Замок на карточке в /task: как запрет завершения защищает цепочку работ

21.11.2026 · PDF-версия статьи с сайта cms.dolinais.ru

В проектной работе почти всегда есть задачи, которые нельзя начинать или, тем более, завершать, пока не сделан другой кусок работы. Классический пример: релиз не может уйти в прод, пока не пройдены тесты; макет не отправляется клиенту, пока не согласован брендбук; интеграция с платёжным провайдером не готова, пока не получен доступ к API. В /task такие отношения между задачами оформляются как **зависимости** — формальные связи, которые влияют на статус карточки и поведение интерфейса.

В этой статье разберём именно техническую и практическую механику: что такое блокирующая задача, как работают списки «Ждёт завершения» и «Блокирует», откуда берётся замок на карточке и почему система физически не даёт закрыть задачу раньше её блокираторов. Отдельно разберём типичные ошибки настройки и то, как эта механика помогает срокам.

Живое демо /task

Ниже — интерактивная демонстрация /task: можно переключать виды, открывать задачи и пробовать действия с зависимостями прямо в интерфейсе.

ДЕМО /TASK — КАНБАН: ПРИМЕР ДАННЫХ

Задачи в блоке не заданы.

Что такое блокирующая задача

Блокирующая задача (blocker, блокиратор) — это задача, которая должна быть завершена раньше, чем другая задача сможет считаться выполненной. В отличие от простой заметки «сначала сделай X», блокировка — это запись в модели данных: у задачи B появляется ссылка на задачу A с типом «блокируется A». Из этой записи система сама выводит два симметричных списка.

- **«Ждёт завершения»** (blocked by) — список задач, которые блокируют текущую. Пока хотя бы одна из них не закрыта, текущая считается ожидающей.
- **«Блокирует»** (blocks) — обратный список: какие задачи ждут завершения текущей. Он виден на карточке блокиратора и показывает, что именно «повиснет», если затянуть работу.

Ключевая идея: связь задаётся один раз, а отображается с двух сторон. Не нужно вручную дублировать информацию в описании — достаточно связать две карточки, и оба списка появятся автоматически.

> Зависимость — это не комментарий и не тег. Это структурная связь, которая меняет поведение интерфейса: от подсветки статуса до запрета действий.

Замок на карточке: как выглядит запрет

Когда у задачи есть незакрытые блокираторы, на её карточке появляется визуальный индикатор — условный

замок. Он сообщает две вещи: во-первых, задача сейчас в состоянии ожидания, во-вторых, часть действий с ней ограничена. На практике это означает:

1. Кнопка завершения задачи недоступна или приводит к понятному сообщению о причине.
2. В карточке видно, какие именно задачи нужно закрыть — список «Ждёт завершения» служит чек-листом разблокировки.
3. После закрытия последнего блокиратора замок снимается, и задача снова становится активной.

Запрет завершения до закрытия блокираторов — главная ценность механики. Без него зависимости остаются «на совести» исполнителя: кто-то закроет задачу «на будущее», а потом выяснится, что тесты не пройдены, а макет не согласован. Формальный запрет убирает этот класс ошибок: статус задачи отражает реальность, а не оптимизм.

Как настроить зависимости: пошагово

Порядок действий в /task обычно такой:

1. **Найдите задачу, которую нужно ограничить.** Откройте карточку задачи B, которая должна ждать.
2. **Добавьте блокиратор.** В блоке зависимостей выберите задачу A и укажите тип связи «блокируется». После сохранения A появится в списке «Ждёт завершения» задачи B.
3. **Проверьте обратную сторону.** Откройте карточку A — задача B должна быть в её списке «Блокирует».
4. **Проверьте замок.** У задачи B должна появиться индикация ожидания, а попытка завершить её — блокироваться.
5. **Закройте блокиратор и убедитесь, что связь снялась.** После завершения A задача B разблокируется автоматически.

Полезно сразу договориться в команде о правиле: блокировку ставит тот, кто знает зависимость, а не тот, кто «наткнулся» на неё в последний день. Чем раньше связь появилась в системе, тем раньше её увидят в плане и на диаграмме.

Типичные ошибки и как их избежать

- **Цепочки ради цепочек.** Если задача A блокирует B, а B блокирует C, то закрытие C требует завершения всей цепочки. Это нормально, но стоит проверять, нет ли лишних звеньев: каждая связь — это обязательство.
- **Блокировка «на всякий случай».** Связь, которая никогда не сработает, только запутывает списки. Удаляйте зависимости, потерявшие смысл.
- **Циклы.** A ждёт B, а B ждёт A — тупик, из которого задача не выйдет. Перед сохранением проверяйте, что новая связь не замыкает круг.
- **Игнорирование обратного списка.** «Блокирует» — это индикатор риска: если у вашей задачи длинный список ожидающих, её задержка ударит по многим.
- **Ручное закрытие в обход.** Если замок мешает, причина обычно в незакрытом блокираторе, а не в интерфейсе. Сначала разберитесь с ним.

Зачем это нужно: плюсы механики

Формальные зависимости дают команде несколько практических выгод:

- **Честный статус.** Задачу нельзя закрыть раньше блокиратора, поэтому «готово» означает реально готово.
- **Раннее выявление рисков.** Список «Блокирует» показывает, сколько работы зависит от одной задачи, и помогает расставить приоритеты.
- **Прозрачность для новичков.** Новый участник видит не только задачу, но и её место в цепочке — меньше вопросов «почему нельзя начать».
- **Меньше ручной синхронизации.** Связь задаётся один раз и отображается с двух сторон, без дублирования в описаниях.
- **Основа для планирования.** Зависимости — это фундамент для оценки сроков: цепочка блокировок подсказывает критический путь проекта.

Отдельно стоит сказать про дисциплину. Замок на карточке — не бюрократия, а способ договориться о правилах игры заранее. Когда команда привыкает, что задача закрывается только после блокираторов, споры «а я думал, что уже можно» исчезают: система показывает состояние цепочки объективно.

Короткий итог

Зависимости в /task — это три связанных элемента: блокирующая задача как запись в модели, два симметричных списка («Ждёт завершения» и «Блокирует») как способ её увидеть, и замок на карточке как механизм запрета преждевременного завершения. Настроить связь — дело нескольких кликов, но эффект накапливается: статусы задач перестают врать, риски видны заранее, а план опирается на реальную последовательность работ, а не на предположения.

Источник: https://cms.dolinais.ru/zamok-na-kartochke-v-task-kak-zapret-zaversheniya-zaschischaet-cepochku-rabot