Зависимости задач в /task: почему задачу нельзя закрыть раньше блокиратора

14.11.2026 · PDF-версия статьи с сайта cms.dolinais.ru

В проектной работе редко встречается задача, которая полностью независима. Обычно релиз ждёт тестирования, тестирование — сборки, сборка — согласованных макетов. Если команда закрывает задачи «по ощущению готовности», план быстро расходится с реальностью: отчёт зелёный, а продукт не работает. В /task эту проблему решает механизм зависимостей. В этой статье — практический взгляд на его строгую часть: **запрет завершения задачи, пока не закрыты блокираторы**, и на то, как этот запрет проявляется в интерфейсе.

Смежные материалы о списках и связях уже есть на сайте, поэтому здесь мы сосредоточимся именно на правиле закрытия, замке на карточке и на том, как выстроить рабочий процесс вокруг этого ограничения.

Живое демо /task

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

ДЕМО /TASK — КАНБАН: ПРИМЕР ДАННЫХ

Задачи в блоке не заданы.

Что такое блокирующая задача

Блокирующая задача (блокиратор) — это задача, результат которой необходим для выполнения другой задачи. В /task связь направленная: у каждой задачи есть два встречных списка.

- **«Ждёт завершения» —** задачи, которые должна закрыть эта карточка, прежде чем её саму можно будет завершить. Проще говоря: «я жду вот эти задачи».
- **«Блокирует» —** задачи, которые ждут завершения текущей. Проще говоря: «меня ждут вот эти задачи».

Оба списка — это две стороны одной связи. Если вы добавили задачу А в «Ждёт завершения» задачи В, то в карточке А она автоматически появится в «Блокирует». Дублировать связь с двух сторон не нужно — это важное свойство: оно исключает рассинхрон, когда одна карточка «думает», что ждёт, а другая об этом не знает.

Чем это отличается от подзадачи

Подзадача — это декомпозиция: часть работы внутри родительской задачи. Зависимость — это отношение между задачами, которые могут жить в разных разделах и даже разных проектах. Подзадача может не быть блокиратором, а блокиратор может не быть подзадачей. Путать эти сущности не стоит: декомпозиция отвечает на вопрос «из чего состоит работа», зависимость — на вопрос «что должно быть готово раньше».

Замок на карточке: как выглядит запрет

Когда у задачи есть незакрытые блокираторы, на её карточке появляется **замок —** визуальный индикатор того, что завершение заблокировано. Это не «мягкое» предупреждение, которое можно проигнорировать, а жёсткое правило: пока хотя бы один блокиратор в статусе, отличном от завершённого, задача не закрывается.

Что это даёт на практике:

- **Честный статус.** «Готово» означает, что готовы все предпосылки, а не только текущая работа.
- **Защита от оптимизма.** Разработчик не закроет задачу «на будущее», надеясь, что смежники успеют.
- **Прозрачность.** Замок сразу показывает, где именно застрял поток, и ведёт к конкретным карточкам-блокираторам.

  Правило простое: задача закрывается тогда, когда закрыты все, кого она ждёт. Всё остальное — договорённости, которые легко нарушить под дедлайном.

Как работать с зависимостями: пошагово

1. **Определите предпосылки до старта.** При планировании спросите: «Что должно быть готово, чтобы эта задача вообще имела смысл?» Каждый ответ — кандидат в блокираторы.
2. **Свяжите задачи.** Откройте карточку и добавьте нужные задачи в «Ждёт завершения». Вторая сторона обновится сама.
3. **Проверьте встречные списки.** В карточке блокиратора убедитесь, что в «Блокирует» появились ожидающие задачи — так вы увидите свою нагрузку как «поставщика» результата.
4. **Работайте по порядку.** Начинайте с задач без замка: они не ждут никого. Заблокированные задачи остаются в плане, но не создают ложного ощущения прогресса.
5. **Закрывайте блокиратор — снимайте замок.** Как только последний блокиратор завершён, ограничение снимается, и задачу можно закрыть обычным способом.

Пример из практики

Команда готовит обновление личного кабинета. Задача «Выкатить релиз» ждёт «Пройти регресс» и «Обновить документацию». «Пройти регресс», в свою очередь, ждёт «Починить авторизацию». Пока авторизация не закрыта, регресс заблокирован, а релиз — тем более. Менеджер видит замок на двух карточках и понимает: узкое место — авторизация, а не «медленный релиз». Без такого механизма релиз могли бы закрыть «частично», а проблема всплыла бы у пользователей.

Как не превратить запрет в тормоз

Строгое правило полезно, но у него есть цена: задача может «зависнуть» из-за формальности. Несколько приёмов, которые помогают этого избежать.

- **Не делайте блокиратором всё подряд.** Связывайте только реальные предпосылки. Если задача может выполняться параллельно и результат не взаимозависим, связь не нужна.
- **Дробите крупные блокираторы.** Если блокиратор — большая задача, зависшая надолго, разбейте её: пусть закрывается по частям, снимая замок с зависимых карточек.
- **Пересматривайте связи.** Планы меняются: связь, актуальная месяц назад, сегодня может быть

лишней. Регулярно просматривайте списки «Ждёт завершения» и «Блокирует».
- **Фиксируйте причину.** Если задача действительно больше не зависит от блокиратора, уберите связь осознанно, а не «в обход» правила.

Плюсы подхода

- **Достоверный прогресс.** Процент завершённых задач отражает реальную готовность, а не количество закрытых карточек.
- **Раннее выявление узких мест.** Цепочки зависимостей показывают, где скапливается работа.
- **Меньше переделок.** Смежники не начинают работу по неготовым входным данным.
- **Прозрачная ответственность.** Видно, кто кого ждёт и кто задерживает поток.
- **Единый источник правды.** Двусторонние списки исключают расхождение между карточками.

Запрет завершения — это не бюрократия, а способ сделать план честным. Начните с нескольких ключевых цепочек в текущем спринте: свяжите релиз с тестированием, тестирование — с исправлениями. Замок на карточке быстро покажет, где процесс действительно буксует, и подскажет, что чинить в первую очередь.

Источник: https://cms.dolinais.ru/zavisimosti-zadach-v-task-pochemu-zadachu-nelzya-zakryt-ranshe-blokiratora