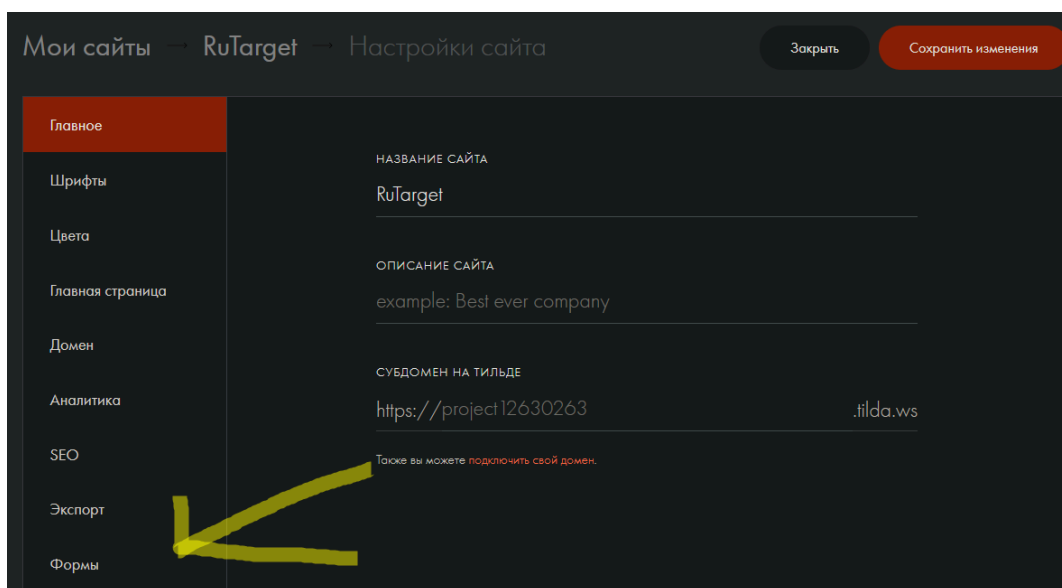


# Руководство разработчика по интеграции Tilda Publishing с Google Forms.

## Настройка Tilda.

1. Перейдите на сайт Tilda.
2. Нажмите «Настройка сайта».
3. В боковом меню слева нажмите на пункт «Формы».



4. Выберите параметр «Google Sheets» для отправки данных с формы.
5. Выдайте название таблице и папке.

## Сервис приёма данных Google.Sheet

При клике на кнопку «Подключить» появится окно, где нужно разрешить приложению доступ. После разрешения страница перезагрузится, и вы сможете выбрать или создать папку для таблицы, а также указать название таблицы.

### НАЗВАНИЕ ТАБЛИЦЫ

Tilda\_Leads\_Test

Укажите название таблицы / [Обновить](#)

### ПАПКА

/Tilda Leads/ - application folder

Выберите папку / [Обновить](#)

### ДОПОЛНИТЕЛЬНЫЕ НАСТРОЙКИ

- ☐ Посылать Cookies
- ☐ Посылать UTM
- ☐ Отправлять только после оплаты

6. Сохраните все указанные изменения.
7. Подключите составленную форму к заявкам Tilda.

Сохранить

Сохранить и закрыть

ШАГИ КВИЗ-ФОРМЫ

КОНСУЛЬТАНТ

ДРУГОЕ

ПРИЕМ ДАННЫХ ИЗ ФОРМЫ

Отправленные данные будут храниться в **разделе «Заявки»**. Также вы можете подключить сервисы приема данных, интегрированных с Тильдой.

TILDA CRM

☐ AUCHAN\_PRODAVEC

Перейти в Tilda CRM

ПОДКЛЮЧЕННЫЕ СЕРВИСЫ:

☒ GSHEET2:  
1MHQIRBH4XGJHC2TIF5\_JO84SP8KHJMMSN\_SP2FPM0RU

☒ MEGAPLAN: MP44662712

☒ TELEGRAM: APP-ID-5302446840

Вы можете получать данные на Email, в документ Google или в сервисы, интегрированные с Тильдой: MailChimp, GetResponse, UniSender, SendGrid, amoCRM, Мегаллан и другие. Подключите сервис для приема данных в настройках сайта, **раздел Формы**. Читайте **подробную инструкцию** в нашем Справочном центре.

## Настройка форм Tilda на Google диск.

1. Внутри google диска нажмите «Tilda Leads».
2. Перейдите в указанный путь «Tilda\_Leads\_Test».
3. Выберите «Расширения».
4. Нажмите «Apps Script».
5. Удалите все данные в открывшемся виртуальном окне.
6. Впишите указанные данные в командую строку.

```
// Основная функция для обработки новых заявок
function createSheetForNewLead() {
  try {
    // Открываем основную таблицу
    const mainSpreadsheet = SpreadsheetApp.getActiveSpreadsheet();
    const mainSheet = mainSpreadsheet.getSheetByName("Лист1"); //
    Укажите правильное имя листа
    const data = mainSheet.getDataRange().getValues();

    // Проверяем, есть ли данные (кроме заголовков)
    if (data.length < 2) {
      Logger.log("Нет новых данных для обработки");
      return;
    }

    // Проходим по строкам в обратном порядке (от последней к первой),
```

чтобы обработать последнюю строку

```
for (let i = data.length - 1; i >= 1; i--) {  
  const row = data[i];  
  const processed = row[26]; // Флаг обработки в 37-м столбце (индекс  
36)
```

```
  Logger.log(`Проверка строки ${i + 1}, processed: ${processed}`);
```

```
  // Проверяем, не обработана ли строка
```

```
  if (processed !== "Processed") {
```

```
    // Получаем данные для имени файла
```

```
    const clientName = row[0] ?
```

```
    row[0].toString().replace(/[\\\/\?%*:|"<>]/g, "_") : "Клиент_" + i; //  
Имя клиента или номер строки
```

```
    // Обработка временной метки
```

```
    let timestamp;
```

```
    if (row[1] && typeof row[1] === "string" && row[1].trim() !==  
    "") {
```

```
      const dateValue = new Date(row[1].replace(".",  
"-").replace(",", "-")); // Поддержка форматов вроде "21.07.2025"
```

```
      timestamp = !isNaN(dateValue.getTime())
```

```
        ? dateValue.toISOString().replace(/[:.]/g, "-")
```

```
        : new Date().toISOString().replace(/[:.]/g, "-"); // Если
```

```
дата некорректна, используем текущую
```

```
    } else {
```

```
      timestamp = new Date().toISOString().replace(/[:.]/g, "-"); //
```

```
Если нет даты, используем текущую
```

```
    }
```

```
    // Создаём новую таблицу
```

```
    const newSpreadsheet =
```

```
    SpreadsheetApp.create(`Заявка_${clientName}_${timestamp}`);
```

```
    const newSheet = newSpreadsheet.getSheets()[0];
```

```
    // Копируем заголовки (36 пунктов)
```

```
    const headers = data[0].slice(0, 26); // Берем первые 36  
столбцов
```

```
    newSheet.getRange(1, 1, 1, 26).setValues([headers]);
```

```
    // Копируем данные текущей заявки (36 пунктов)
```

```
    const clientData = row.slice(0, 26); // Берем первые 36 столбцов  
текущей строки
```

```
    newSheet.getRange(2, 1, 1, 26).setValues([clientData]);
```

```
    // Перемещаем новую таблицу в нужную папку (опционально)
```

```

    const folderId = "1QurF_3KA1LD8oaFdDSApkp1YbMbwjWki"; //
    Замените на реальный ID папки или оставьте пустым
    if (folderId && folderId.trim() !== "") {
        try {
            const folder = DriveApp.getFolderById(folderId);
            DriveApp.getFileById(newSpreadsheet.getId()).moveTo(folder);
            Logger.log(`Таблица перемещена в папку с ID: ${folderId}`);
        } catch (folderError) {
            Logger.log(`Ошибка при перемещении в папку:
            ${folderError.message}. Таблица осталась в корне.`);
        }
    } else {
        Logger.log("Папка не указана или ID пустой, таблица осталась в
        корне.");
    }

    // Помечаем строку как обработанную с отладочным логом
    const rowNumber = i + 1; // Номер строки в таблице (начиная с 1)
    Logger.log(`Попытка записи "Processed" в строку ${rowNumber},
    столбец 37`);
    mainSheet.getRange(rowNumber, 27).setValue("Processed");
    Logger.log(`Запись в строку ${rowNumber}, столбец 37 выполнена`);

    // Логируем создание файла
    Logger.log(`Создана таблица для строки ${rowNumber}:
    Заявка_${clientName}_${timestamp}`);
    break; // Прерываем цикл после обработки первой необработанной
    строки
    }
    }
} catch (e) {
    Logger.log(`Ошибка: ${e.message}`);
}
}

// Функция для настройки триггера
function setupTrigger() {
    // Удаляем существующие триггеры, чтобы избежать дублирования
    const triggers = ScriptApp.getProjectTriggers();
    triggers.forEach(trigger => ScriptApp.deleteTrigger(trigger));

    // Создаём новый триггер
    ScriptApp.newTrigger("createSheetForNewLead")
        .forSpreadsheet(SpreadsheetApp.getActiveSpreadsheet())
        .onChange()
        .create();
}

```

```
Logger.log("Триггер успешно установлен");  
}
```

7. Измените вручную в 57 строке аргумент `const` на `folderid`.
8. Выберите id папки в google диске.
9. Получите ссылку на обновленный google disk.
10. Обновите данные в 57 строке -- загрузив полученную ссылку в командную строку.
11. Перейдите во вкладку «триггеры».
12. Нажмите на кнопку «добавить триггер».
13. Укажите настройки показанные на скриншоте.
14. Перейдите во вкладку «код».
15. Нажмите «выполнить».
16. Перейдите во вкладку «журнал выполнения».

Готово! Вы подключили google формы к Tilda Publishing.