

???????????? ???? ??????
????? ? ??

????? - ?????????? ?????????? ????? ? ??

<https://drive.google.com/file/d/1H-59es2TBHsLfQOFSJ19qLfKQqq9kamo/view?usp=sharing>

image.png

Пример таблицы с политикой тегов

<https://docs.google.com/spreadsheets/d/17Aenkk8mBiiXaTggyUDVpMad6FQ6uu9Njqp0e00mLL0/edit?gid=375214728#gid=375214728>

JS для вставки в ГК

```
/**
 * GetCourse Offer Tag Manager
 * Скрипт для управления обязательными тегами офферов
 *
 * Формат тега: TYPE:PRODUCT:CAT (например: SPR:МК:P)
 */

(function() {
    'use strict';

    // =====
    // КОНФИГУРАЦИЯ
    // =====
    const Config = {
        // Google Sheets
        spreadsheetId: '<spreadsheetId>...j02z2jEP7Sp',
        sheets: ['TYPE', 'PRODUCT', 'CAT'],

        // Google Apps Script Web App URL для добавления значений
        appsScriptUrl: 'https://script.google.com/macros/s/<appsScriptUrl>/exec',
```

```
// Формат обязательного тега
tagFormat: {
    separator: ':',
    // Базовый паттерн: 3 части через двоеточие, заглавные буквы/цифры
    pattern: /^[A-Z0-9]+:[A-Z0-9]+:[A-Z]$/,
    // Строгий паттерн с известными значениями (будет обновлен после загрузки
данных)
    strictPattern: null
},

// Селекторы GetCourse
selectors: {
    offerForm: '#offerForm',
    headerWithTags: '.header-with-tags',
    offerUpdate: '.offer-update',
    saveButton: '.save-offer',
    copyButton: '.copy-offer',
    // Селекторы для чтения тегов
    tagsContainer: '.gc-tags-editable',
    tagsInputName: 'Offer[tags][]'
},

// UI настройки
ui: {
    containerId: 'mandatory-tag-container',
    selectClass: 'mandatory-tag-select'
},

// Сообщения
messages: {
    selectPlaceholder: '-- Выберите --',
    missingParams: 'Выберите все параметры обязательного тега',
    validTag: 'Тег сформирован',
    copyHint: 'Скопируйте и вставьте тег в список тегов оффера',
    missingTag: 'Обязательный тег отсутствует в списке тегов оффера',
    invalidTag: 'Обязательный тег имеет неверный формат',
    multipleTags: 'Найдено несколько обязательных тегов. Должен быть ровно один.',
    tagFound: 'Обязательный тег найден в списке',
```

```

        labels: {
            type: 'Тип',
            product: 'Продукт',
            cat: 'Категория',
            preview: 'Тег',
            title: 'Обязательный тег оффера'
        },
        addValue: {
            title: 'Добавить новое значение',
            code: 'Код (заглавные буквы/цифры)',
            label: 'Название',
            description: 'Описание (необязательно)',
            submit: 'Добавить',
            cancel: 'Отмена',
            success: 'Значение успешно добавлено',
            error: 'Ошибка при добавлении значения',
            invalidCode: 'Код должен содержать только заглавные буквы и цифры',
            duplicateCode: 'Такой код уже существует'
        }
    }
};

// =====
// ЗАГРУЗЧИК ДАННЫХ
// =====
const DataLoader = {
    cache: {},

    /**
     * URL для Google Visualization Query API
     */
    buildUrl(sheetName) {
        return
`https://docs.google.com/spreadsheets/d/${Config.spreadsheetId}/gviz/tq?tx=out:json&sheet=${encodeURIComponent(sheetName)}`;
    },

    /**
     * Парсинг ответа Google Visualization API

```

```

*/
parseGvizResponse(responseText) {
  try {
    // Удаляем JSONP обертку: google.visualization.Query.setResponse({...});
    const jsonStr = responseText
      .replace(/^[^()]*\(/, '')
      .replace(/\);?\s*$/, '');

    const data = JSON.parse(jsonStr);
    const rows = data.table.rows || [];

    // Пропускаем первую строку (заголовки)
    const dataRows = rows.slice(1);

    return dataRows.map(row => {
      const cells = row.c || [];
      return {
        code: cells[0]?.v?.toString() || '',
        label: cells[1]?.v?.toString() || cells[0]?.v?.toString() || '',
        description: cells[2]?.v?.toString() || ''
      };
    }).filter(item => item.code && item.code.trim());
  } catch (e) {
    console.error('[OfferTagManager] Parse error:', e);
    return [];
  }
},

/**
 * Загрузить один лист
 */
async loadSheet(sheetName) {
  if (this.cache[sheetName]) {
    return this.cache[sheetName];
  }

  const url = this.buildUrl(sheetName);

  const response = await fetch(url);

```

```

    if (!response.ok) {
        throw new Error(`HTTP ${response.status} for sheet ${sheetName}`);
    }

    const text = await response.text();
    const data = this.parseGvizResponse(text);

    if (data.length === 0) {
        throw new Error(`Empty data for sheet ${sheetName}`);
    }

    this.cache[sheetName] = data;
    return data;
},

/**
 * Загрузить все листы параллельно
 */
async loadAllSheets() {
    const results = await Promise.all(
        Config.sheets.map(sheet =>
            this.loadSheet(sheet)
                .then(data => ({ sheet, data, error: null }))
                .catch(error => ({ sheet, data: [], error })))
    );

    const sheetsData = {};
    const errors = [];

    for (const result of results) {
        if (result.error) {
            errors.push(`${result.sheet}: ${result.error.message}`);
        } else {
            sheetsData[result.sheet] = result.data;
        }
    }

    if (errors.length > 0) {

```

```

        throw new Error(errors.join('; '));
    }

    return sheetsData;
}
};

// =====
// ПАРСЕР ТЕГОВ
// =====
const TagParser = {
    knownValues: {
        TYPE: [],
        PRODUCT: [],
        CAT: []
    },

    /**
     * Установить известные значения из загруженных данных
     */
    setKnownValues(sheetsData) {
        this.knownValues.TYPE = sheetsData.TYPE.map(item => item.code);
        this.knownValues.PRODUCT = sheetsData.PRODUCT.map(item => item.code);
        this.knownValues.CAT = sheetsData.CAT.map(item => item.code);

        // Построить строгий regex
        const typePattern = this.knownValues.TYPE.join('|');
        const productPattern = this.knownValues.PRODUCT.join('|');
        const catPattern = this.knownValues.CAT.join('|');

        Config.tagFormat.strictPattern = new RegExp(
            `^(${typePattern}):(${productPattern}):(${catPattern})$`
        );
    },

    /**
     * Проверить, похож ли тег на обязательный (по формату)
     */
    isMandatoryTagFormat(tag) {

```

```

        return Config.tagFormat.pattern.test(tag);
    },

    /**
     * Строгая проверка тега (известные значения)
     */
    isValidMandatoryTag(tag) {
        if (!Config.tagFormat.strictPattern) {
            return this.isMandatoryTagFormat(tag);
        }
        return Config.tagFormat.strictPattern.test(tag);
    },

    /**
     * Разобрать тег на компоненты
     */
    parseTag(tag) {
        if (!this.isMandatoryTagFormat(tag)) {
            return null;
        }

        const parts = tag.split(Config.tagFormat.separator);
        return {
            type: parts[0],
            product: parts[1],
            cat: parts[2]
        };
    },

    /**
     * Собрать тег из компонентов
     */
    buildTag(type, product, cat) {
        if (!type || !product || !cat) return null;
        return [type, product, cat].join(Config.tagFormat.separator);
    },

    /**
     * Найти обязательные теги в массиве

```

```

    */
    findMandatoryTags(tags) {
        const mandatory = tags.filter(tag => {
            const matches = this.isMandatoryTagFormat(tag);
            console.log(`[OfferTagManager] Tag "${tag}" matches pattern: ${matches}`);
            return matches;
        });
        return {
            tags: mandatory,
            count: mandatory.length,
            first: mandatory[0] || null
        };
    }
};

// =====
// SEARCHABLE SELECT КОМПОНЕНТ
// =====
const SearchableSelect = {
    /**
     * Создать searchable dropdown вместо обычного select
     */
    create(id, options, placeholder) {
        const $wrapper = $('<div>')
            .addClass('searchable-select')
            .attr('data-id', id);

        const $input = $('<input>')
            .attr('type', 'text')
            .attr('id', id)
            .attr('placeholder', placeholder)
            .attr('autocomplete', 'off')
            .addClass('form-control searchable-input')
            .css('fontFamily', 'monospace');

        const $hidden = $('<input>')
            .attr('type', 'hidden')
            .addClass('searchable-value');
    }
};

```

```

const $dropdown = $('<div>')
    .addClass('searchable-dropdown');

// Создать опции
for (const item of options) {
    const $option = $('<div>')
        .addClass('searchable-option')
        .attr('data-value', item.code)
        .attr('data-code', item.code.toLowerCase())
        .attr('data-label', item.label.toLowerCase())
        .attr('title', item.description || '')
        .text(`${item.code} – ${item.label}`);
    $dropdown.append($option);
}

$wrapper.append($input).append($hidden).append($dropdown);

// Привязка событий
this._bindEvents($wrapper);

return $wrapper;
},

/**
 * Привязка событий к компоненту
 */
_bindEvents($wrapper) {
    const $input = $wrapper.find('.searchable-input');
    const $dropdown = $wrapper.find('.searchable-dropdown');
    const $options = $dropdown.find('.searchable-option');
    const self = this;
    let activeIndex = -1;

    // Фокус – показать dropdown
    $input.on('focus', function() {
        self._showAllOptions($wrapper);
        $dropdown.show();
        activeIndex = -1;
    });

```

```

// Ввод текста – фильтрация
$input.on('input', function() {
    const query = $(this).val().toLowerCase().trim();
    self._filterOptions($wrapper, query);
    activeIndex = -1;
    self._updateActiveOption($wrapper, activeIndex);
});

// Клавиатурная навигация
$input.on('keydown', function(e) {
    const $visible = $dropdown.find('.searchable-option:not(.hidden)');
    const visibleCount = $visible.length;

    if (e.key === 'ArrowDown') {
        e.preventDefault();
        if (visibleCount > 0) {
            activeIndex = (activeIndex + 1) % visibleCount;
            self._updateActiveOption($wrapper, activeIndex);
        }
    } else if (e.key === 'ArrowUp') {
        e.preventDefault();
        if (visibleCount > 0) {
            activeIndex = activeIndex <= 0 ? visibleCount - 1 : activeIndex -
1;

            self._updateActiveOption($wrapper, activeIndex);
        }
    } else if (e.key === 'Enter') {
        e.preventDefault();
        const $active = $dropdown.find('.searchable-option.active');
        if ($active.length) {
            self._selectOption($wrapper, $active);
        }
    } else if (e.key === 'Escape') {
        $dropdown.hide();
        $input.blur();
    }
});

```

```

// Клик на опцию
$dropdown.on('click', '.searchable-option', function() {
    self._selectOption($wrapper, $(this));
});

// Hover на опцию
$dropdown.on('mouseenter', '.searchable-option', function() {
    $options.removeClass('active');
    $(this).addClass('active');
    activeIndex = $dropdown.find('.searchable-
option:not(.hidden)').index($(this));
});

// Клик вне — закрыть dropdown
$(document).on('click', function(e) {
    if (!$wrapper.is(e.target) && $wrapper.has(e.target).length === 0) {
        $dropdown.hide();
    }
});

},

/**
 * Показать все опции (сбросить фильтр)
 */
_showAllOptions($wrapper) {
    $wrapper.find('.searchable-option').removeClass('hidden active');
},

/**
 * Фильтрация опций по запросу
 */
_filterOptions($wrapper, query) {
    const $options = $wrapper.find('.searchable-option');

    if (!query) {
        $options.removeClass('hidden');
        return;
    }
}

```

```

$options.each(function() {
    const $opt = $(this);
    const code = $opt.attr('data-code');
    const label = $opt.attr('data-label');

    // Поиск по коду И по label
    if (code.includes(query) || label.includes(query)) {
        $opt.removeClass('hidden');
    } else {
        $opt.addClass('hidden');
    }
});
},

/**
 * Обновить активную опцию (подсветка)
 */
_updateActiveOption($wrapper, index) {
    const $visible = $wrapper.find('.searchable-option:not(.hidden)');
    $visible.removeClass('active');

    if (index >= 0 && index < $visible.length) {
        const $active = $visible.eq(index);
        $active.addClass('active');

        // Прокрутка к активной опции
        const $dropdown = $wrapper.find('.searchable-dropdown');
        const optionTop = $active.position().top;
        const optionHeight = $active.outerHeight();
        const dropdownHeight = $dropdown.height();
        const scrollTop = $dropdown.scrollTop();

        if (optionTop < 0) {
            $dropdown.scrollTop(scrollTop + optionTop);
        } else if (optionTop + optionHeight > dropdownHeight) {
            $dropdown.scrollTop(scrollTop + optionTop + optionHeight -
dropdownHeight);
        }
    }
}
}

```

```

},

/**
 * Выбрать опцию
 */
_selectOption($wrapper, $option) {
    const value = $option.attr('data-value');
    const text = $option.text();

    $wrapper.find('.searchable-input').val(text);
    $wrapper.find('.searchable-value').val(value);
    $wrapper.find('.searchable-dropdown').hide();

    // Trigger change event для обновления превью
    $wrapper.trigger('searchable:change');
},

/**
 * Получить выбранное значение
 */
getValue(id) {
    const $wrapper = $(`.searchable-select[data-id="${id}"]`);
    return $wrapper.find('.searchable-value').val() || '';
},

/**
 * Установить значение программно
 */
setValue(id, value) {
    const $wrapper = $(`.searchable-select[data-id="${id}"]`);
    if (!$wrapper.length) return;

    const $option = $wrapper.find(`.searchable-option[data-value="${value}"]`);
    if ($option.length) {
        $wrapper.find('.searchable-input').val($option.text());
        $wrapper.find('.searchable-value').val(value);
    } else {
        $wrapper.find('.searchable-input').val('');
        $wrapper.find('.searchable-value').val('');
    }
}

```

```

    }
  },

  /**
   * Добавить новую опцию в dropdown
   */
  addOption(id, item) {
    const $wrapper = $('`searchable-select[data-id="${id}"]`');
    if (!$wrapper.length) return;

    const $dropdown = $wrapper.find('.searchable-dropdown');

    const $option = $('<div>')
      .addClass('searchable-option')
      .attr('data-value', item.code)
      .attr('data-code', item.code.toLowerCase())
      .attr('data-label', item.label.toLowerCase())
      .attr('title', item.description || '')
      .text(`${item.code} – ${item.label}`);

    $dropdown.append($option);

    // Автоматически выбрать новую опцию
    this._selectOption($wrapper, $option);
  }
};

// =====
// МОДАЛЬНОЕ ОКНО ДОБАВЛЕНИЯ ЗНАЧЕНИЯ
// =====
const AddValueModal = {
  $modal: null,
  currentSheet: null,
  onSuccess: null,

  /**
   * Инициализация модального окна
   */
  init() {

```

```

    if (this.$modal) return;

    const msg = Config.messages.addValue;

    const modalHtml = `
        <div id="add-value-modal" class="add-value-modal-overlay" style="display:
none;">

            <div class="add-value-modal">
                <div class="add-value-modal-header">
                    <h4>${msg.title}: <span id="add-value-sheet-name"></span></h4>
                    <button type="button" class="add-value-modal-
close">&times;</button>
                </div>
                <div class="add-value-modal-body">
                    <div class="form-group">
                        <label for="add-value-code">${msg.code} *</label>
                        <input type="text" id="add-value-code" class="form-
control"
                            pattern="[A-Z0-9]+" style="text-transform:
uppercase; font-family: monospace;">
                    </div>
                    <div class="form-group">
                        <label for="add-value-label">${msg.label} *</label>
                        <input type="text" id="add-value-label" class="form-
control">
                    </div>
                    <div class="form-group">
                        <label for="add-value-
description">${msg.description}</label>
                        <input type="text" id="add-value-description" class="form-
control">
                    </div>
                    <div id="add-value-error" class="alert alert-danger"
style="display: none;"></div>
                </div>
                <div class="add-value-modal-footer">
                    <button type="button" class="btn btn-default add-value-
cancel">${msg.cancel}</button>
                    <button type="button" class="btn btn-primary add-value-

```

```

submit">${msg.submit}</button>

        </div>

    </div>

</div>

`;

    $('body').append(modalHtml);
    this.$modal = $('#add-value-modal');

    this._bindEvents();
},

/**
 * Привязка событий
 */
_bindEvents() {
    const self = this;

    // Закрытие модального окна
    this.$modal.on('click', '.add-value-modal-close, .add-value-cancel',
function() {
        self.hide();
    });

    // Клик на overlay закрывает
    this.$modal.on('click', function(e) {
        if ($(e.target).hasClass('add-value-modal-overlay')) {
            self.hide();
        }
    });

    // Escape закрывает
    $(document).on('keydown', function(e) {
        if (e.key === 'Escape' && self.$modal.is(':visible')) {
            self.hide();
        }
    });

    // Отправка формы

```

```

        this.$modal.on('click', '.add-value-submit', function() {
            self._submit();
        });

        // Enter в полях отправляет форму
        this.$modal.on('keydown', 'input', function(e) {
            if (e.key === 'Enter') {
                e.preventDefault();
                self._submit();
            }
        });

        // Автоматический uppercase для кода
        $('#add-value-code').on('input', function() {
            $(this).val($(this).val().toUpperCase().replace(/^[A-Z0-9]/g, ''));
        });
    },

    /**
     * Показать модальное окно
     */
    show(sheetName, onSuccess) {
        this.init();
        this.currentSheet = sheetName;
        this.onSuccess = onSuccess;

        // Название листа
        const sheetLabels = {
            'TYPE': Config.messages.labels.type,
            'PRODUCT': Config.messages.labels.product,
            'CAT': Config.messages.labels.cat
        };

        $('#add-value-sheet-name').text(sheetLabels[sheetName] || sheetName);

        // Очистка полей
        $('#add-value-code').val('');
        $('#add-value-label').val('');
        $('#add-value-description').val('');
        $('#add-value-error').hide();
    }
};

```

```

        this.$modal.fadeIn(200);
        $('#add-value-code').focus();
    },

    /**
     * Скрыть модальное окно
     */
    hide() {
        this.$modal.fadeOut(200);
        this.currentSheet = null;
        this.onSuccess = null;
    },

    /**
     * Показать ошибку
     */
    _showError(message) {
        $('#add-value-error').text(message).show();
    },

    /**
     * Отправка данных
     */
    async _submit() {
        const code = $('#add-value-code').val().trim();
        const label = $('#add-value-label').val().trim();
        const description = $('#add-value-description').val().trim();

        // Валидация
        if (!code) {
            this._showError(Config.messages.addValue.invalidCode);
            $('#add-value-code').focus();
            return;
        }

        if (!/^[A-Z0-9]+$/.test(code)) {
            this._showError(Config.messages.addValue.invalidCode);
            $('#add-value-code').focus();
        }
    }
}

```

```

        return;
    }

    if (!label) {
        this._showError('Введите название');
        $('#add-value-label').focus();
        return;
    }

    // Проверка на дубликат
    const existingCodes = DataLoader.cache[this.currentSheet]?.map(item =>
item.code) || [];
    if (existingCodes.includes(code)) {
        this._showError(Config.messages.addValue.duplicateCode);
        $('#add-value-code').focus();
        return;
    }

    // Блокировка кнопки
    const $submitBtn = this.$modal.find('.add-value-submit');
    $submitBtn.prop('disabled', true).text('Добавление...');

    try {
        await this._sendToAppsScript({
            sheet: this.currentSheet,
            code: code,
            label: label,
            description: description
        });

        // Добавляем в локальный кэш
        const newItem = { code, label, description };
        if (DataLoader.cache[this.currentSheet]) {
            DataLoader.cache[this.currentSheet].push(newItem);
        }

        // Обновляем known values в TagParser
        if (TagParser.knownValues[this.currentSheet]) {
            TagParser.knownValues[this.currentSheet].push(code);
        }
    }

```

```

        // Перестраиваем strict pattern
        TagParser.setKnownValues(DataLoader.cache);
    }

    // Callback
    if (this.onSuccess) {
        this.onSuccess(newItem);
    }

    this.hide();
    console.log('[OfferTagManager] Value added:', newItem);

} catch (error) {
    console.error('[OfferTagManager] Error adding value:', error);
    this._showError(Config.messages.addValue.error + ': ' + error.message);
} finally {
    $submitBtn.prop('disabled', false).text(Config.messages.addValue.submit);
}
},

/**
 * Отправка в Google Apps Script
 */
async _sendToAppsScript(data) {
    if (Config.appsScriptUrl === 'YOUR_APPS_SCRIPT_WEB_APP_URL') {
        throw new Error('Apps Script URL не настроен. См. инструкцию в
INSTALL.md');
    }

    const response = await fetch(Config.appsScriptUrl, {
        method: 'POST',
        mode: 'no-cors', // Apps Script требует no-cors для POST
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify(data)
    });

    // no-cors не возвращает тело ответа, считаем успехом если нет исключения

```

```

        return true;
    }
};

// =====
// ЧТЕНИЕ ТЕГОВ ИЗ GC-TAGS-EDITABLE
// =====
const TagReader = {
    $container: null,

    /**
     * Инициализация
     */
    init() {
        this.$container = $(Config.selectors.tagsContainer).first();
        return this.$container.length > 0;
    },

    /**
     * Получить текущие теги
     */
    getCurrentTags() {
        const tags = [];

        // Способ 1: из hidden inputs (самый надёжный)
        $(Config.selectors.tagsContainer)
            .find(`input[name="${Config.selectors.tagsInputName}"]`)
            .each(function() {
                const val = $(this).val();
                if (val && val.trim()) {
                    tags.push(val.trim());
                }
            });

        if (tags.length > 0) {
            console.log('[OfferTagManager] Found tags from inputs:', tags);
            return tags;
        }
    }
};

```

```

        // Способ 2: из data-атрибута
        const dataTags = $(Config.selectors.tagsContainer).attr('data-tags');
        if (dataTags) {
            const parsed = dataTags.split(',').map(t => t.trim()).filter(Boolean);
            console.log('[OfferTagManager] Found tags from data-tags:', parsed);
            return parsed;
        }

        console.log('[OfferTagManager] No tags found');
        return [];
    },

    /**
     * Найти обязательные теги в списке
     */
    findMandatoryTags() {
        const allTags = this.getCurrentTags();
        const result = TagParser.findMandatoryTags(allTags);
        console.log('[OfferTagManager] Mandatory tags check:', result);
        return result;
    }
};

// =====
// ВАЛИДАЦИЯ
// =====
const Validation = {
    $saveButton: null,
    $copyButton: null,

    /**
     * Инициализация
     */
    init() {
        this.$saveButton = $(Config.selectors.saveButton);
        this.$copyButton = $(Config.selectors.copyButton);

        this._interceptFormSubmit();
        this._setupTagsObserver();
    }
};

```

```

},

/**
 * Перехват отправки формы
 */
_interceptFormSubmit() {
    const $form = $(Config.selectors.offerForm);
    const self = this;

    $form.on('submit', function(e) {
        const result = self.validate();

        if (!result.valid) {
            e.preventDefault();
            e.stopImmediatePropagation();

            UIBuilder.updateStatus(result.error, 'danger');

            // Прокрутка к блоку
            const $container = $('#${Config.ui.containerId}');
            if ($container.length) {
                $('html, body').animate({
                    scrollTop: $container.offset().top - 100
                }, 300);
            }

            alert(result.error);
            return false;
        }
    });
},

/**
 * Наблюдатель за изменениями в контейнере тегов
 */
_setupTagsObserver() {
    const $tagsContainer = $(Config.selectors.tagsContainer);
    if (!$tagsContainer.length) return;

```

```

const self = this;
const observer = new MutationObserver(() => {
    console.log('[OfferTagManager] Tags changed, re-validating...');
    self.updateButtonsState();
});

observer.observe($tagsContainer[0], {
    attributes: true,
    childList: true,
    subtree: true,
    characterData: true
});
},

/**
 * Валидация
 */
validate() {
    const mandatory = TagReader.findMandatoryTags();

    // Проверка 1: тег должен быть
    if (mandatory.count === 0) {
        return { valid: false, error: Config.messages.missingTag };
    }

    // Проверка 2: ровно один тег
    if (mandatory.count > 1) {
        return { valid: false, error: Config.messages.multipleTags };
    }

    // Проверка 3: тег валиден (из известных значений)
    if (!TagParser.isValidMandatoryTag(mandatory.first)) {
        return { valid: false, error: Config.messages.invalidTag };
    }

    return { valid: true, error: null, tag: mandatory.first };
},

/**

```

```

    * Обновить состояние кнопок
    */
updateButtonsState() {
    const result = this.validate();

    if (result.valid) {
        this.$saveButton.prop('disabled', false);
        this.$copyButton.prop('disabled', false);
        UIBuilder.updateStatus(Config.messages.tagFound + ': ' + result.tag,
'success');
    } else {
        this.$saveButton.prop('disabled', true);
        this.$copyButton.prop('disabled', true);
        UIBuilder.updateStatus(result.error, 'danger');
    }
}

};

// =====
// ПОСТРОЕНИЕ UI
// =====
const UIBuilder = {
    /**
     * Создать основной контейнер
     */
    createTagSelector(sheetsData) {
        const $container = $('<div>')
            .attr('id', Config.ui.containerId)
            .addClass('panel panel-info')
            .css({
                marginTop: '15px',
                marginBottom: '15px'
            });

        // Заголовок панели
        const $heading = $('<div>')
            .addClass('panel-heading')
            .html(`<strong>${Config.messages.labels.title}</strong>`);

```

```

// Тело панели
const $body = $('<div>')
    .addClass('panel-body')
    .css('paddingBottom', '10px');

// Строка с селектами
const $row = $('<div>').addClass('row');

// TYPE селект
$row.append(this._createSelectColumn('type', Config.messages.labels.type,
sheetsData.TYPE));

// PRODUCT селект
$row.append(this._createSelectColumn('product',
Config.messages.labels.product, sheetsData.PRODUCT));

// CAT селект
$row.append(this._createSelectColumn('cat', Config.messages.labels.cat,
sheetsData.CAT));

// Превью тега с кнопкой копирования
const $preview = $('<div>')
    .addClass('col-md-3 col-sm-6')
    .html(`
        <div class="form-group" style="margin-bottom: 0">
            <label>${Config.messages.labels.preview}</label>
            <div style="display: flex; align-items: center; gap: 8px;">
                <div id="mandatory-tag-preview"
                    style="font-family: monospace; font-weight: bold; font-
size: 15px;
                                padding: 7px 12px; min-height: 34px; color: #999;
                                border: 1px dashed #ccc; border-radius: 4px;
background: #f9f9f9;">
                    —
                </div>
                <button type="button" id="copy-mandatory-tag"
                    style="display: none; border: none; background: none;
cursor: pointer;
                                padding: 6px 10px; border-radius: 4px;

```

```

transition: all 0.2s ease;"

        title="Скопировать тег">
        <svg width="18" height="18" viewBox="0 0 24 24"
fill="none" stroke="#5cb85c" stroke-width="2">
            <rect x="9" y="9" width="13" height="13" rx="2"
ry="2"></rect>
            <path d="M5 15H4a2 2 0 0 1-2-2V4a2 2 0 0 1 2-2h9a2 2 0
0 1 2 2v1"></path>
        </svg>
    </button>
</div>
    <div id="copy-feedback" style="display: none; color: #5cb85c;
font-size: 11px; margin-top: 4px;">
        Скопировано
    </div>
</div>
    `);
    $row.append($preview);

    $body.append($row);

    // Статус
    const $status = $('<div>')
        .attr('id', 'mandatory-tag-status')
        .addClass('alert alert-warning')
        .css({ marginTop: '10px', marginBottom: '0' })
        .text(Config.messages.missingParams);
    $body.append($status);

    $container.append($heading).append($body);

    return $container;
},

/**
 * Создать колонку с searchable select
 */
_createSelectColumn(name, label, options) {
    const selectId = `mandatory-tag-${name}`;

```

```

const sheetName = name.toUpperCase();

const $col = $('<div>').addClass('col-md-3 col-sm-6');

const $group = $('<div>')
    .addClass('form-group')
    .css('marginBottom', '0');

$group.append(
    $('<label>')
        .attr('for', selectId)
        .text(label)
);

// Контейнер для селекта и кнопки добавления
const $inputWrapper = $('<div>')
    .addClass('searchable-select-wrapper')
    .css({ display: 'flex', gap: '6px', alignItems: 'flex-start' });

// Использовать SearchableSelect вместо обычного select
const $searchable = SearchableSelect.create(
    selectId,
    options,
    Config.messages.selectPlaceholder
);
$searchable.addClass(Config.ui.selectClass).attr('data-param', name);
$searchable.css('flex', '1');

// Кнопка добавления нового значения
const $addBtn = $('<button>')
    .attr('type', 'button')
    .addClass('btn btn-default add-value-btn')
    .attr('data-sheet', sheetName)
    .attr('title', 'Добавить новое значение')
    .html('+')
    .css({
        padding: '6px 10px',
        fontSize: '16px',
        fontWeight: 'bold',
    });

```

```

        lineHeight: '1',
        minWidth: '34px',
        height: '34px'
    });

    $inputWrapper.append($searchable).append($addBtn);
    $group.append($inputWrapper);
    $col.append($group);

    return $col;
},

/**
 * Обновить превью тега
 */
updatePreview(tag) {
    const $preview = $('#mandatory-tag-preview');
    const $copyBtn = $('#copy-mandatory-tag');

    if (tag) {
        $preview.text(tag).css('color', '#3c763d');
        $copyBtn.show();
    } else {
        $preview.text('-').css('color', '#999');
        $copyBtn.hide();
    }

    // Скрыть feedback при изменении
    $('#copy-feedback').hide();
},

/**
 * Копировать тег в буфер обмена
 */
copyTagToClipboard() {
    const tag = $('#mandatory-tag-preview').text();
    if (!tag || tag === '-') return;

    navigator.clipboard.writeText(tag).then(() => {

```

```

        // Показать feedback
        const $feedback = $('#copy-feedback');
        $feedback.show();

        // Скрыть через 2 секунды
        setTimeout(() => {
            $feedback.fadeOut();
        }, 2000);

        console.log('[OfferTagManager] Tag copied:', tag);
    }).catch(err => {
        // Fallback для старых браузеров
        const textArea = document.createElement('textarea');
        textArea.value = tag;
        textArea.style.position = 'fixed';
        textArea.style.left = '-999999px';
        document.body.appendChild(textArea);
        textArea.select();

        try {
            document.execCommand('copy');
            $('#copy-feedback').show();
            setTimeout(() => {
                $('#copy-feedback').fadeOut();
            }, 2000);
        } catch (e) {
            alert('Не удалось скопировать. Выделите тег вручную и нажмите
Ctrl+C');
        }

        document.body.removeChild(textArea);
    });
},

/**
 * Обновить статус
 */
updateStatus(message, type) {
    const $status = $('#mandatory-tag-status');

```

```

        $status
            .removeClass('alert-success alert-warning alert-danger alert-info')
            .addClass(`alert-${type}`)
            .text(message);
    },

    /**
     * Заполнить селекты из тега
     */
    fillSelectsFromTag(parsed) {
        if (!parsed) return;

        SearchableSelect.setValue('mandatory-tag-type', parsed.type);
        SearchableSelect.setValue('mandatory-tag-product', parsed.product);
        SearchableSelect.setValue('mandatory-tag-cat', parsed.cat);
    },

    /**
     * Получить значения из селектов
     */
    getSelectValues() {
        return {
            type: SearchableSelect.getValue('mandatory-tag-type'),
            product: SearchableSelect.getValue('mandatory-tag-product'),
            cat: SearchableSelect.getValue('mandatory-tag-cat')
        };
    }
};

// =====
// ГЛАВНЫЙ МОДУЛЬ
// =====
const OfferTagManager = {
    initialized: false,
    sheetsData: null,
    currentMandatoryTag: null,

    /**
     * Проверка страницы

```

```

    */
    isOfferEditPage() {
        return /\pl\sales\offer\update\?id=\d+/.test(window.location.href);
    },

    /**
     * Точка входа
     */
    async init() {
        if (!this.isOfferEditPage()) {
            return;
        }

        console.log('[OfferTagManager] Initializing...');

        try {
            await this._waitForDependencies();

            // Загрузка данных
            this.sheetsData = await DataLoader.loadAllSheets();
            console.log('[OfferTagManager] Data loaded:', this.sheetsData);

            // Установить известные значения
            TagParser.setKnownValues(this.sheetsData);

            // Построить UI
            this._buildUI();

            // События
            this._bindEvents();

            // Валидация
            Validation.init();
            Validation.updateButtonsState();

            // Стили
            this._injectStyles();

            this.initialized = true;

```

```

        console.log('[OfferTagManager] Initialized successfully');

        } catch (error) {
            console.error('[OfferTagManager] Initialization failed:', error);
            this._showError(error.message);
        }
    },

    /**
     * Ожидание зависимостей
     */
    _waitForDependencies() {
        return new Promise((resolve, reject) => {
            let attempts = 0;
            const maxAttempts = 100; // 10 секунд

            const check = () => {
                attempts++;

                if (typeof jQuery !== 'undefined' &&
                    jQuery(Config.selectors.offerForm).length > 0) {
                    resolve();
                    return;
                }

                if (attempts >= maxAttempts) {
                    reject(new Error('Timeout waiting for page elements'));
                    return;
                }

                setTimeout(check, 100);
            };

            check();
        });
    },

    /**
     * Построение UI

```

```

*/
_buildUI() {
    const $selector = UIBuilder.createTagSelector(this.sheetsData);

    // Вставка: после .header-with-tags или перед формой
    const $header = $(Config.selectors.headerWithTags);
    const $offerUpdate = $(Config.selectors.offerUpdate);

    if ($header.length) {
        $header.after($selector);
    } else if ($offerUpdate.length) {
        $offerUpdate.find('h1').first().after($selector);
    } else {
        $(Config.selectors.offerForm).before($selector);
    }
},

/**
 * Привязка событий
 */
_bindEvents() {
    const self = this;

    // Изменение searchable selects
    $(`${Config.ui.selectClass}`).on('searchable:change', function() {
        self._onSelectChange();
    });

    // Кнопка копирования тега
    $('#copy-mandatory-tag').on('click', function() {
        UIBuilder.copyTagToClipboard();
    });

    // Кнопки добавления новых значений
    $('.add-value-btn').on('click', function() {
        const sheetName = $(this).attr('data-sheet');
        const selectId = `mandatory-tag-${sheetName.toLowerCase()}`;

        AddValueModal.show(sheetName, function(newItem) {

```

```

        // Добавить новую опцию в dropdown и выбрать её
        SearchableSelect.addOption(selectId, newItem);
        // Обновить превью тега
        self._onSelectChange();
    });
});
},

/**
 * Обработчик изменения селекта
 */
_onSelectChange() {
    const values = UIBuilder.getSelectValues();

    if (values.type && values.product && values.cat) {
        const newTag = TagParser.buildTag(values.type, values.product,
values.cat);
        this.currentMandatoryTag = newTag;
        UIBuilder.updatePreview(newTag);
        UIBuilder.updateStatus(Config.messages.validTag + '. ' +
Config.messages.copyHint, 'success');
    } else {
        this.currentMandatoryTag = null;
        UIBuilder.updatePreview(null);
        UIBuilder.updateStatus(Config.messages.missingParams, 'warning');
    }
},

/**
 * Показать ошибку
 */
_showError(message) {
    const $error = $('<div>')
        .addClass('alert alert-danger')
        .css({ margin: '15px 0' })
        .html(`
            <strong>Ошибка загрузки:</strong> ${message}<br>
            <small>Попробуйте обновить страницу. Если проблема сохраняется,
обратитесь в поддержку.</small>

```

```

        `);

const $header = $(Config.selectors.headerWithTags);
if ($header.length) {
    $header.after($error);
} else {
    $(Config.selectors.offerForm).before($error);
}

// Заблокировать сохранение при ошибке загрузки
$(Config.selectors.saveButton).prop('disabled', true);
$(Config.selectors.copyButton).prop('disabled', true);
},

/**
 * Внедрение стилей
 */
_injectStyles() {
    if ($('#offer-tag-manager-styles').length) return;

    const styles = `
        <style id="offer-tag-manager-styles">
            /* Main Container */
            #${Config.ui.containerId} {
                border: none;
                border-radius: 8px;
                box-shadow: 0 2px 8px rgba(0,0,0,0.08);
            }
            #${Config.ui.containerId} .panel-heading {
                background: #5bc0de;
                border: none;
                color: #fff;
                padding: 14px 20px;
            }
            #${Config.ui.containerId} .panel-heading strong {
                font-size: 14px;
                letter-spacing: 0.3px;
            }
            #${Config.ui.containerId} .panel-body {

```

```
        background: #fafbfc;
        padding: 20px;
    }

    #${Config.ui.containerId} .form-group label {
        font-weight: 600;
        color: #374151;
        font-size: 12px;
        text-transform: uppercase;
        letter-spacing: 0.5px;
        margin-bottom: 6px;
    }

    /* Status Alert */
    #mandatory-tag-status {
        border: none;
        border-radius: 6px;
        font-size: 13px;
        padding: 12px 16px;
    }

    #mandatory-tag-status.alert-success {
        background: #dff0d8;
        color: #3c763d;
    }

    #mandatory-tag-status.alert-warning {
        background: #fcf8e3;
        color: #8a6d3b;
    }

    #mandatory-tag-status.alert-danger {
        background: #f2dede;
        color: #a94442;
    }

    /* Tag Preview */
    #mandatory-tag-preview {
        background: #f5f5f5 !important;
        border: 2px dashed #dee2e6 !important;
        border-radius: 6px !important;
        font-size: 16px !important;
        letter-spacing: 1px;
```

```
}
#copy-mandatory-tag {
    border-radius: 6px !important;
}
#copy-mandatory-tag:hover {
    background: rgba(91, 192, 222, 0.1) !important;
}
#copy-mandatory-tag:hover svg {
    stroke: #5bc0de;
}
#copy-feedback {
    color: #5bc0de !important;
    font-weight: 500;
}

/* Searchable Select */
.searchable-select {
    position: relative;
}
.searchable-select .searchable-input {
    cursor: pointer;
    background: #fff;
    border: 2px solid #e5e7eb;
    border-radius: 6px;
    padding: 8px 12px;
    font-size: 13px;
    transition: border-color 0.15s ease, box-shadow 0.15s ease;
}
.searchable-select .searchable-input:hover {
    border-color: #d1d5db;
}
.searchable-select .searchable-input:focus {
    cursor: text;
    border-color: #5bc0de;
    outline: none;
}
.searchable-dropdown {
    position: absolute;
    top: calc(100% + 4px);
```

```
    left: 0;
    right: 0;
    z-index: 1000;
    max-height: 350px;
    overflow-y: auto;
    border: 2px solid #e5e7eb;
    border-radius: 8px;
    background: #fff;
    display: none;
    box-shadow: 0 10px 40px rgba(0,0,0,0.12);
}

.searchable-dropdown::-webkit-scrollbar {
    width: 6px;
}

.searchable-dropdown::-webkit-scrollbar-track {
    background: #f1f1f1;
    border-radius: 3px;
}

.searchable-dropdown::-webkit-scrollbar-thumb {
    background: #c1c1c1;
    border-radius: 3px;
}

.searchable-dropdown::-webkit-scrollbar-thumb:hover {
    background: #a1a1a1;
}

.searchable-option {
    padding: 10px 14px;
    cursor: pointer;
    font-family: 'SF Mono', 'Monaco', 'Consolas', monospace;
    font-size: 13px;
    color: #374151;
    border-bottom: 1px solid #f3f4f6;
    transition: background-color 0.1s ease;
}

.searchable-option:last-child {
    border-bottom: none;
}

.searchable-option:hover {
    background: #f9fafb;
```

```
}

.searchable-option.active {
    background: #5bc0de;
    color: #fff;
}

.searchable-option.hidden {
    display: none;
}

/* Add Value Button */
.add-value-btn {
    background: #5cb85c;
    border: none;
    color: #fff;
    border-radius: 6px;
    font-weight: 600;
}

.add-value-btn:hover {
    background: #449d44;
    color: #fff;
}

/* Modal Overlay */
.add-value-modal-overlay {
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background: rgba(17, 24, 39, 0.6);
    backdrop-filter: blur(4px);
    z-index: 10000;
    display: flex;
    align-items: center;
    justify-content: center;
}

/* Modal */
.add-value-modal {
```

```
        background: #fff;
        border-radius: 12px;
        width: 100%;
        max-width: 420px;
        margin: 20px;
        box-shadow: 0 20px 60px rgba(0, 0, 0, 0.3);
        overflow: hidden;
    }

    .add-value-modal-header {
        display: flex;
        justify-content: space-between;
        align-items: center;
        padding: 18px 24px;
        background: #5bc0de;
    }

    .add-value-modal-header h4 {
        margin: 0;
        font-size: 15px;
        font-weight: 600;
        color: #fff;
    }

    .add-value-modal-close {
        background: rgba(255,255,255,0.2);
        border: none;
        width: 28px;
        height: 28px;
        border-radius: 6px;
        font-size: 18px;
        color: #fff;
        cursor: pointer;
        display: flex;
        align-items: center;
        justify-content: center;
        transition: background 0.15s ease;
    }

    .add-value-modal-close:hover {
        background: rgba(255,255,255,0.3);
    }

    .add-value-modal-body {
```

```
        padding: 24px;
    }
    .add-value-modal-body .form-group {
        margin-bottom: 18px;
    }
    .add-value-modal-body .form-group:last-of-type {
        margin-bottom: 0;
    }
    .add-value-modal-body label {
        display: block;
        margin-bottom: 6px;
        font-weight: 600;
        font-size: 12px;
        text-transform: uppercase;
        letter-spacing: 0.5px;
        color: #374151;
    }
    .add-value-modal-body .form-control {
        border: 2px solid #e5e7eb;
        border-radius: 6px;
        padding: 10px 12px;
        font-size: 14px;
        transition: border-color 0.15s ease, box-shadow 0.15s ease;
    }
    .add-value-modal-body .form-control:focus {
        border-color: #5bc0de;
        outline: none;
    }
    .add-value-modal-body .alert {
        border: none;
        border-radius: 6px;
        margin-top: 16px;
        font-size: 13px;
    }
    .add-value-modal-footer {
        padding: 16px 24px;
        background: #f9fafb;
        display: flex;
        justify-content: flex-end;
```

```

        gap: 10px;
    }
    .add-value-modal-footer .btn {
        padding: 10px 20px;
        border-radius: 6px;
        font-weight: 500;
        font-size: 14px;
        transition: all 0.15s ease;
    }
    .add-value-modal-footer .btn-default {
        background: #fff;
        border: 2px solid #e5e7eb;
        color: #374151;
    }
    .add-value-modal-footer .btn-default:hover {
        background: #f9fafb;
        border-color: #d1d5db;
    }
    .add-value-modal-footer .btn-primary {
        background: #5bc0de;
        border: none;
        color: #fff;
    }
    .add-value-modal-footer .btn-primary:hover {
        background: #46b8da;
    }
    .add-value-modal-footer .btn-primary:disabled {
        opacity: 0.6;
    }
</style>
`;

$('head').append(styles);
}
};

// =====
// АВТОЗАПУСК
// =====

```

```
if (document.readyState === 'loading') {
    document.addEventListener('DOMContentLoaded', function() {
        setTimeout(function() { OfferTagManager.init(); }, 100);
    });
} else {
    setTimeout(function() { OfferTagManager.init(); }, 100);
}

})();
```

JS GAS

```
/**
 * Google Apps Script для добавления значений в таблицу тегов
 *
 * ИНСТРУКЦИЯ ПО УСТАНОВКЕ:
 * 1. Откройте Google Sheets с данными тегов
 * 2. Меню: Расширения → Apps Script
 * 3. Скопируйте этот код в редактор
 * 4. Нажмите "Развернуть" → "Новое развертывание"
 * 5. Тип: "Веб-приложение"
 * 6. Выполнять как: "Я"
 * 7. Доступ: "Все" (для работы с GetCourse)
 * 8. Нажмите "Развернуть"
 * 9. Скопируйте URL веб-приложения
 * 10. Вставьте URL в Config.appsScriptUrl в offer-tag-manager.js
 */

/**
 * Обработчик POST-запросов
 */
function doPost(e) {
    try {
        const data = JSON.parse(e.postData.contents);

        const sheet = data.sheet;
        const code = data.code;
```

```
const label = data.label;
const description = data.description || '';

// Валидация
if (!sheet || !code || !label) {
    return createResponse(false, 'Missing required fields');
}

if (!/^[A-Z0-9]+$/.test(code)) {
    return createResponse(false, 'Invalid code format');
}

// Получить лист
const ss = SpreadsheetApp.getActiveSpreadsheet();
const targetSheet = ss.getSheetByName(sheet);

if (!targetSheet) {
    return createResponse(false, 'Sheet not found: ' + sheet);
}

// Проверить на дубликат
const existingData = targetSheet.getDataRange().getValues();
for (let i = 1; i < existingData.length; i++) {
    if (existingData[i][0] === code) {
        return createResponse(false, 'Code already exists');
    }
}

// Добавить новую строку
targetSheet.appendRow([code, label, description]);

return createResponse(true, 'Value added successfully');

} catch (error) {
    return createResponse(false, error.message);
}

}
```

/**

```
* Обработчик GET-запросов (для тестирования)
*/
function doGet(e) {
  return createResponse(true, 'Apps Script is working');
}

/**
 * Создание ответа
 */
function createResponse(success, message) {
  const response = {
    success: success,
    message: message
  };

  return ContentService
    .createTextOutput(JSON.stringify(response))
    .setMimeType(ContentService.MimeType.JSON);
}
```

Revision #4

Created 2026-04-07 11:56:49 UTC by David

Updated 2026-04-07 13:25:12 UTC by David