran, dret, esquerra, petit.

Per tant, la sintaxi que fem servir és:

- Prefix del projecte [dss]
- dss-[bloc]
- dss-[bloc]_[element]
- dss-[bloc--[modificador]]
- dss-[element]--[modificador]
- dss-[bloc]_[element]--[modificador]

3.3 Elements Lit

Els components creats estan a la carpeta components i els arxius es creen amb: nom-del-component.ts

Els components estan realitzats en typescript i els estils del component van encapsulats en aquest mateix document a través de la funció get styles(). En aquest bloc, fem servir les següents convencions per als noms:

Classes

- Primera lletra en majúscules.
- Substantius i singulars.
- Si està composta per més d'una paraula, fer servir pascal case (cada paraula comença amb majúscules).
- Noms simples i descriptius.
- Les classes privades sempre comencen per barra baixa “_”.

Variables

- Primera lletra en minúscula.
- No poden començar per números (Error: Syntax error on token).
- Si està conformada per dos paraules, la primera d'aquestes haurà d'estar en minúscula i la segona en majúscula o ambdues separades per guionet (camel case, underscore).
- És recomanable evitar noms de una sola lletra, amb excepció de variables temporals (i, j, m, n).
- No han de començar amb guionet baix _ o signe dòlar \$ (no és obligatori, Java ho permet).
- No han de començar amb caràcters estranys */+- (Error: cannot be resolved to a variable).

Classes CSS

- Prefix "dss-".
- Segueix les mateixes convencions per als noms que les variables.
- Segueix la convenció BEM per a nomenar les concatenacions de blocs, elements i modificadors.

Component

- Prefix "dss-".
- Segueix les mateixes convencions per als noms que les classes CSS.

3.4 Storybook

Els fitxers de Storybook s'anomenen "stories", i estan a la carpeta "stories". S'estructuren en base a Atomic Design. Els fitxers es creen amb:

Nom-del-component.stories.mdx

3.5 Prettier

Prettier és un formatejador automàtic de codi, i el fem servir com una extensió de Visual Studio Code. L'apliquem a cada fitxer abans de pujar-lo al repositori, i tenim un fitxer propi d'aquest formatejador en el projecte per indicar-li les regles que volem que apliqui.

Per posar-ho en pràctica:

1. Instal·lar l'extensió de Prettier si no la tenim
2. Configurar Prettier com a formatejador predefinit
3. Possiblement en la majoria de programes per escriure codi, aparegui en el menú del botó dret, amb "Donar format al document".

3.6 EsLint

És una eina que serveix per examinar el codi que es vol escriure, i avisa quan detecta errors de sintaxi, codi incorrecte o males pràctiques. En el nostre projecte fem servir l'EsLint, en concret l'extensió standard scss.

3.7 Husky

És una eina que ens permet executar Git Hooks de forma més senzilla. Els Git Hooks són scripts que s'executen en moments determinats com abans d'un commit, un push, etc. D'aquesta forma garantim que es corregeixen les proves pertinents en el codi i d'aquesta forma no enviem un bug o inconsistències al repositori del projecte.

En aquest projecte, s'ha establert executar Husky abans dels commits, i avaluarà que passi el filtre d'EsLint.

4. Referències

- <https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow>
- <https://www.scaler.com/topics/git/git-branch-naming-conventions/>
- <https://dev.to/varbsan/a-simplified-convention-for-naming-branches-and-commits-in-git-il4>
- <https://www.conventionalcommits.org/en/v1.0.0/>
- <https://carlosazaustre.es/conventional-commits>
- <https://lit.dev/>
- <https://sass-lang.com/>
- <https://nodejs.org/es>
- <https://code.visualstudio.com/>
- <https://getbem.com/naming/>
- <https://storybook.js.org/docs/6.5/web-components/get-started/install>
- <https://storybook.js.org/docs/6.5/web-components/get-started/whats-a-story>
- <https://eslint.org/docs/latest/use/getting-started>
- <https://www.npmjs.com/package/stylelint-config-standard-scss>
- <https://typicode.github.io/husky/#/>
- <https://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks>

/Salut  **Generalitat
de Catalunya**