Замок на карточке в /task: как блокировки защищают сроки

26.11.2026 · PDF-версия статьи с сайта cms.dolinais.ru

В /task блокировки задач — это не просто пометки для порядка. Это формальный контракт между задачами: одна не может быть завершена, пока не закрыта другая. Такой контракт защищает цепочку работ от случайного или намеренного «преждевременного закрытия» и помогает команде видеть реальный критический путь. В этой статье разберём, как устроен замок на карточке, зачем нужны два списка — «Ждёт завершения» и «Блокирует», — и как выстроить процесс, чтобы блокировки ускоряли, а не тормозили проект.

Живое демо /task

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

ДЕМО /TASK — КАНБАН: ПРИМЕР ДАННЫХ

Задачи в блоке не заданы.

Что такое блокирующая задача и замок на карточке

Блокирующая задача — это задача, которая должна быть завершена до того, как другую задачу можно будет считать выполненной. В /task такая связь отображается на карточке задачи в виде замка: пока блокиратор не закрыт, кнопка завершения для зависимой задачи неактивна. Это и есть **замок на карточке** — визуальный и функциональный запрет завершения.

Важно различать два типа связей:

- **«Ждёт завершения» —** список задач, которые должны быть закрыты, прежде чем текущая задача может быть завершена. Это входящие блокираторы.
- **«Блокирует» —** список задач, которые не могут быть завершены, пока не закрыта текущая. Это исходящие блокировки.

Оба списка видны на карточке задачи. Замок появляется, если в «Ждёт завершения» есть хотя бы одна незакрытая задача. Как только все блокираторы закрыты, замок снимается автоматически — без ручных действий.

Зачем нужен запрет завершения до закрытия блокираторов

Без такого запрета команда сталкивается с типичными проблемами:

- **Ложная готовность.** Задачу закрывают, хотя результат ещё не проверен или не получен от смежника. В отчётах всё зелёное, а на деле работа не сделана.
- **Срыв сроков.** Закрытая задача перестаёт отслеживаться, а её последствия всплывают позже — уже на этапе тестирования или релиза.

- **Потеря контекста.** Новый участник команды не понимает, что от чего зависит, и берётся за задачу, которая ещё не готова к выполнению.

Замок решает эти проблемы на уровне процесса: он не даёт закрыть задачу, пока не выполнены все условия. Это особенно важно в цепочках, где результат одной задачи является входом для другой: дизайн → вёрстка → интеграция → тестирование. Если вёрстку закрыть до утверждения дизайна, интеграция начнётся с неверных макетов.

Как работать с блокировками: пошаговый сценарий

1. **Создайте задачи и связи.** На карточке задачи, которая должна ждать, откройте раздел зависимостей и добавьте блокирующую задачу в список «Ждёт завершения». В блокираторе автоматически появится запись в «Блокирует».
2. **Проверьте замок.** Убедитесь, что на карточке зависимой задачи отображается замок и кнопка завершения неактивна. Это значит, что связь работает.
3. **Закрывайте блокираторы первыми.** Работайте над задачами из списка «Ждёт завершения». Когда все они закрыты, замок снимется, и зависимую задачу можно будет завершить.
4. **Используйте списки для планирования.** Список «Блокирует» показывает, кого вы держите. Если вы задерживаете задачу, команда сразу видит, какие ещё задачи стоят.
5. **Не обходите замок вручную.** Если задача действительно больше не зависит от блокиратора, удалите связь осознанно — но не закрывайте задачу в обход правил.

   Совет: перед стартом спринта просмотрите все задачи с замком. Если блокиратор не может быть закрыт в этом спринте, зависимую задачу лучше не брать в работу — иначе она зависнет.

Пример из практики

Команда разрабатывает мобильное приложение. Задача «Экран оплаты» ждёт завершения задачи «API платежей». Пока API не готов, кнопка завершения экрана оплаты заблокирована. Разработчик видит замок и не может закрыть задачу «по готовности вёрстки». В списке «Блокирует» у задачи «API платежей» отображается «Экран оплаты» — и бэкенд-разработчик понимает, что его задержка блокирует фронтенд. Это заставляет перераспределить приоритеты и ускорить API. В итоге цепочка не рвётся, а сроки остаются реалистичными.

Плюсы и ограничения

Плюсы:

- Прозрачность зависимостей: каждый видит, что от чего зависит.
- Защита от ложного закрытия задач и искажения отчётности.
- Автоматическое снятие замка — не нужно помнить о ручных проверках.
- Помощь в планировании: блокировки показывают критический путь.

Ограничения: блокировки не заменяют декомпозицию. Если задача слишком крупная, лучше разбить её на подзадачи и связать их. Также не стоит создавать циклические зависимости — /task не позволит замку сняться, и задача останется заблокированной навсегда. Проверяйте цепочки перед сохранением.

Итог

Замок на карточке в /task — это простой и надёжный механизм, который защищает сроки за счёт формального запрета завершения до закрытия блокираторов. Списки «Ждёт завершения» и «Блокирует» дают полную картину зависимостей, а автоматическое снятие замка избавляет от рутины. Используйте блокировки осознанно, разбивайте крупные задачи и следите за циклами — тогда зависимости станут инструментом ускорения, а не бюрократии.

Источник: https://cms.dolinais.ru/zamok-na-kartochke-v-task-kak-blokirovki-zaschischayut-sroki