Вступление

Защищаются студенты группы Б9121-09.03.03пикд Рузин Михаил Александрович, Баздуков Валентин Александрович и Ли Дмитрий Сенунович по теме “Разработка шаблонизатора для методологии FSD”

Feature Sliced Design

Начнем с определения FSD. Методология Feature Sliced Design сосредоточена на разделении кода frontend-приложения на слои, слайсы и сегменты.

- Слои - это первый уровень организационной иерархии в Feature-Sliced Design. Их цель - разделить код на основе того, сколько ответственности ему требуется и от скольких других модулей в приложении он зависит.
- Слайсы - это второй уровень в организационной иерархии Feature-Sliced Design. Их основное назначение - группировать код по его значению для продукта, бизнеса или просто приложения.
- Сегменты - это третий и последний уровень в организационной иерархии, их цель - группировать код по его технической природе.

С помощью композиции нижних слоев в верхних, получается достичь переиспользуемости, хорошей разделяемости кода, и легкости в добавлении нового функционала.

Проблема

Есть проблема. Методология описывает конкретную структуру => из-за этого образуется большое количество шаблонного кода, процесс создания которого можно автоматизировать.

При анализе решений, которые бы закрывали эту задачу, обнаружилось, что пока не существует инструмента, который бы в полной мере соответствовал методологии и свободно распространялся.

Решение

Было решено разработать свое решение. Мы придумали suipta (просто с корейского)
Suipta - это шаблонизатор с консольным интерфейсом, распространяющийся через npm. У него есть 2 функции: генератор слайса и генератор компонента в сегменте.

Слайс создается в выбранном слое и называется на усмотрение пользователя.

Компонент создается в выбранном слое, в выбранном сегменте и называется на усмотрение пользователя. Так же ре-экспорт компонента добавляется в index файл сегмента.

Гибкость

Самая главная фишка suipta в ее гибкости:

- Распространение через npm и cli
- Возможность использовать аргументы для кастомизации поведения генератора
- Возможность написать пользовательские шаблоны
- Конфигурируемость: (Путь до корневой папки проекта, список доступных слоев для генератора слайса и компонента в сегменте и т.д.)

Описание решения

Инструменты разработки:

- Язык - typescript, javascript

- Библиотеки - plop (node-plop), cmd-ts, chalk, ora, handlebars (node-plop) и пр.
- Для сборки использовался esbuild
- Для поддержания чистоты кода использовались eslint и prettie
- Для тестов - vitest
- lefthook для проверки кода перед коммитом
- С помощью github actions был реализован автоматический релиз в npm

За основу suipta взята библиотека node-plop.
С помощью cmd-ts создается консольная команда, которая запускает node-plop с переданными аргументами (bypass). Те аргументы, которые невозможно передать в plop напрямую (не bypass) записываются в файл arguments.yaml. Дале node-plop запускает plopfile.ts под капотом. В plopfile.ts описаны все генераторы suipta. Там же читается конфиг и аргументы и на основании всех данных выполняется генерация.

Формат шаблонов

Формат шаблонов очень прост. Мы пишем обычный текст, который нам необходим. При этом если мы хотим использовать динамические значения из генератора. Мы используем следующих синтаксис.

{{модификатор переменная}}

Переменные для слайсов:
layer, slice
Переменные для компонентов в сегменте:
layer, segment, component

Модификаторы:
camelCase, pascalCase, snakeCase, kebabCase, etc

Тестирование

Для тестирования была использована библиотека vitest.
Были написаны, как юнит тесты, так и e2e тесты (если их можно так назвать).

Заключение

Suipta уже упрощает жизнь мне, как фронтенд-разработчику. Со временем, она обрастет новым функционалом и избавится от лишнего. Время покажет, насколько она будет полезна для других разработчиков.

Мы же довольны, что наш первый вклад в open source начался именно так