Связи задач в /task: как читать граф блокировок и не ломать план

27.11.2026 · PDF-версия статьи с сайта cms.dolinais.ru

В проекте почти всегда есть задачи, которые нельзя начинать или закрывать раньше других. Если эти связи держать в голове или в переписке, они быстро теряются: кто-то берёт в работу задачу, которой ещё нет топлива, а кто-то закрывает свою часть, хотя смежная ещё в процессе. В /task такие связи формализованы: задача может **блокировать** другую задачу и **ждать завершения** третьей. В этой статье разберём не отдельные кнопки, а то, как читать получившийся граф зависимостей и принимать по нему решения.

Живое демо /task

Ниже — интерактивная демонстрация /task: можно переключать виды, открывать задачи и пробовать действия со связями и замком. Ничего не сломается — это песочница.

ДЕМО /TASK — КАНБАН: ПРИМЕР ДАННЫХ

Задачи в блоке не заданы.

Что такое граф зависимостей и зачем он нужен

Граф зависимостей — это набор задач, соединённых направленными связями: от блокиратора к зависимой задаче. В /task у каждой задачи две стороны одной связи:

- **«Ждёт завершения» —** список задач, которые должны быть закрыты раньше этой. Это «входы»: без них работа не имеет смысла.
- **«Блокирует» —** список задач, которые не могут двигаться, пока не закрыта текущая. Это «выходы»: то, что вы держите своим результатом.

Обе стороны — отражение одной и той же связи, поэтому не нужно заводить её дважды: добавили блокиратор в «Ждёт завершения» — он автоматически появился в «Блокирует» у другой задачи.

Зачем это нужно на практике:

- **Видно порядок работ.** Планировщик и исполнитель не спорят, что делать первым — это следует из связей.
- **Видно риски.** Если у задачи три блокиратора и все в работе, она не начнётся завтра, как бы этого ни хотелось.
- **Видно простой.** Длинная цепочка из задач без исполнителя сразу подсвечивает узкое место.
- **Видно ответственность.** Блокиратор — это не абстракция, а конкретная задача с исполнителем и сроком.

Как читать связи: три типовых сценария

1. Цепочка «дизайн → вёрстка → тесты»

Классическая последовательность. У вёрстки в «Ждёт завершения» стоит макет, у тестов — вёрстка. Плюс в том, что при сдвиге макета видно, какие задачи поедут следом: достаточно открыть «Блокирует» у задержавшейся задачи. Минус — в жёсткости: если тесты можно частично начать раньше, связь стоит ставить не на всю задачу, а на конкретный результат (например, «тесты API» не ждут вёрстку).

2. Веер «одна задача блокирует пять»

Один блокиратор — например, доступы к среде или согласование договора — держит сразу несколько задач. Это нормально, но именно такие узлы чаще всего становятся причиной простоя. Практика: у задачи-«веера» должен быть явный владелец и срок, а не статус «в работе» на две недели.

3. Схождение «три задачи ждут одну»

Наоборот, несколько задач сходятся к одной. Здесь полезно проверять, что зависимая задача действительно не может стартовать без всех трёх блокираторов: иногда достаточно двух, а третий — просто «сосед по проекту».

> Правило чтения графа: если связь не влияет на решение «можно начинать / можно закрывать», её, скорее всего, не нужно заводить. Лишние связи превращают план в запутанную схему, которую никто не смотрит.

Замок на карточке: что он значит и как его читать

Замок на карточке — визуальный признак того, что задача не может быть переведена в завершённое состояние, пока открыт хотя бы один блокиратор. Это не «запрет работать», а запрет *закрывать*: исполнитель может двигаться по задаче, но финальный переход система не пропустит.

Что важно понимать про замок:

- **Он следствие связей, а не отдельная настройка.** Сняли блокировку — замок исчез сам.
- **Он защищает от ложного «готово».** Задача, закрытая раньше блокиратора, создаёт иллюзию прогресса и ломает следующие по цепочке.
- **Он объясняет причину.** При попытке закрыть задачу видно, какие именно блокираторы ещё открыты — это экономит переписку «почему не закрывается».

Отдельно про сроки: замок не двигает дедлайны и не меняет оценку, он лишь не даёт зафиксировать результат раньше времени. Если блокиратор задерживается, честнее сдвинуть срок зависимой задачи или пересобрать связи, чем обходить замок.

Пошаговый порядок работы

1. **Сформулируйте результат задачи.** Связь имеет смысл, когда понятно, какой именно артефакт или решение нужен от блокиратора.
2. **Добавьте блокиратор в «Ждёт завершения».** Одна связь — одна причина. Не смешивайте «ждёт макет» и «ждёт согласования» в одну строку.
3. **Проверьте обратную сторону.** Откройте «Блокирует» у блокиратора и убедитесь, что там ровно те задачи, которые вы ожидаете.
4. **Назначьте владельцев блокираторов.** Задача без исполнителя в начале цепочки — главный

источник простоя.

5. **Следите за узлами.** Раз в неделю просматривайте задачи с наибольшим числом исходящих связей — это ваши точки риска.
6. **Закрывайте в порядке графа.** Сначала блокиратор, затем зависимая задача; замок сам подскажет, что ещё открыто.

Частые ошибки и как их избежать

- **Связь «на всякий случай».** Лечится вопросом: «Если блокиратор закроется завтра, что изменится в этой задаче?» Нет ответа — связь лишняя.
- **Циклы.** A ждёт B, B ждёт A — обе задачи навсегда под замком. При добавлении связи проверяйте, что не создаёте петлю.
- **Блокиратор-«эпик».** Связь с большой задачей, которая закроется нескоро, держит замок месяцами. Дробите блокиратор на конкретные результаты.
- **Обход замка «через силу».** Если очень нужно закрыть задачу раньше — пересмотрите связи и зафиксируйте решение, а не снимайте ограничение молча.

Что в итоге даёт команде граф зависимостей

Прозрачный порядок работ вместо устных договорённостей, раннее обнаружение узких мест, честные статусы и меньше переписок в духе «а мы можем закрыть?». Связи и замок не усложняют процесс — они делают видимым то, что и так существует в проекте. Начните с одной реальной цепочки: добавьте блокиратор, проверьте обе стороны списка и пройдите путь до закрытия. Дальше это станет привычкой, а план перестанет рассыпаться при первом же сдвиге сроков.

Источник: https://cms.dolinais.ru/svyazi-zadach-v-task-kak-chitat-graf-blokirovok-i-ne-lomat-plan