# Зависимости задач в /task: блокирующие задачи, списки и замок на карточке

16.11.2026 · PDF-версия статьи с сайта cms.dolinais.ru

В проектной работе редко бывает так, что задачи существуют независимо друг от друга. Обычно выстроена цепочка: дизайн ждёт согласования макетов, релиз ждёт прохождения тестов, отчёт ждёт данных из аналитики. Если эту цепочку держать в голове, рано или поздно кто-то закроет задачу, которая фактически ещё не готова, и сорвёт сроки. В /task эта проблема решается формально — через блокирующие связи между задачами. Ниже разберём, что это за механизм, зачем он нужен и как с ним работать на практике.

## Живое демо /task

Ниже — интерактивная демонстрация /task: можно переключать виды, открывать задачи и пробовать действия. Удобно посмотреть, как выглядит карточка с блокировками вживую.

ДЕМО /TASK — КАНБАН: ПРИМЕР ДАННЫХ

Задачи в блоке не заданы.

## Что такое блокирующие задачи

**Блокирующая задача** — это задача, которая должна быть завершена раньше, чем можно считать готовой другую задачу. В /task такая связь задаётся направленно: одна задача выступает блокиратором, другая — зависимой. Направление важно: если задача B заблокирована задачей A, то A блокирует B, но не наоборот. Это не «связанные задачи» и не подзадачи — это именно отношение «сначала должно закрыться одно, потом другое».

Отдельно стоит различать уровни:

- **Подзадача** — часть работы внутри задачи, её удобно дробить по этапам.
- **Связанная задача** — соседняя по смыслу, но без строгого порядка.
- **Блокирующая связь** — жёсткий порядок завершения, который влияет на возможность закрыть задачу.

Путать их не стоит: подзадача не обязана завершаться до родителя по определению (хотя часто так происходит), а блокиратор — обязан.

## Два списка на карточке: «Ждёт завершения» и «Блокирует»

На карточке задачи блокировки видны в двух списках — и это ключ к пониманию механизма.

### Список «Ждёт завершения»

Здесь собраны задачи-блокираторы, которые должны закрыться раньше текущей. Простыми словами: «эта задача ждёт, пока завершат эти». Пока в списке есть незакрытые элементы, задача считается заблокированной.

Именно этот список определяет замок на карточке.

### Список «Блокирует»

Здесь обратная сторона: какие задачи не смогут закрыться, пока не завершится текущая. Это удобно для контроля последствий. Менеджер открывает карточку и сразу видит, что задержка одной задачи «держит» ещё три — и может приоритизировать работу осознанно.

> Полезное правило: если в списке «Блокирует» у задачи много пунктов, а в «Ждёт завершения» пусто — это потенциально узкое место проекта. Стоит проверить, не является ли она критическим путём.

Оба списка доступны прямо в карточке задачи, без перехода в отдельный раздел проекта. Это экономит время: не нужно держать в голове граф зависимостей, он виден на месте.

## Замок на карточке и запрет завершения

**Замок на карточке** — визуальный индикатор того, что задача заблокирована. Он появляется, когда в списке «Ждёт завершения» есть хотя бы одна незавершённая задача. Замок — не просто иконка: за ним стоит правило, которое не даёт закрыть задачу, пока блокираторы не завершены.

Зачем это нужно? Причины простые:

1. **Порядок не нарушается случайно.** Нельзя «закрыть на всякий случай», обойдя реальную готовность.
2. **Статусы отражают реальность.** Если задача в статусе «Готово», значит, все её предпосылки выполнены.
3. **Меньше ручных проверок.** Не нужно сверять цепочку глазами — система сделает это сама.
4. **Прозрачность для команды.** Любой участник видит, почему задача ещё не может быть закрыта.

Запрет завершения — это не «ограничение ради ограничения». Он защищает от типичной ошибки, когда формально закрытая задача на деле ещё не готова, а следующий этап уже стартовал на неверных входных данных.

## Как работать с блокировками: пошагово

Вот практический сценарий на примере релиза мобильного приложения.

1. **Определите цепочку.** Например: «Подготовить сборку» → «Прогнать регресс» → «Опубликовать в стор». Каждая следующая задача зависит от предыдущей.
2. **Откройте зависимую задачу** и добавьте блокиратор в список «Ждёт завершения». В обратном списке «Блокирует» он появится автоматически.
3. **Проверьте карточку.** У зависимой задачи должен появиться замок, а у блокиратора — запись в «Блокирует».
4. **Работайте в порядке цепочки.** Сначала закрывается блокиратор, затем замок снимается, и зависимую задачу можно завершать.
5. **Пересматривайте связи.** Если требования изменились и порядок больше не актуален, уберите лишнюю связь, чтобы не блокировать работу зря.

Совет: заводите блокировки только там, где действительно есть жёсткая последовательность. Если две задачи можно делать параллельно, связь между ними не нужна — иначе вы искусственно удлините критический путь.

## Плюсы подхода

- **Предсказуемость сроков.** Блокировки подсвечивают, что реально можно начинать, а что ещё ждёт.
- **Меньше «фантомной готовности».** Закрытая задача означает выполненную работу, а не договорённость.
- **Быстрая диагностика.** Списки «Ждёт завершения» и «Блокирует» сразу показывают узкие места.
- **Прозрачная коммуникация.** Замок на карточке объясняет статус без дополнительных вопросов в чате.
- **Меньше ручной дисциплины.** Правила соблюдает система, а не память менеджера.

В итоге блокирующие задачи — это небольшой, но важный слой проектного управления в /task. Он не усложняет работу, а убирает ручные проверки и снижает риск закрыть то, что ещё не готово. Начните с одной понятной цепочки в текущем проекте: добавьте блокиратор, посмотрите, как появился замок и список «Блокирует», — и механизм станет очевидным.

Источник: https://cms.dolinais.ru/zavisimosti-zadach-v-task-blokiruyuschie-zadachi-spiski-i-zamok-na-kartochke