# Démystifions la création de blocs Gutenberg sur-mesure

## Comparaison de 3 approches : IA, ACF Blocks et React

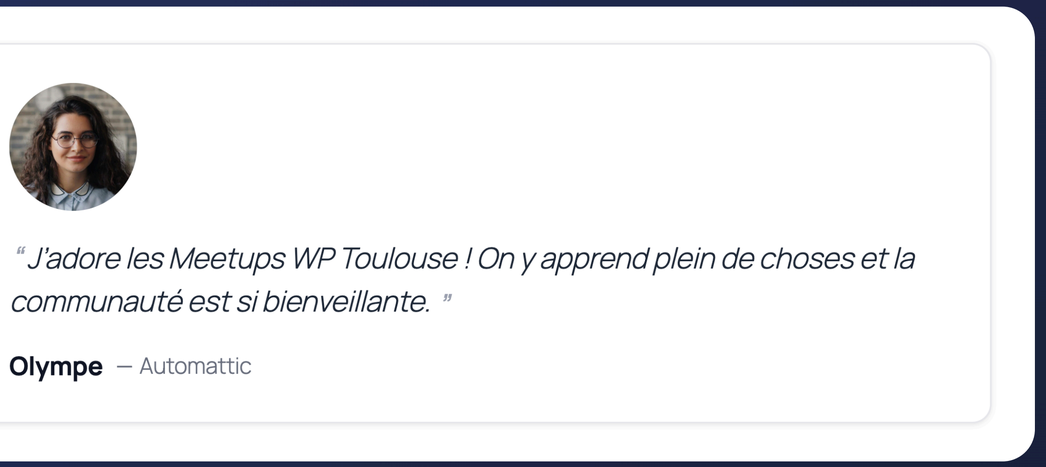
Lundi 11 mai 2026 • Valentin Grenier • Développeur WordPress FSE

Demain, ton client te demande d’ajouter
un bloc “témoignage” dans ses articles.

**Qu’est-ce que tu lui proposes comme solution ?**

Le cahier des charges

# Le bloc qu'on va construire ce soir.




4 zones éditables

- Photo — image
- Citation — WYSIWYG
- Auteur — texte
- Entreprise — texte

Visuel identique entre les 3 versions.

On jugera l'implémentation, pas le design.

Programme

# Et pour construire ce bloc, on va étudier trois manières de faire.

01

## Telex

Génération de bloc grâce à l'IA, par Automattic

02

## ACF Blocks

Conception d'un bloc en PHP avec des champs personnalisés

03

## React

L'approche native pour des blocs Gutenberg aux petits oignons

## Chaque approche est bonne.

Il suffit de choisir celle qui vous convient le mieux.

Lexique

# Mais … c’est quoi un bloc Gutenberg ?

01

## `block.json`

La carte d'identité du bloc : nom, attributs, métadonnées, dépendances.

02

## `edit`

Ce que voit l’éditeur dans Gutenberg. L'expérience côté admin.

03

## `save` ou `render_callback`

Ce qui finit sur le front. C'est là que les 3 approches divergent.

WordPress 5.8+

# block.json, la base du bloc.

```json
{
  "apiVersion": 3,
  "name":       "wp-toulouse/testimonial",
  "title":      "Témoignage client",
  "category":   "design",
  "attributes": { "…" }
}
```

Convention WordPress, pas de duplication PHP / JS. On le retrouve dans les **3 approches**.

Données du bloc

# Les attributs du bloc.

```
{
  "image":    { "type": "object" },
  "citation": { "type": "string" },
  "author":   { "type": "string" },
  "company":  { "type": "string" }
}
```

4 attributs partagés par les **3 versions**. `source: html` permet à RichText de fonctionner.

01

Approche n°1

# Telex.

Génération de bloc par IA.

Génération de bloc par IA

# Telex en 30 secondes.

- Outil créé par **Automattic AI Labs**
- Prompt en langage naturel → **bloc généré dans un plugin**
- Statut : **expérimental**, à tester avant mise en production
- Usage **à la portée de tous**

Approche 01 · Telex

# DÉMO.

telex.automattic.ai

# Un raccourci qui n’en est pas vraiment un.

- Vitesse de prototypage
- IA pour WordPress, par WordPress
- Pas de setup

- Qualité de code variable
- Dépendance à un outil externe
- Minimum de compréhension requise pour maintenir ce qui a été généré

02

Approche n°2

# ACF Blocks.

L’approche PHP-first, render_callback.

# ACF Blocks en 30 secondes.

- Extension **ACF Pro** requise (à partir de 50€ par an)
- Attributs du bloc déclarés comme des groupes de champs classiques
- Enregistrement du bloc via `block.json`
- Un seul fichier pour le rendu éditeur et le rendu front-end.

Approche 02 · ACF Blocks

# DÉMO.

# Le confort de PHP et une dépendance assumée.

- Conception simple quand on connait déjà ACF
- Sidebar ACF auto-générée
- Pas de build, pas de Node
- Un seul fichier pour les deux rendus

- Dépendance à ACF Pro
- Options de personnalisation du bloc limitées par les champs ACF
- Pas d’édition des données depuis la fenêtre d’édition de la page

03

Approche n°3

# React.

L’approche native.

React - JSX natif

# Le triptyque

01

## `block.json`

Schéma + métadonnées,
le contrat du bloc.

02

## `edit`

Composant React, ce que voit
l'éditeur dans Gutenberg.

03

## `save`

Composant React, ce qui finit dans
la base + sur le front.

Deux composants React séparés, qui partagent les attributes mais rendent différemment.

@wordpress/create-block

# Générer un bloc en une ligne de commande.

```
$ npx @wordpress/create-block testimonial-jsx
```

**Plugin PHP**

`register_block_type`

**Sources**

block.json + edit.js + save.js + style.scss

**Tooling**

`wp-scripts` — build, start

Approche 03 · React

# DÉMO.

# Le pari long terme.

- Contrôle total sur l’expérience éditeur
- Aucune dépendance externe donc moins de conflits
- Compatible avec le reste de l’écosystème Gutenberg

- Setup et build à maintenir
- Courbe d’apprentissage de React
- Du code pour tout, même pour le plus simple

# On a vu trois approches.

# Et maintenant ?

# Le comparatif des solutions abordées.

| | Telex | ACF Blocks | React |
|---|---|---|---|
| **Setup** | Aucun | ACF Pro | Node + scaffold |
| **Courbe d'apprentissage** | Faible | Moyenne (PHP+ACF) | Élevée (React) |
| **Contrôle** | Limité | Bon (sauf UX éditeur) | Total |
| **Dépendances** | Telex | ACF Pro | Zéro |
| **Fidélité éditeur** | Variable | Limitée | Native |
| **Maintenabilité** | Variable | Bonne | Excellente si build maintenu |

# Et donc, quelle solution utiliser ?

### Telex

**Prototype rapide.**

Bloc one-shot. Démarrage de POC. Quand le code généré est jetable.

### ACF Blocks

**Équipe à dominante PHP**

Contenu éditorial classique, en complément d'un autre builder.

### React

**Meilleure intégration**

Conception FSE, contrôle du code, meilleure UX au sein de l'éditeur.

Deux composants React séparés, qui partagent les attributes mais rendent différemment.

# Pour aller plus loin.

- **Dépôt Github des 3 implémentations**
  `github.com/valentin-grenier/wptoulouse-meetup-blocks`
- **Doc block.json**
  `developer.wordpress.org/block-editor/`
- **ACF Blocks**
  `advancedcustomfields.com/resources/blocks/`
- **@wordpress/create-block**
  `developer.wordpress.org/block-editor/reference-guides/packages/packages-create-block/`

# Pas de gagnant universel.

Le bon choix dépend du contexte projet et de l'équipe,
pas d'une supériorité technique.

Télécharger les slides

# Merci pour votre écoute !

Si vous avez des questions, c’est le moment.

**Valentin Grenier** • studio-val.fr • Développeur WordPress FSE