MWS [Svelte-WP-display] v-0.0.5

FR : moonkiosk.monwoo.com/produit/mws-svelte-wp-display/

Démo : mws.monwoo.com/demos/svelte-wp-display

LICENSE terms

Thoses source codes are part of researches and developments done by Miguel Monwoo.

They are transferred to you subject to respect of associated copyright and confidentiality.

Please respect professional secrecy by not disclosing thoses source codes without written consent from Miguel Monwoo.

Following your purchase on moonkiosk.monwoo.com, a use for one domain name is granted to you.

Unlike the source code, you may distribute the production build as you wish, for private or commercial use, as long as the sources remain confidential with attribution to Miguel Monwoo as 'participant' at minimum.

This private license disallow usage of the provided sources outside of the confidential developments necessary for a single domain name indicated during your purchase.

For a completely free license, you can use our custom development services by paying for our time spent creating a new personalized starter for your aims at the price rates in effect on moonkiosk.monwoo.com (or via service@monwoo.com).

Whatever obtained license (Limited, Open or illegal), by using or getting helps from thoses codes, you must credit the author for his inalienable moral rights for heping you or being used by you : Miguel Monwoo (miguel.monwoo.com)

You can sell this source code AS IS within your products on other domains by buying a new licence per domain names for your target products and requiering same kind of usage for your products.

You can also use this starter for confidential educational purpose with no rights to duplicate.

Contact us if you plan to buy a huge amount of licenses : service@monwoo.com.

Opened files for this starter can be used under Apache-2.0 lisence if present under :

- github.com/Monwoo/web-starters-free
- github.com/Monwoo/web-starters-free/blob/main/apps

Supports :

- You can use regular features of : github.com/Monwoo/web-starters-free/labels/mws-svelte-wp-display
- To help us open more software, send a subvention : www.monwoo.com/don

Quick usage

```
# Ensure your node version :
node -v
# Should be 'v18.4.0' or higher.

# Configure your install with targeted MWS starter :
node mws-install.mjs wp-display --with-deps

# run coverage, functional and end to end tests
pnpm run test

# YATA, YOUR PRODUCTION BUILD is in ./build folder.
# Indeed, e2e tests did build it with your local .env
alias code="/Applications/Visual Studio Code.app/Contents/MacOS/Electron"
# Need to change CONFIG and re-build ?
code .env

# Then we advice you to re-do the full tests
# But if you are in a hurry, sure of what you do,
# simply launch the build :
pnpm run build
```

Installations and builds

```
# Ensure your node version :
node -v
# Should be 'v18.4.0' or higher. Not tested yet for versions below
v18.4.0.

# Configure your install with targeted MWS starter :
node mws-install.mjs wp-display --with-deps

# generate smui styles (optionnal, do on each SMUI or SCSS theme changes)
pnpm run prepare

# test in dev
pnpm run dev

# launching unit tests any time you like and before each commit.
# do branch if want to commit faling code
# instead of the starter production branches...
pnpm run test:unit

# Run coverage tests with web live reports
# of unit tests launched in watch mode ;)
pnpm run test:coverage:watch -- --ui
```

```
# Open the coverage report in your browser :
open coverage/index.html

# Run functional tests
pnpm run test:functional

# Run end to end tests
pnpm run test:e2e

# run coverage, functional and end to end tests
pnpm run test
# build for production
rm -rf .svelte-kit
pnpm run build

# Testing the build with local server :
pnpm run preview

# Testing the build with MAMP (configured with self signed SSL) :
mkdir /Applications/MAMP/htdocs/demos
ln -s "$PWD/build" /Applications/MAMP/htdocs/demos/svelte-wp-display
/Applications/Google\ Chrome.app/Contents/MacOS/Google\ Chrome \
 --user-data-dir=/tmp/foo --ignore-certificate-errors \
 --unsafely-treat-insecure-origin-as-secure=https://localhost \
 https://localhost/demos/svelte-wp-display

# If you use GIT, save multi-branche logs :
git log --branches --tags --remotes --full-history --date-order \
--format='%ai;%f;%h' > git-logs.csv

# If you have global turbo from npm, remove it :
npm r -g turbo

# Setup you global pnpm bin repo folder :
pnpm setup
source ~/.zshrc
# If you use turbo repo, install it globally with 'pnpm'
pnpm install turbo --global
pnpm install # run it in your turbo repo root folder

turbo --version # will run with turbo version from local node_modules
```

Add some unit tests

Starting from version 0.0.5, we ensure 100% coverage per release builds :

```
############################
# Unit testings will focus on small parts of code

# Create a file in 'src' folder with '.test.ts' extension
touch ./src/components/wp-display/BuyThisStarter.test.ts

# Get inspiration from existing tests :
find . -type f  -name '*.test.ts' -o \
\( -path ./node_modules -o -path ./tests \) \
-prune -false | sort | sed 's,\./, - ,g'
```

- src/components/core/configs/FontCustomizer.test.ts
- src/components/core/configs/LangSwitch.test.ts
- src/components/core/configs/ThemeToggler.test.ts
- src/components/wp-display/AppLogo.test.ts
- src/components/wp-display/AppMenu.test.ts
- src/components/wp-display/ui/BlogItem.test.ts
- src/components/wp-display/ui/BlogItemFullPage.test.ts
- src/components/wp-display/ui/BlogItemHeroCard.test.ts
- src/components/wp-display/ui/BlogItemHeroCardLight.test.ts
- src/components/wp-display/ui/BlogItemSummary.test.ts
- src/components/wp-display/ui/BlogList.test.ts
- src/components/wp-display/ui/Caroussel.test.ts
- src/components/wp-display/ui/CarousselWithGsap.test.ts
- src/components/wp-display/ui/CarousselWithSplide.test.ts
- src/components/wp-display/ui/DonateCard.test.ts
- src/components/wp-display/ui/MenuItem.test.ts
- src/components/wp-display/ui/Steps.test.ts
- src/components/wp-display/ui/Title.test.ts
- src/routes/(core)/[[i18n=i18n]]/mws/core/page.test.ts
- src/routes/(wp-display)/[[i18n=i18n]]/mws/wp-display/[t_contact=i18n_contact]/page.svelte.test.ts
- src/routes/(wp-display)/[[i18n=i18n]]/mws/wp-display/[t_missions=i18n_missions]/page.svelte.test.ts
- src/routes/(wp-display)/[[i18n=i18n]]/mws/wp-display/[t_newsletter=i18n_newsletter]/page.svelte.test.ts
- src/routes/(wp-display)/[[i18n=i18n]]/mws/wp-display/[t_visualIdentity=i18n_visualIdentity]/page.svelte.test.ts
- src/routes/(wp-display)/[[i18n=i18n]]/mws/wp-display/error.svelte.test.ts
- src/routes/(wp-display)/[[i18n=i18n]]/mws/wp-display/page.svelte.test.ts
- src/service-worker.test.ts
- src/services/gc.test.ts
- src/services/init-i18n.test.ts
- src/services/logger.test.ts
- src/services/wp-display/request.test.ts

- src/stores/LocalStore.db.test.ts
- src/stores/app-config.browser.test.ts
- src/stores/app-config.browser.with-db.test.ts
- src/stores/app-config.test.ts
- src/stores/app-config.wp-display.test.ts
- src/stores/loaders.wp-display.test.ts
- src/stores/transformers.wp-display.test.ts

```
# We use vitest to launch and show coverage reports
# Use the watch mode if you're in building phase :
pnpm run test:unit:watch
# Watch mode for coverage score :
pnpm run test:coverage:watch
pnpm run test:coverage:watch src/components/wp-
display/ui/CarousselWithGsap.test.ts
```

Add some functional tests

```
############################
# Test technico-functionnal features with thoses tests.
# It reflect all past and forseen functionalities found by the dev team.
# As for unit tests, English is the common language for thoses.
#
# Get inspiration from existing tests inside 'tests/functional' :
find ./tests/functional -type f  -name '*.test.ts' \
| sort | sed 's,\./, - ,g'
```

- tests/functional/core/i18n-routings.test.ts
- tests/functional/core/lang-switch.test.ts
- tests/functional/core/theme-storage.test.ts
- tests/functional/wp-display/i18n-routings.test.ts
- tests/functional/wp-display/service-worker.test.ts

```
# Run specific test :
test:functional -- tests/functional/core/theme-storage.test.ts

# BE SURE to have playwrite installed (it will warn you if not :)
npx playwright install

# Debug functional tests :
PWDEBUG=1 pnpm run test:functional tests/functional/wp-display/service-
worker.test.ts

# link build to MAMP :
mkdir /Applications/MAMP/htdocs/demos
ln -s "$PWD/build" /Applications/MAMP/htdocs/demos/svelte-wp-display
ls /Applications/MAMP/htdocs/demos/svelte-wp-display

# Debug functional with MAMP SSL build :
PWDEBUG=1 pnpm run test:functional:mamp tests/functional/wp-
display/service-worker.test.ts

# Run chrome for service worker debugs with chrome :
/Applications/Google\ Chrome.app/Contents/MacOS/Google\ Chrome \
 --user-data-dir=/tmp/foo --ignore-certificate-errors \
 --unsafely-treat-insecure-origin-as-secure=https://localhost
```

Add some end to end tests

```
############################
# Test commercial features with thoses tests
#
# We use codeception to write tests in our CLIENT'S language
# So we review the business functionality with the CLIENT'S language
#
# Get inspiration from existing tests inside 'tests/e2e' :
find ./tests/e2e -type f  -name '*.test.ts' \
| sort | sed 's,\./, - ,g'
```

- tests/e2e/core/main-page.test.ts
- tests/e2e/wp-display/visual-identity.test.ts

```
# Run specific test :
test:e2e -- tests/e2e/core/main-page.test.ts

# if you change languages from tests/e2e/translations
# re-generate types with :
cd tests/e2e
npx codeceptjs def -c codecept.conf.cjs

# Then remove 'J' from imports and declare 'J' like 'Je' :
# interface J extends WithTranslation<Methods> {}
```

Handle languages and translations

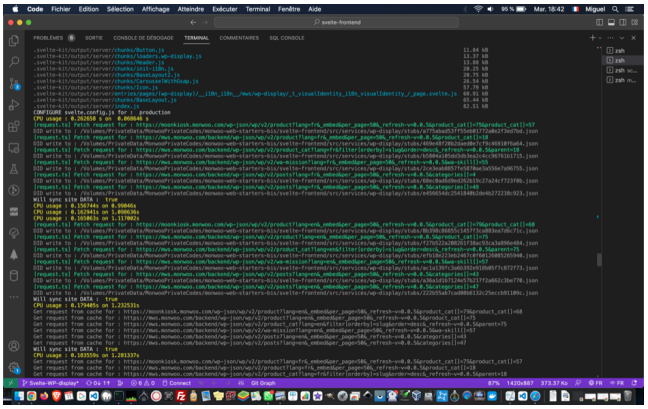
```
############################
# ADVICE for VSCode, use nice translation extension like :
# https://github.com/lokalise/i18n-ally
alias code="/Applications/Visual Studio Code.app/Contents/MacOS/Electron"

# If you get error :
#   "Cannot format a message without first setting the initial locale."
# => MEANS you did update the 'locale' from locale.set() with null value
# OR that import initI18n from '@app/services/init-i18n';
# initI18n(); is missing BEFORE translation try....

# list locales files :
ls src/locales

# if no file, mws-install did fail (run on `pnpm install`).
# Sources are in : src/mws-configs/wp-display/locales/

# TIPS : USE npm instead of pnpm for LOCAL FOLDER installation,
# to avoid the command from the root workspace
# To load our starter configs to your app, run :
npm run mws:install:wp-display

# BUT we advise you to build your own starter by duplicatings the files
# and then configure what to keep or change from it for your new starter
cp mws-install.wp-display.mjs mws-install.my-new-starter.mjs

# After duplication or change of your configs, you can now run :
npm run mws:install -- my-new-starter
# or dump back to src config for backup of dev config changes
npm run mws:install -- my-new-starter --dump-to-src

# i18n source usages :
# https://github.com/kaisermann/svelte-i18n/blob/main/docs/Svelte-Kit.md
# https://github.com/kaisermann/svelte-i18n

# setup starter languages and failback defaults :
code src/services/init-i18n.ts

# setup language Switcher defaults, cf :
# allLocales in locale.subscribe(...) listener
code src/components/core/configs/LangSwitch.svelte

# translate from source code with i18n-ally extension :
# https://github.com/lokalise/i18n-ally
code src/routes/(app)/+page.svelte
```

Be carfull with translations, they must be inside ONE html tag to work nicely in first load of pre-rendered html.

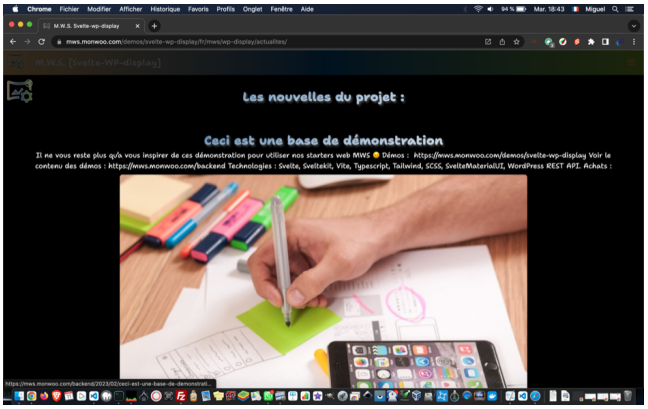
```
// WRONG : even if will work in dev,
// production build will not hydrate for first load....
<div class="p-7">
    {@html $t('mws.core.routes.page.text_2')}
    <p class="p-1">src/routes/(core)/mws/</p>
</div>

// OK, will work in DEV and PRODUCTION build :
<div class="p-7">
    <span>{@html $t('mws.core.routes.page.text_2')}</span>
    <p class="p-1">src/routes/(core)/mws/</p>
</div>
```

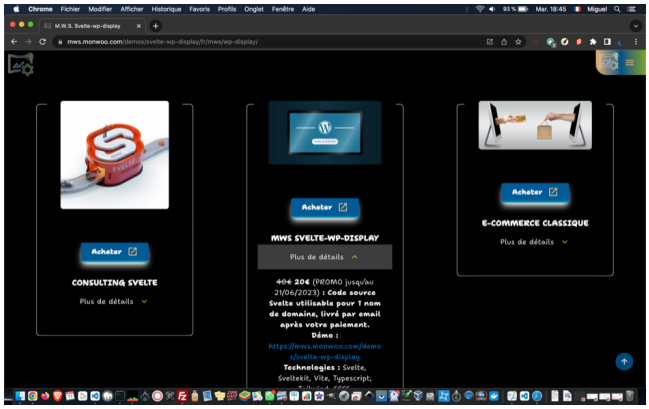
User paths

01-A - Build from fresh server source if available

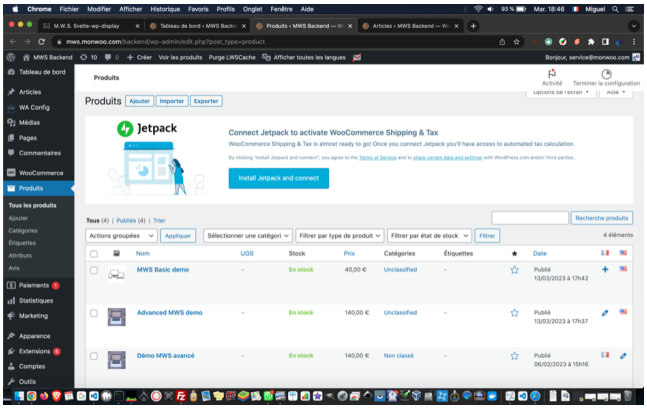


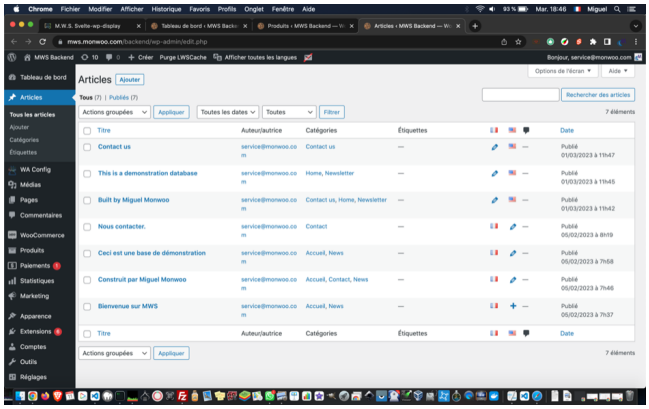
01-B - Show updated values from final build preview



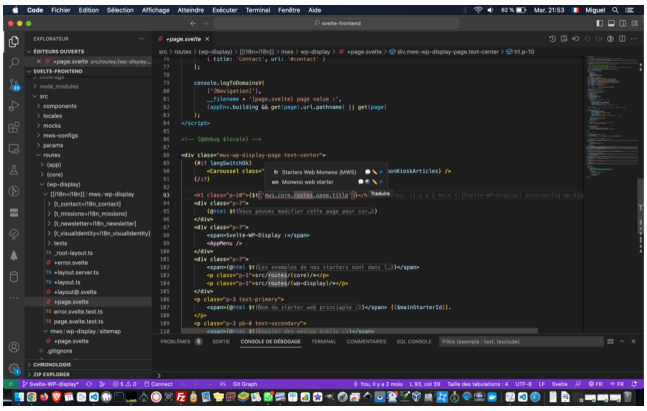


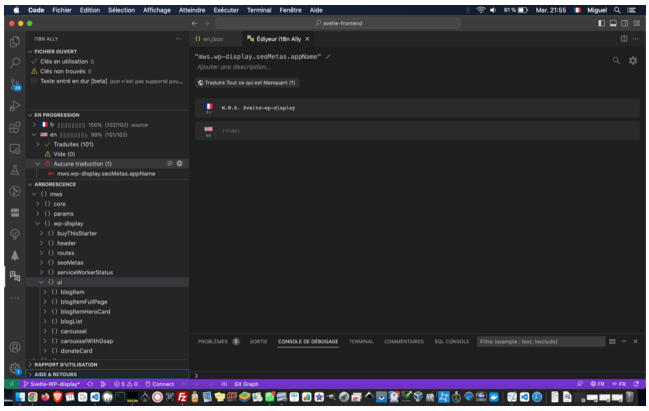
01-C - Edit data from backend to update frontend pre-rendered build



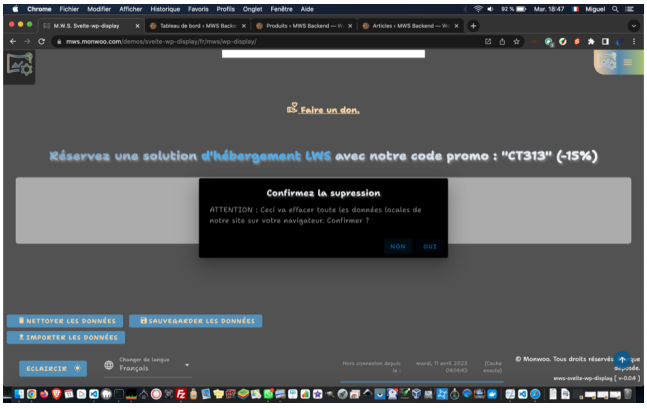


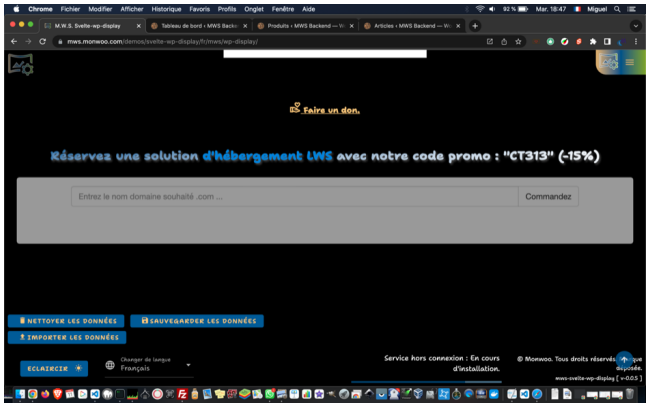
02 - Edit translations



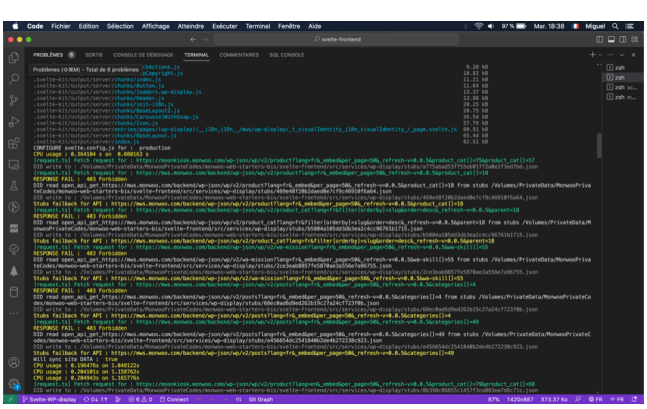


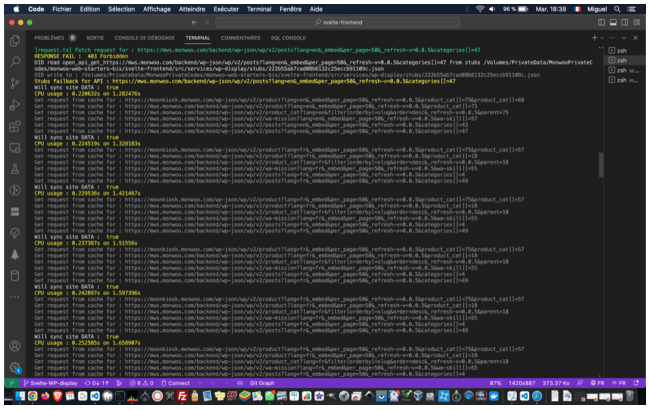
03 - Use production website offline



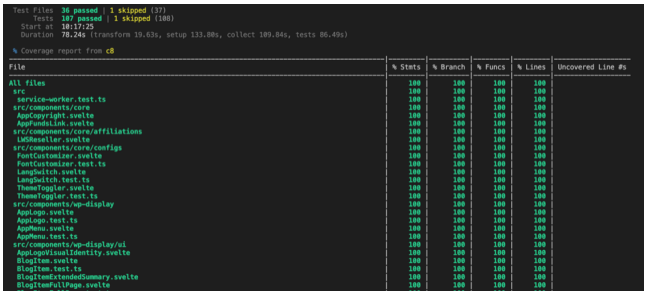


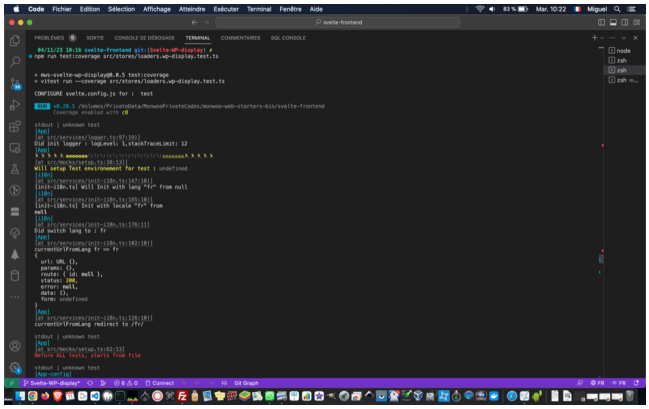
04 - Use source developpement offline

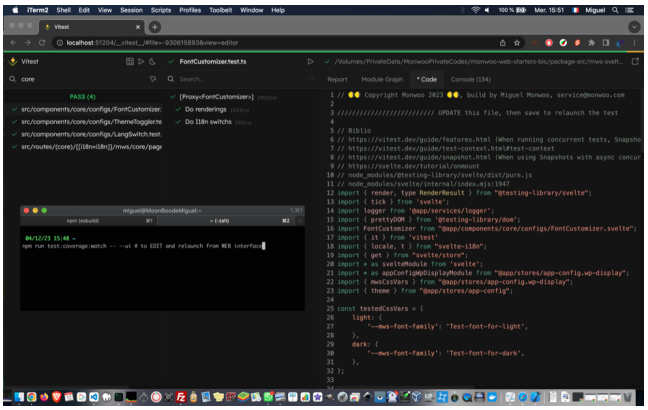




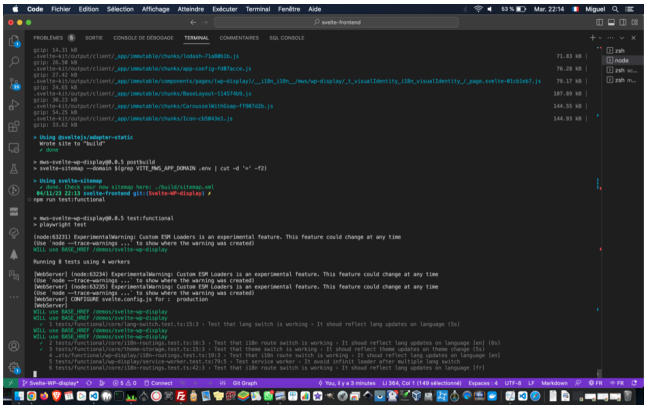
05 - Keep unitest coverage at 100% of all rendered pages (Developper(s) point of view)

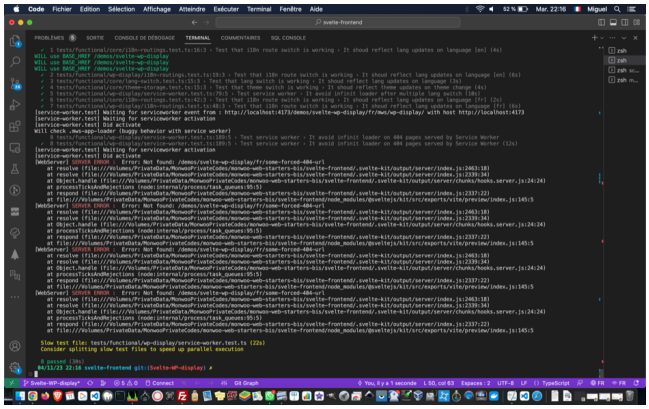


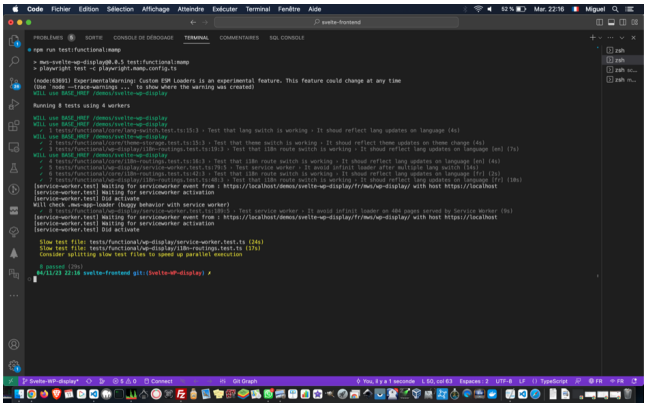




06 - Use functional tests closest to the production target (Dev team point of view)







07 - Use end to end tests for business values (Client point of view)