

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

- [Как доработать систему](#)
- [Примеры скриптов конфигуратора](#)

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

В YouGile есть встроенный редактор кода (конфигуратор), позволяющий полностью переделать систему под себя.

Пользователи YouGile с правами администратора могут модифицировать работу системы под нужды команды при помощи простых скриптов на javascript.

Чтобы попасть в конфигуратор, нужно нажать комбинацию клавиш ctrl + ~

Подробнее про работу конфигуратора — в данном мануале:

<https://ru.yougile.com/media/docs/yougile-api-manual.pdf>

В этом 10-минутном видео показаны примеры реализации скриптов, решающих конкретные задачи:

<https://rutube.ru/play/embed/d3a9eae5e123bc80f3cbe9cc24f2bdf3>

Также с примерами скриптов Вы можете ознакомиться в статье [Примеры скриптов конфигуратора](#)

Видеокейс внедрения YouGile на завод кранов. Для построения бизнес-процессов реализованы доработки с использованием конфигуратора:

<https://rutube.ru/play/embed/a941077f074e9035251fbe7480d67b3f/>

Доработки в YouGile также можно реализовать через REST API, инструкция по ссылке <https://ru.yougile.com/api-v2#/>

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

В статье представлены скрипты, которые вы можете использовать в конфигураторе и редактировать в любых ваших целях.

ВАЖНО: скрипты не проходили полноценного тестирования и не являются штатным функционалом, поэтому можно ожидать, что они будут сбоить или работать неожиданным образом.

Список возможностей, которыми вы можете дополнить YouGile через конфигуратор:

- Убирает адресата из названия задачи в определенной колонке (для интеграции с почтой)
- Множит по клику на карточку задачи через Shift задачи на исполнителей
- Добавляет к названию задачи названия проекта и доски
- Берет первые три цифры из описания и крепит их к задаче числовым стикером
- При клике на карточку задачи через Shift назначать себя исполнителем, а через Alt — еще удалить участников чата
- Прочитывает все чаты по Ctrl + Shift + Backspace
- Открывает созданную задачу, если не зажата Alt. Если зажата → добавляет как обычно
- Создает несколько задач из списка
- Добавляет второго исполнителя ко всем задачам первого исполнителя
- Ограничивает количество задач в колонке
- Отписывается от всех задач доски через кнопку
- Подписка на все чаты задач (при их создании)
- Уведомление в чат о перемещении задачи
- Архивация задач при выполнении
- Считает количество задач в колонке и печатает его в интерфейсе
- Запрашивает обязательный комментарий при перемещении задачи в другую колонку и печатает его в описание
- Назначение пользователя в определенной колонке
- Скрывает задачу не из "белого списка" аккаунтов, если на нее навесить определенный стикер
- Сообщение в чат при удалении исполнителя
- Сообщение в чат при добавлении файла в описание задачи
- Сообщение в чат о прикреплении/изменении дедлайна
- Не дает добавить задачу, если есть с таким же названием в компании
- Отправка сообщения в чат при изменении чеклистов

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

```
const deadline = Stickers.get('Deadline') // стикер

Items.onUpdate = function(obj) { //вызываем при обновлении
if (obj.type === 'Task') { //если задача
const deadlineValue = Stickers.getValue(obj, deadline) //записываем
    if (obj.isCompleted() === true && //если выполнена
        deadlineValue !== undefined && //и дедлайн прикреплен
        deadlineValue.current.deadline < App.time()) { //и истек
        obj.setCompleted(false) //развыполняем
        Notifier.error('Дедлайн истек и задачу не выполнить') //пишем предупреждение
    }}
```

???????? ?????????? ?? ?????????? ??????? ? ????????????????? ????????? (???
[????????????? ? ????????\)](#)

```
const column1 = "7c270d09-33a3-473c-8963-0b1c54a10f25"; //тут пишем нужную колонку
//ниже можно написать и другие колонки по образцу выше

const columns = [column1]; //указываем эти колонки через запятую

function cutFirst(text) {
    return text.substr(text.indexOf(" ") + 1);
}

Items.onUpdate = function (obj) {
    // при обновлении
    if (columns.includes(obj.id)) {
        //если колонка совпадает
        for (const task of obj.list()) {
            //перебираем ее задачи
            if (task.name.includes("@")) {
                //если есть @
                task.name = cutFirst(task.name); //отрезаем название по первому пробелу
            }
        }
    }
}
```

```
}  
};
```

?????? ?? ????? ?? ????????? ?????? ?????? Shift ?????? ?? ?????????????

```
Items.onClick = function (object) { // при клике  
  if (object.type === 'Task') { // на задачу  
    if (Events.isShiftPressed()) { // с зажатым шифтом  
      const originalName = object.name // берем название  
      const executors = Users.getAssigned(object, true) // исполнителей  
      const checklists = Chat.getSubtasks(object) // чеклисты  
      const description = Chat.getDescription(object) // описание  
      const usersInChat = Chat.listUsers(object)  
      function stickersInTask () { // функция для сбора стикеров в задаче  
        const stickers = Stickers.getAll().filter(sticker => Stickers.isPinned(object,  
sticker))  
  
        const values = stickers.map(sticker => ({  
          'key': sticker,  
          'value': Stickers.getValue(object, sticker)}))  
        return values  
      }  
  
      if (executors.length > 1) { // если несколько исполнителей  
        Users.assignUser(executors[0], object) // оставляем в задаче первого  
исполнителя  
  
        object.name = `[${executors[0].name}] ${originalName}` // ставим его имя в  
название  
  
        usersInChat.forEach(user => Chat.removeUser(object, user)) //чистим чат  
        Chat.addUser(object, executors[0]) // подписываем  
        executors.shift() // удаляем из массива  
        executors.forEach(executor => { // у остальных  
          object.up().add(`${executor.name} ${originalName}`) // добавляем по задаче  
          const currentTask = object.up().find(`${executor.name} ${originalName}`) //  
ищем ее  
  
          stickersInTask().forEach(sticker => { //для каждого стикера  
            setTimeout(function () { // таймаут для стабильности  
              Stickers.pin(currentTask, sticker.key, sticker.value) // ставим стикеры  
              Chat.setSubtasks(currentTask, checklists) // чеклисты
```

```

        Chat.setDescription(currentTask, description) // описание
        Users.assignUser(executor, currentTask) // исполнителя
        usersInChat.forEach(user => Chat.removeUser(currentTask, user)) //чистим чат
        Chat.addUser(currentTask, executor) // подписываем
    }, 1000)}})}}
    return false // не открываем задачу
  }
  return true // иначе открываем
}

```

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

```

Items.onAdd = function (object) { //при добавлении
  if (object.type === 'Task' && //задачи
    object.up()) { // на доску
    object.name = `[${object.up().up().up().name}, ${object.up().up().name}] ${object.name}`
  }
  // переименовываем
}
};

```

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

```

const yourNumSticker = Stickers.get('Число') // напишите название своего стикера

Items.onUpdate = function(obj) { // при обновлении
  if (obj.type === 'Task') { // задачи
    let descDump = obj.getData().description; //берем наш дамп описания задачи
    if (descDump === undefined) {
      descDump = ''
    } //если дампа нет, то оставляем его пустым
    const currenDesc = Chat.getDescription(obj); // берем описание задачи
    if (descDump !== currenDesc) { // если разное с дампом
      const twoLetters = parseInt(currenDesc.slice(0, 6).replaceAll(/<\/?[^\>]+(>|$/gi, ""))
    }
    // берем первые знаки как числа
    const numSticker = yourNumSticker // берем стикер
    if (!Number.isNaN(twoLetters)) { // если нормальное число
      Stickers.pin(obj, numSticker, twoLetters) // крепим стикер
    }
  }
}

```

```

    }}
    obj.setData({
      description: currenDesc
    }) // сохраняем текущее описание в дамп на будущее
  }
};

```

??? ????? ?? ????????? ?????? ????? Shift ?????????? ????? ??????????????, ?
 ????? Alt — ??? ????????? ????????????? ?????

```

Items.onClick = function (object) { // при клике
  if (object.type === 'Task') { // на задачу
    if (Events.isShiftPressed()) { // если зажата Shift
      Users.assignUser(Current.user, object); // назначаем себя на задачу
      Chat.addUser(object, Current.user) // добавляем себя в чат
      return false // не открываем задачу
    }
    if (Events.isAltPressed()) { // если зажата Alt
      const listInChat = Chat.listUsers(object) // собираем участников чатов
      listInChat.forEach(user => Chat.removeUser(object, user)) // удаляем их из чата
      Users.assignUser(Current.user, object); // назначаем себя
      Chat.addUser(object, Current.user) // подписываем себя на чат
      return false // не открываем задачу
    }
  }
  return true; // открываем задачу
};

```

????????????? ??? ?????? ?????, ????? ?????????? ??????????????, ?? Ctrl + Shift +
 Backspace

```

Events.onKeyUp = function (event) { // следим за клавиатурой
  if (Events.isShiftPressed() && // когда нажать Shift
    Events.isCtrlPressed() && // и Ctrl
    event.keyCode === 8) { // и Backspace
    const warn = confirm( "Вы действительно хотите прочитать все чаты?" ); //размещаем

```

предупреждение

```
if (warn) { //если все ок
  for (const project of Current.company.list()) // разбиваем компанию на проекты
    for (const boards of project.list()) // проекты на доски
      for (const columns of boards.list()) // доски на колонки
        for (const task of columns.list()) { // колонки на задачи
          if (Chat.isUserInChat(task, Current.user)) { // берем только подписки
            Chat.markRead(task, Current.user); // прочитываем задачи
          }
        }
      }
    }
  }
}
```

????????? ?????????? ??????, ???? ?? ?????? Alt. ???? ?????? ? ??????????
??? ??????

```
Items.onAdd = function (task) { // при добавлении
  if (task.type === 'Task') { // задачи
    !Events.isAltPressed()? Chat.open(task) : true //при незажатой Alt? открывает чат : иначе
    просто добавляет задачу
  }
};
```

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

```
Items.onBeforeAdd = function (location, name) { // перед добавлением
  if (name.includes('\n')) { // если есть перенос строк
    const tasks = name.split('\n').reverse() // режем строки на массивы по переносу обратным
    порядком
    tasks.forEach(name => location.add(name)) //добавляем задачи
    return false //не добавляя изначальную
  }
}
```

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

Для запуска скрипта нужно из-под администратора нажать Ctrl + Shift + / , в первом поле заполнить почту уже существующего исполнителя, во втором — нового исполнителя.

Скрипт включает второго пользователя во все проекты существующего под ролью Сотрудник, не забудьте ее сменить.

Работает только в браузере с возможностью открыть запрос (prompt).

```
Events.onKeyUp = function (event) { // следим за клавиатурой
  if (Events.isShiftPressed() &&
    Events.isCtrlPressed() &&
    Current.user.isAdmin &&
    event.keyCode === 191) { // если вызывает админ через Shift, Ctrl и /
    const assignedUserEmail = prompt('Введите email текущего исполнителя') // запрашиваем
    первый емейл
    const userToAssignEmail = prompt('Введите email нового исполнителя') // запрашиваем второй
    const assignedUser = Users.listAll().find(user => user.email === assignedUserEmail) //
    найдем текущего исполнителя
    const userToAssign = Users.listAll().find(user => user.email === userToAssignEmail)//
    найдем второго исполнителя
    // если емейлы не найдены, то показываем ошибку
    if (assignedUser === undefined) {alert('Email текущего исполнителя не найден')}
    if (userToAssign === undefined) {alert('Email нового исполнителя не найден')}

    if (userToAssign !== undefined && userToAssign !== undefined) { // если все нормально
      for (const project of Current.company.list())
        for (const boards of project.list()) // проекты на доски
          for (const columns of boards.list()) // доски на колонки
            for (const task of columns.list()) { // колонки на задачи
              if (Users.isUserAssigned(assignedUser, task)) { // если старый сотрудник подписан
                const usersInTask = Users.getAssigned(task, true) // получим список уже подписанных
                сотрудников
                const newUsersList = [...usersInTask, userToAssign] // добавим в него нового
                Users.assignUser(newUsersList, task) // применим этот список
                Users.addToChat(userToAssign, task) // подпишем в чат
                if (Users.isInProject(assignedUser, project)) { // в проекты, в которые добавлен текущий
                исполнитель
                  Users.addToProject(userToAssign, project) // подпишем второго исполнителя
```

```
    }  
  }  
}
```

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

//в скрипте нужно заполнить два поля: id колонки в column и ограничение количества задач в ней

```
const column = '0da6cadd-7717-41cd-8dbe-6b571d15301a' // id колонки  
const count = 10 // максимальное количество задач в колонке
```

```
Items.onBeforeAdd = function (location, name) { // смотрим перед добавлением  
  if (location.id === column) { // если колонка совпадает с нужной  
    const columnCount = location.list().filter(x => x.isArchived() === false).length // получаем количество неархивных задач  
    if (columnCount >= count) { // если больше нашего количества  
      alert('В колонке больше 10 задач!') // выводим сообщение  
      return false // запрещаем добавление  
    }  
  } else { // иначе  
    return true // разрешаем  
  }  
};
```

```
Items.onMove = function (object, from, to) { // при перемещении задачи  
  if (to.id === column) { // если колонка назначения совпадает с нужной  
    const columnCount = to.list().filter(x => x.isArchived() === false).length // получаем количество неархивных задач  
    if (columnCount >= count) { // если больше нашего количества  
      alert('В колонке больше 10 задач!') // выводим сообщение  
      return false // запрещаем добавление  
    }  
  } else { // иначе  
    return true // разрешаем  
  }  
};
```

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

Работает только в браузере с возможностью открыть диалог подтверждения (confirm).

```
if (Current.board && Current.project) { //есть есть доска и проект
    Current.onBoardChange = function (oldBoard, newBoard) {
        //при смене доски
        const boardList = Current.project.list(); //собираем все проекты
        boardList.forEach(function (board) {
            //для каждой доски
            const btn = UI.button("Отписаться от всех задач"); //создаем кнопку
            board.ui.clear(); // предварительно очищаем, чтобы удалить предыдущий и не дублировать
            btn.onClick = function (e) {
                //в кнопке
                const warn = confirm(
                    "Вы действительно хотите отписаться от всех задач доски?"
                ); //размещаем предупреждение
                if (warn) {
                    //если все ок
                    for (const column of Current.board.list()) //для колонок в доске
                        for (const task of column.list()) {
                            //для задач в колонке
                            Chat.removeUser(task, Current.user); //отписываем пользователя
                        }
                }
            };
            board.ui.add(btn); //добавляем кнопку в интерфейс
        });
    };
};
```

???????? ?? ??? ????? ????? (??? ?? ??????????)

Скрипт работает только для администраторов компании, внутри скрипта нужно указать их список

Не работает для подзадач.

```
// Нужно заполнить e-mail-ы администраторов в listOfEmails, которые будут подписываться на чаты
```

```
const listOfEmails = ["example1@example.com", "example2@example.com"]; // список emailов для подписки

const findId = listOfEmails.map((email) => Users.get(email)); // переведем почты в id
const justAdmins = findId.filter((user) => user.isAdmin); // отфильтруем только администраторов

Items.onAdd = function (task) {
  // при добавлении
  if (task.type === "Task") {
    // задачи
    justAdmins.forEach((admin) => Chat.addUser(task, admin)); // подписываем на чат
  }
};
```

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

Не работает для подзадач.

```
Items.onMove = function (task, from, to) { // при перемещении
  if (task.type === 'Task') { // задачи
    Chat.postMessage(task, '', `Сотрудник ${Current.user.name} переместил задачу из колонки «${from.name}» в колонку «${to.name}».`) // пишем сообщение в чат
  }
};
```

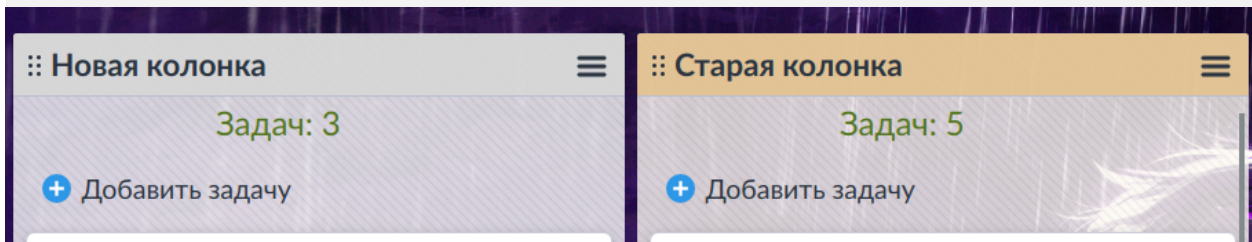
??????????? ?????? ??? ?????????????

Может быть реализовано штатным функционалом YouGile: перейдите в Настройки доски > Основные и убедитесь, что галочка Автоматически архивировать выполненные задачи включена.

```
// Архивирует задачи при их выполнении
Items.onUpdate = function(obj) { // вызываем при обновлении
  if (obj.isCompleted() === true) { // если задача выполнена
    obj.setTaskCompleted(true); // то архивируем ее
  }
}
```

```
};
```

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



```
// Пишет количество задач в колонках при отрисовке страницы и обновлении объектов.
// Вносить в него ничего не надо, можно в p.style поменять что-то на свой вкус
if (Current.board){ //если есть доска
Current.onBoardChange = function (oldBoard, newBoard) { //работаем при смене доски
  const listOfColumns = Current.board.list() //получаем список колонок
  listOfColumns.forEach(function (i) { // перебираем каждую колонку
    if (!Items.get(i.id).isMirror() && !Items.get(i.id).isReport()) { // если колонка не
зеркало и не сводка
      const column = Items.get(i.id); // получаем id колонки
      const numberTasks = column.list() // берем общее количество задач в колонке
      const tasksIsNotArchived = numberTasks.filter(x => x.isArchived() === false) // заберем
задачи не в архиве
      const p = UI.panel(); // создаем контейнер
      column.ui.clear() // предварительно очищаем, чтобы удалить предыдущий и не дублировать
      column.ui.add(p);
      p.style = {
        margin: 'auto',
        width: '100px',
        padding: 'auto',
        color: '#572'
      }; // оформляем контейнер
      p.add(UI.text('Задач: ' + tasksIsNotArchived.length)); // пишем текст
    })
  }

Items.onUpdate = function (obj) { //вызываем функцию при обновлении,
  // чтобы работало в реальном времени
  Current.onBoardChange()
```

```
};}
```

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

Работает только в браузере с возможностью открыть запрос (prompt).

```
// Запрашивает обязательный комментарий при перемещении задачи
Items.onMove = function(obj, from, to) {
  if (to.id === 'id колонки') { //если id колонки совпадает
    let comment = prompt('Введите комментарий') //запрашиваем комментарий
    if (!comment) { //если комментария нет
      return false // отказываем
    } else { //иначе
      Chat.postMessage(obj, '', `Задача перемещена пользователем ${Current.user.name}`)
      //пишем в чат, что переместили задачу
      Chat.setDescription(obj, Chat.getDescription(obj) + '<br> <p><b>Комментарий  
пользователя ' + Current.user.name + ': </b></p>' +
        '<p>' + comment + '</p>') //пишем в описание комментарий
    }
    return true
  }
}
```

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

```
// Добавляет исполнителя в задачу, если ее переместить в колонку и у задачи нет исполнителя.
// Надо вписать id колонки и id пользователя.
// Для текущего пользователя (кто перемещает задачу) можно вместо id пользователя вписать
Current.user.id без кавычек, это сделает его исполнителем
Items.onMove = function(obj, from, to) {
  const task = Items.get(obj.id) //берем id taska
  const user = Users.get('id пользователя'); //берем id пользователя
  if (to.id === '<id колонки>' && !Users.getAssigned(task)) { //если id колонки совпадает
    //и в задаче нет исполнителя
  }
}
```

```
Users.assignUser(user, task); // назначаем пользователя
}
};
```

???????? ???? ? ? "????? ?????" ?????????, ???? ? ? ? ? ? ? ?
???????????? ? ? ? ?

```
// Скрывает задачу, если на него закреплен стикер (в примере – состояние "Запрет" для стикера
"Ограничение")
// userRestrict – массив, в который вносятся email, для которых задача скрывается
Items.isHidden = function(object) { // функция скрывает
    const stickerRestrict = Stickers.get('Ограничение'); // забираем нужный текстовый стикер
    const allowToSee = Stickers.getValue(object, stickerRestrict); //забираем состояние этого
    стикера
    const userRestrict = ['example1@example.com', 'example2@example.com']; //массив почт, для
    которых задача скрывается
    function checkEmail(email) { // функция, которая возвращает совпадение запрещенного списка
    с текущим пользователем
        return email === Current.user.email
    }
    const restrict = userRestrict.find(checkEmail); // ищет совпадение

    if (object.type === 'Task' && allowToSee === 'Запрет' && restrict !== undefined) { //если
    задача со стикером Запрет
        // и текущий пользователь попадает в список
        return true // скрываем
    }
    return false; // в остальных случаях должно быть открыто
};
```

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

```
Stickers.onUnpin = function(task, sticker) { //при откреплении стикера
    if (sticker.name === 'User') { //если стикер – исполнитель
        const userID = Stickers.getValue(task, sticker) //берем id исполнителя
        const userName = Users.get(userID).name //получаем его имя
```

```

        Chat.postMessage(task, '', `${userName} удален из исполнителей задачи`) // пишем в чат задачи
    }
    return true;
}

```

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

```

//Ниже – единожды запускаемый процесс просмотра всех задач для фиксации дампа описания
for (const project of Current.company.list()) // разбиваем компанию на проекты
    for (const boards of project.list()) // проекты на доски
        for (const columns of boards.list()) // доски на колонки
            for (const task of columns.list()) { //колонки на задаче
                const dump = task.getData() // читаем дамп
                const desc = Chat.getDescription(task) //и описание
                if (dump.description !== desc) { // если не совпадают
                    dump.description = desc
                    task.setData(dump) } // пишем дамп
            }
//далее работаем по обновлению задач
Items.onUpdate = function(obj) { // при обновлении
    if (obj.id === Current.chat.id) { //фильтруем по открытой задаче (избегаем множества срабатываний)
        const dump = obj.getData() //читаем дамп
        const currentDesc = Chat.getDescription(obj); // берем описание задачи
        const diff = findDiff(dump.description, currentDesc) // сравниваем и находим разницу
        if (diff.includes('rel="noopener noreferrer"')) { // если в разнице есть свойство вложения
            Chat.postMessage(obj, '', 'Файл добавлен в описание') // пишем в чат
        }
        dump.description = currentDesc //переписываем дамп в любом случае
        obj.setData(dump) // сохраняем текущее описание в дамп на будущее
    }
};

function findDiff(str1, str2) { // это функция, которая сравнивает две строки и выводят между ними разницу
    let diff = "";

```



```

str2.split('').forEach(function(val, i) {
    if (val !== str1.charAt(i))
        diff += val;
});
return diff;
}

```

????????? ? ??? ? ?????????????/????????? ????????

```

//Реагирует на создание и изменение дедлайна задачи, отписывая сообщение в чат задачи.
//Удаление дедлайна игнорирует
Stickers.onPin = function(task, sticker, value) { //при прикреплении стикера
    if (sticker.type === 'Deadline' && value !== undefined) { //если это дедлайн и у него у
него есть значение
        Chat.postMessage(task, '', 'Прикреплен/изменен дедлайн'); //пишем сообщение в чат
    }
};

```

?? ????? ????????? ???????, ????? ????? ? ?????? ?? ?????????? ? ??????????

Не работает для подзадач.

```

Items.onBeforeAdd = function (location, name) { // перед добавлением
if (location.type === 'Column') // задачи
    for (const project of Current.company.list()) // разбиваем компанию на проекты
        for (const boards of project.list()) // проекты на доски
            for (const columns of boards.list()) // доски на колонки
                for (const task of columns.list()) { // колонки на задачи
                    if (name === task.name) { // если название совпадает
                        Notifier.error('Задача с таким названием уже есть'); // показываем ошибку
                        return false // запрещаем
                    }
                }
    return true // разрешаем, если ничего не запретилося
}

```

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

Если чат задачи открыт несколькими сотрудниками, то сообщения в чат будут дублироваться

Изменение чеклиста.webm

```
Items.onUpdate = function(obj) { //при обновлении
    if (obj.id === Current.chat.id) { //если объект – текущий чат
        const checkListsFact = Chat.getSubtasks(obj); // получаем чеклисты
        const checkListsDump = obj.getData().checklists; // получаем дамп записи чеклистов
        if (JSON.stringify(checkListsFact) !== JSON.stringify(checkListsDump)) { //если дамп и
чеклисты не совпадают
            if (checkListsDump === undefined) { //если дамп пуст
                obj.setData({
                    checklists: checkListsFact
                }); //записываем дамп
            } else if (checkListsFact.length !== checkListsDump.length) { //иначе сравниваем
длину чеклиста и дампа.
                // если разница есть → что-то делали со списком чеклистов
                obj.setData({
                    checklists: checkListsFact
                }); // записываем дамп
                Chat.postMessage(obj, Users.get(''), 'Изменен список чеклиста') // пишем в чат
об изменении списка чеклиста
            } else { //иначе → если менялся не список чеклиста, то менялся пункт чеклиста
                obj.setData({
                    checklists: checkListsFact
                }); // записываем дамп
                Chat.postMessage(obj, Users.get(''), 'Изменен пункт чеклиста') // пишем в чат,
что менялся пункт чеклиста
            }
        }
    }
};
```

????????? id ?????? ? ?????? ?????? ?? Ctrl + Alt + C

```
Events.onKeyDown = function (event) {
    if (
```

```
Current.chat &&
Current.chat.type === "Task" && //если задача
Events.isCtrlPressed() && //и зажат контрол
Events.isAltPressed() && //и альт
event.keyCode === 67 // и C
) {
  const companyTail = Current.company.id.substring(Current.company.id.length - 12) //
режим хвостик
  const taskTail = Current.chat.id.substring(Current.chat.id.length - 12); //режим
хвостик
  App.copyToClipboard(
    `https://yougile.com/team/${companyTail}/#chat:${taskTail}`,
  ); // в буфер
  Notifier.success("Ссылка на тикет скопирована"); // уведомление
}
}
```