План разработки: CSV Export Plugin

Дата создания: 17 декабря 2025
Статус: Планирование
Оценка времени: 7-9 часов

Обзор

Плагин для экспорта произвольных типов постов (Collection, Photo) в CSV формат через стандартный механизм WordPress Bulk Actions. Интегрируется непосредственно в админ-панель списков постов, использует существующие фильтры Polylang и WordPress.

Требования

Функциональные

1. **Интеграция в списки постов CPT**
   - Добавить действие "Экспорт в CSV" в Bulk Actions
   - Работа для типов постов: `collection`, `photo`
   - Работает со стандартным функционалом "Select All" (без дополнительной логики)

2. **Фильтрация**
   - Использовать стандартный переключатель языков Polylang
   - Использовать стандартные фильтры WordPress (таксономии, даты, поиск)
   - Экспортировать только `post_status = publish`

3. **Выводимые данные**

   Стандартные поля:
   - ID поста
   - Название (title)
   - Slug
   - Статус публикации
   - Дата создания
   - Язык (Polylang)

   Таксономии:
   - Выводить человекочитаемые названия (name, не slug)
   - Несколько терминов через запятую
   - Для Collection: жанры, техники, материалы, стили, статус
   - Для Photo: теги и другие таксономии

   ACF поля:

- ✅ **Включить:** text, textarea, number, date_picker, select, radio, checkbox, url, email, wysiwyg
- ✅ **Image/File:** выводить URL изображения
- ✅ **Post Object:** выводить название поста
- ✅ **Relationship:** выводить названия постов через `;`
- ✅ **Taxonomy (ACF):** выводить названия терминов
- ❌ **Исключить:** gallery, repeater, flexible_content, clone, group, tab, message, accordion

Специальная обработка Artist (Post Object):

- Для полей типа `post_object` со связью на `artist` CPT
- Формат вывода: `Post Title (Name Patronimic)`
- Пример: `Малевич (Казимир Северинович)`
- Применяется для полей: `depicted_artists` (Photo), `artist_id` (Collection)

4. **Формат CSV**

- Кодировка: UTF-8 with BOM (для корректного открытия в Excel)
- Разделитель: `;` (точка с запятой)
- Заголовки: человекочитаемые названия полей (ACF labels)
- Имя файла: `{post_type}_{lang}_{date}.csv`
  - Пример: `collection_ru_2025-12-17.csv`
  - Если "Все языки": `collection_all_2025-12-17.csv`

5. **Polylang интеграция**

- Автоматическое определение языка из URL параметра `?lang=ru`
- Если выбран конкретный язык → экспортируются только посты этого языка
- Если выбраны "Все языки" (`?lang=all` или пустой параметр) → экспортируются все посты всех языков

Нефункциональные

- Плагин должен быть легковесным (без зависимостей)
- Код должен быть расширяемым (легко добавить новые CPT)
- Использовать WordPress Coding Standards
- Минимальная версия WordPress: 5.8+
- Минимальная версия PHP: 7.4+

Архитектура

Структура файлов

```
www/wordpress/wp-content/plugins/maslovka-csv-exporter/
├── maslovka-csv-exporter.php           # Главный файл плагина
├── includes/
│   ├── class-bulk-export-handler.php   # Регистрация и обработка bulk
actions
│   ├── class-csv-generator.php         # Генерация CSV файла
│   ├── class-field-handler.php         # Обработка ACF полей
```

```
│   └── class-post-query.php            # Получение постов с учетом
фильтров
├── assets/
│   └── admin.css                       # Стили (если потребуются)
└── README.md                           # Документация
```

Классы и их ответственность

1. `Maslovka_CSV_Exporter` (главный класс)

```
// maslovka-csv-exporter.php

/**
 * Инициализация плагина
 * Подключение файлов
 * Регистрация хуков активации/деактивации
 */
```

2. `Maslovka_Bulk_Export_Handler`

```
/**
 * Регистрация bulk actions для CPT
 * Обработка действия "Экспорт в CSV"
 * Получение выбранных постов
 * Запуск генерации CSV
 */

// Основные методы:
public function register_bulk_actions()
public function handle_bulk_action($redirect, $action, $post_ids)
private function get_post_type_from_screen()
private function get_current_language()
```

3. `Maslovka_CSV_Generator`

```
/**
 * Генерация CSV файла
 * Формирование заголовков
 * Обработка строк данных
 * Вывод файла для скачивания
 */

// Основные методы:
public function generate($post_ids, $post_type, $lang)
private function get_headers($post_type)
```

```
private function get_row_data($post_id, $post_type)
private function output_csv($data, $filename)
```

4. `Maslovka_Field_Handler`

```
/**
 * Получение всех полей для CPT
 * Обработка каждого типа поля
 * Форматирование значений
 * Специальная логика для Artist
 */

// Основные методы:
public function get_all_fields($post_type)
public function get_field_value($field, $post_id)
private function handle_text_field($value)
private function handle_image_field($value)
private function handle_post_object_field($value, $field_config)
private function handle_relationship_field($value, $field_config)
private function handle_taxonomy_field($value, $field_config)
private function format_artist_with_details($artist_id)
```

5. `Maslovka_Post_Query`

```
/**
 * Получение постов с учетом фильтров
 * (Опционально, если понадобится обработка "Select All")
 */

// Основные методы:
public function get_filtered_posts($post_type, $query_args)
```

Техническая реализация

1. Регистрация Bulk Actions

```
// includes/class-bulk-export-handler.php

class Maslovka_Bulk_Export_Handler {

    public function __construct() {
        // Регистрируем для Collection
        add_filter('bulk_actions-edit-collection', [$this,
'register_bulk_actions']);
        add_filter('handle_bulk_actions-edit-collection', [$this,
```

```
'handle_bulk_action'], 10, 3);

        // Регистрируем для Photo
        add_filter('bulk_actions-edit-photo', [$this,
'register_bulk_actions']);
        add_filter('handle_bulk_actions-edit-photo', [$this,
'handle_bulk_action'], 10, 3);
    }

    public function register_bulk_actions($actions) {
        $actions['maslovka_export_csv'] = __('Экспорт в CSV', 'maslovka-
csv-exporter');
        return $actions;
    }

    public function handle_bulk_action($redirect, $action, $post_ids) {
        if ($action !== 'maslovka_export_csv') {
            return $redirect;
        }

        // Получаем тип поста из текущего экрана
        $post_type = $this->get_post_type_from_screen();

        // Получаем текущий язык из Polylang фильтра
        $lang = $this->get_current_language();

        // Генерируем CSV
        $generator = new Maslovka_CSV_Generator();
        $generator->generate($post_ids, $post_type, $lang);

        exit; // Прерываем выполнение после вывода файла
    }

    private function get_post_type_from_screen() {
        $screen = get_current_screen();
        return $screen->post_type;
    }

    private function get_current_language() {
        // Polylang передает язык через GET параметр
        if (isset($_GET['lang'])) {
            return $_GET['lang'] === '' ? 'all' :
sanitize_text_field($_GET['lang']);
        }
        return 'all';
    }
}
```

Важно: WordPress стандартно передает массив `$post_ids` с выбранными постами. Логика "Select All" работает автоматически — WP сам собирает все ID с учетом фильтров. Дополнительная обработка не требуется.

---

2. Генерация CSV

```php
// includes/class-csv-generator.php

class Maslovka_CSV_Generator {

    private $field_handler;

    public function __construct() {
        $this->field_handler = new Maslovka_Field_Handler();
    }

    public function generate($post_ids, $post_type, $lang) {
        if (empty($post_ids)) {
            wp_die(__('Не выбраны посты для экспорта', 'maslovka-csv-exporter'));
        }

        // Получаем заголовки
        $headers = $this->get_headers($post_type);

        // Собираем данные
        $data = [];
        $data[] = $headers; // Первая строка - заголовки

        foreach ($post_ids as $post_id) {
            $data[] = $this->get_row_data($post_id, $post_type);
        }

        // Генерируем имя файла
        $filename = sprintf(
            '%s_%s_%s.csv',
            $post_type,
            $lang,
            date('Y-m-d')
        );

        // Выводим CSV
        $this->output_csv($data, $filename);
    }

    private function get_headers($post_type) {
        $headers = [
            'ID',
            'Название',
            'Slug',
            'Статус',
            'Дата создания',
            'Язык'
        ];

        // Добавляем таксономии
```

```
        $taxonomies = get_object_taxonomies($post_type, 'objects');
        foreach ($taxonomies as $taxonomy) {
            if (!$taxonomy->public || $taxonomy->name === 'language' ||
$taxonomy->name === 'post_translations') {
                continue;
            }
            $headers[] = $taxonomy->label;
        }

        // Добавляем ACF поля
        $acf_fields = $this->field_handler->get_all_fields($post_type);
        foreach ($acf_fields as $field) {
            $headers[] = $field['label'];
        }

        return $headers;
    }

    private function get_row_data($post_id, $post_type) {
        $post = get_post($post_id);

        $row = [
            $post->ID,
            $post->post_title,
            $post->post_name,
            $post->post_status,
            get_the_date('Y-m-d H:i:s', $post_id),
            function_exists('pll_get_post_language') ?
pll_get_post_language($post_id) : '-'
        ];

        // Добавляем таксономии
        $taxonomies = get_object_taxonomies($post_type, 'objects');
        foreach ($taxonomies as $taxonomy) {
            if (!$taxonomy->public || $taxonomy->name === 'language' ||
$taxonomy->name === 'post_translations') {
                continue;
            }

            $terms = get_the_terms($post_id, $taxonomy->name);
            if ($terms && !is_wp_error($terms)) {
                $term_names = wp_list_pluck($terms, 'name');
                $row[] = implode(', ', $term_names);
            } else {
                $row[] = '';
            }
        }

        // Добавляем ACF поля
        $acf_fields = $this->field_handler->get_all_fields($post_type);
        foreach ($acf_fields as $field) {
            $value = $this->field_handler->get_field_value($field,
$post_id);
            $row[] = $value;
```

```
        }

        return $row;
    }

    private function output_csv($data, $filename) {
        // Устанавливаем заголовки для скачивания
        header('Content-Type: text/csv; charset=utf-8');
        header('Content-Disposition: attachment; filename="' . $filename .
'"');
        header('Pragma: no-cache');
        header('Expires: 0');

        // Открываем поток вывода
        $output = fopen('php://output', 'w');

        // Добавляем UTF-8 BOM для Excel
        fprintf($output, chr(0xEF).chr(0xBB).chr(0xBF));

        // Записываем данные
        foreach ($data as $row) {
            // Используем точку с запятой как разделитель
            fputcsv($output, $row, ';');
        }

        fclose($output);
    }
}
```

---

3. Обработка ACF полей

```
// includes/class-field-handler.php

class Maslovka_Field_Handler {

    private $excluded_types = [
        'gallery',
        'repeater',
        'flexible_content',
        'clone',
        'group',
        'tab',
        'message',
        'accordion'
    ];

    public function get_all_fields($post_type) {
        if (!function_exists('acf_get_field_groups')) {
            return [];
        }
```

```
        $fields = [];

        // Получаем все группы полей для данного post_type
        $field_groups = acf_get_field_groups([
            'post_type' => $post_type
        ]);

        foreach ($field_groups as $group) {
            $group_fields = acf_get_fields($group['key']);

            if ($group_fields) {
                foreach ($group_fields as $field) {
                    // Исключаем сложные типы
                    if (!in_array($field['type'], $this->excluded_types))
{
                        $fields[] = $field;
                    }
                }
            }
        }

        return $fields;
    }

    public function get_field_value($field, $post_id) {
        $value = get_field($field['name'], $post_id);

        if (empty($value)) {
            return '';
        }

        // Обработка по типу поля
        switch ($field['type']) {
            case 'image':
            case 'file':
                return $this->handle_image_field($value);

            case 'post_object':
                return $this->handle_post_object_field($value, $field);

            case 'relationship':
                return $this->handle_relationship_field($value, $field);

            case 'taxonomy':
                return $this->handle_taxonomy_field($value, $field);

            case 'select':
            case 'radio':
            case 'checkbox':
                return $this->handle_choice_field($value, $field);

            case 'true_false':
                return $value ? 'Да' : 'Нет';
```

```
            case 'wysiwyg':
                return strip_tags($value);

            default:
                return $this->handle_text_field($value);
        }
    }

    private function handle_text_field($value) {
        if (is_array($value)) {
            return implode(', ', $value);
        }
        return (string) $value;
    }

    private function handle_image_field($value) {
        if (is_array($value) && isset($value['url'])) {
            return $value['url'];
        }
        if (is_numeric($value)) {
            $url = wp_get_attachment_url($value);
            return $url ? $url : '';
        }
        return '';
    }

    private function handle_post_object_field($value, $field_config) {
        if (empty($value)) {
            return '';
        }

        // Если это массив постов
        if (is_array($value)) {
            $titles = [];
            foreach ($value as $post_obj) {
                $post_id = is_object($post_obj) ? $post_obj->ID :
$post_obj;
                $titles[] = $this->get_post_title_with_details($post_id,
$field_config);
            }
            return implode('; ', $titles);
        }

        // Если это один пост
        $post_id = is_object($value) ? $value->ID : $value;
        return $this->get_post_title_with_details($post_id,
$field_config);
    }

    private function handle_relationship_field($value, $field_config) {
        if (empty($value)) {
            return '';
        }
```

```
        $titles = [];
        foreach ($value as $post_id) {
            $titles[] = $this->get_post_title_with_details($post_id,
$field_config);
        }

        return implode('; ', $titles);
    }

    private function get_post_title_with_details($post_id, $field_config)
{
        $post = get_post($post_id);
        if (!$post) {
            return '';
        }

        $title = $post->post_title;

        // Специальная обработка для Artist
        $post_types = isset($field_config['post_type']) ?
(array)$field_config['post_type'] : [];

        if (in_array('artist', $post_types)) {
            $name = get_field('name', $post_id);
            $patronimic = get_field('patronimic', $post_id);

            if ($name || $patronimic) {
                $details = trim($name . ' ' . $patronimic);
                return sprintf('%s (%s)', $title, $details);
            }
        }

        return $title;
    }

    private function handle_taxonomy_field($value, $field_config) {
        if (empty($value)) {
            return '';
        }

        if (!is_array($value)) {
            $value = [$value];
        }

        $term_names = [];
        foreach ($value as $term_id) {
            $term = get_term($term_id);
            if ($term && !is_wp_error($term)) {
                $term_names[] = $term->name;
            }
        }

        return implode(', ', $term_names);
```

```
    }

    private function handle_choice_field($value, $field_config) {
        if (empty($value)) {
            return '';
        }

        // Если есть массив choices, получаем label
        if (isset($field_config['choices']) &&
is_array($field_config['choices'])) {
            if (is_array($value)) {
                $labels = [];
                foreach ($value as $key) {
                    $labels[] = isset($field_config['choices'][$key])
                        ? $field_config['choices'][$key]
                        : $key;
                }
                return implode(', ', $labels);
            } else {
                return isset($field_config['choices'][$value])
                    ? $field_config['choices'][$value]
                    : $value;
            }
        }

        return is_array($value) ? implode(', ', $value) : $value;
    }
}
```

Этапы разработки

Фаза 1: Инфраструктура и базовая интеграция (2 часа)

Задачи:

1. Создать структуру плагина
2. Создать главный файл с метаданными
3. Создать базовые классы (пустые заглушки)
4. Зарегистрировать bulk actions для Collection и Photo
5. Базовая обработка: вывести `var_dump($post_ids)` и `die()`

Результат: Действие "Экспорт в CSV" появляется в списке, при клике показывает выбранные ID

Тестирование:

- ✅ Bulk action появляется в выпадающем списке
- ✅ При выборе постов и клике на "Применить" срабатывает наш обработчик
- ✅ Выводятся ID выбранных постов

Фаза 2: Генерация CSV со стандартными полями (2 часа)

Задачи:

1. Реализовать `Maslovka_CSV_Generator::generate()`
2. Реализовать `get_headers()` для стандартных полей
3. Реализовать `get_row_data()` для стандартных полей
4. Реализовать `output_csv()` с UTF-8 BOM и разделителем `;`
5. Добавить вывод таксономий (человеческие названия)

Результат: Скачивается CSV с ID, Title, Slug, Date, Language и таксономиями

Тестирование:

- ✅ CSV файл скачивается корректно
- ✅ Имя файла: `collection_ru_2025-12-17.csv`
- ✅ Открывается в Excel без проблем с кириллицей
- ✅ Таксономии выводятся корректно (названия, не slug)

Фаза 3: Обработка простых ACF полей (1.5 часа)

Задачи:

1. Реализовать `get_all_fields()` — получение всех ACF полей для CPT
2. Исключить сложные типы (gallery, repeater, etc)
3. Реализовать обработку простых полей:
   - text, textarea, number, date_picker
   - select, radio, checkbox (с выводом label, не value)
   - url, email
   - true_false
4. Добавить ACF поля в заголовки и строки CSV

Результат: CSV включает все простые ACF поля

Тестирование:

- ✅ Все простые поля выводятся
- ✅ Select/Radio показывают label (не value)
- ✅ Gallery/Repeater не попадают в CSV

Фаза 4: Обработка сложных ACF полей (2 часа)

Задачи:

1. Реализовать `handle_image_field()` — вывод URL
2. Реализовать `handle_post_object_field()` — вывод названий
3. Реализовать `handle_relationship_field()` — вывод списка через `;`
4. Реализовать `handle_taxonomy_field()` — вывод названий терминов
5. **Специальная обработка Artist:**
   - Определять поля со связью на `artist` CPT
   - Получать `name` и `patronimic` из связанного поста

- Формат: `Post Title (Name Patronimic)`
- Применять для `depicted_artists`, `artist_id`

Результат: Все типы полей обрабатываются корректно, Artist выводится с деталями

Тестирование:

- ✅ Image поля выводят URL
- ✅ Post Object выводит названия постов
- ✅ Relationship выводит список через `;`
- ✅ Artist выводится как `Малевич (Казимир Северинович)`

Фаза 5: Интеграция Polylang (0.5 часа)

Задачи:

1. Получать текущий язык из `$_GET['lang']`
2. Добавлять язык в имя файла
3. Тестировать с разными языками

Результат: Язык правильно определяется и попадает в имя файла

Тестирование:

- ✅ При выборе RU: файл `collection_ru_2025-12-17.csv`
- ✅ При выборе EN: файл `collection_en_2025-12-17.csv`
- ✅ При "Все языки": файл `collection_all_2025-12-17.csv`
- ✅ Экспортируются только посты выбранного языка (проверка WP Query)

Фаза 6: Тестирование и отладка (1 час)

Тестовые сценарии:

1. **Collection CPT:**
   - Выбрать 5 постов → экспортировать
   - Выбрать язык RU → "Select All" → экспортировать
   - Применить фильтр по таксономии "Живопись" → экспортировать
   - Проверить все ACF поля (artist_id, year_created, dimensions, etc)
2. **Photo CPT:**
   - Выбрать 10 фото → экспортировать
   - Проверить поле `depicted_artists` (должен быть формат с Name и Patronimic)
   - Проверить `date_taken`, `photographer`, `place_taken`
3. **Polylang:**
   - Выбрать RU → экспортировать → проверить, что только RU посты
   - Выбрать EN → экспортировать → проверить, что только EN посты

    - Выбрать "Все языки" → экспортировать → проверить, что все посты

4. **Excel:**
    - Открыть CSV в Excel → проверить кириллицу
    - Проверить корректность разделителей
    - Проверить, что нет "разъезжания" данных по столбцам

Баги для отлова:

- Проблемы с кодировкой (BOM)
- Неправильные разделители
- Пустые значения в полях
- Ошибки при отсутствующих ACF полях
- Проблемы с экранированием спецсимволов в CSV

Возможные расширения (будущее)

Опциональный функционал (не включать в MVP)

1. **История экспортов**
    - Логирование каждого экспорта (кто, когда, что)
    - Страница с историей в админке
2. **Настройки полей**
    - UI для выбора, какие поля включать в экспорт
    - Сохранение "пресетов" экспорта
3. **Расширенные форматы**
    - Excel (.xlsx)
    - JSON
    - XML
4. **Планировщик экспортов**
    - Автоматические экспорты по расписанию
    - Отправка на email / FTP
5. **Импорт CSV**
    - Обратная операция (импорт данных обратно в посты)

Чеклист перед релизом

Код

- [ ] Все классы и методы документированы (PHPDoc)
- [ ] Код соответствует WordPress Coding Standards

- [ ] Нет hardcoded строк (все через `__()` для переводов)
- [ ] Используется `sanitize_text_field()`, `esc_html()` и т.д.
- [ ] Нет прямых SQL запросов (только WP Query API)

Тестирование

- [ ] Экспорт Collection (5+ записей, разные языки)
- [ ] Экспорт Photo (5+ записей, с depicted_artists)
- [ ] "Select All" с фильтрами
- [ ] Проверка в Excel (кириллица, разделители)
- [ ] Проверка с отключенным Polylang
- [ ] Проверка с отключенным ACF

Документация

- [ ] README.md с инструкцией по установке
- [ ] Комментарии в коде
- [ ] Описание фильтров и хуков (если добавляются)

Безопасность

- [ ] Проверка прав пользователя (`current_user_can('edit_posts')`)
- [ ] Nonce проверки (если добавляется AJAX)
- [ ] Санитизация всех входных данных

Финальная оценка времени

Фаза	Описание	Время
1	Инфраструктура и базовая интеграция	2 ч
2	Генерация CSV со стандартными полями	2 ч
3	Обработка простых ACF полей	1.5 ч
4	Обработка сложных ACF полей	2 ч
5	Интеграция Polylang	0.5 ч
6	Тестирование и отладка	1 ч
Итого		**9 часов**

Буфер на непредвиденные сложности: +1-2 часа

Общая оценка: 9-11 часов

Следующие шаги

1. ✅ Согласовать план с клиентом
2. ⏳ Начать разработку (Фаза 1)

3. ⏳ После Фазы 3 — промежуточное тестирование
4. ⏳ После Фазы 6 — финальное тестирование
5. ⏳ Деплой на DEV окружение
6. ⏳ Тестирование клиентом
7. ⏳ Деплой на PROD

Вопросы? Готов начать разработку после подтверждения плана.