

Роли в ИТ-компании: кто за что отвечает и как команда взаимодействует

Интернет · 22.10.2026 · PDF-версия статьи с сайта cms.dolnais.ru

Разбираем роли в продуктовой ИТ-компании: от продакт-менеджера и аналитика до разработчиков, QA, DevOps и поддержки. Показываем зоны ответственности, точки взаимодействия и типичные ошибки, из-за которых команда теряет скорость.

ИТ-компания похожа на сложный организм: продукт придумывают, проектируют, разрабатывают, тестируют, выпускают и поддерживают разные люди. Если роли не разделены, команда теряет скорость: одни задачи дублируются, другие висят, а ответственность за результат размывается.

Ниже — карта ролей в типичной продуктовой ИТ-компании: кто за что отвечает, какие решения принимает и как взаимодействует с остальными. В небольшой команде роли часто совмещают, но зоны ответственности всё равно стоит фиксировать.

Зачем различать роли

Роль — это не должность и не набор задач, а зона ответственности. Один человек может быть и аналитиком, и тестировщиком, и даже продакт-менеджером, особенно на ранней стадии. Проблема начинается, когда у каждой зоны нет явного владельца: «все делают продукт», но никто не отвечает за результат.

Разделение ролей помогает:

- быстрее принимать решения и не возвращаться к одному и тому же вопросу;
- понимать, кто отвечает за требования, качество, сроки и релиз;
- снижать количество переделок и «сюрпризов» на поздних этапах;
- строить понятные карьерные треки и развивать специалистов.

Продуктовый контур

Product Manager

Отвечает за: ценность продукта, его стратегию и результат для бизнеса.

- изучает рынок, пользователей и конкурентов;
- формирует видение продукта и дорожную карту;
- определяет метрики успеха и приоритеты;
- принимает решения о том, что делать, а что — нет.

Product Manager отвечает на вопрос «зачем и что делаем». Он не обязан быть самым сильным в технологиях, но должен понимать ограничения команды и экономику продукта.

Product Owner

Отвечает за: содержание и приоритет бэклога, понятность требований для команды и приёмку результата.

- декомпозирует продуктовые цели до задач и пользовательских сценариев;
- расставляет приоритеты в бэклоге;
- отвечает на вопросы команды по функциональности;
- принимает или отклоняет готовую работу по критериям приёмки.

В Scrum Product Owner — отдельная роль. В реальных командах его функции иногда выполняет Product Manager, но важно, чтобы приоритеты не менялись хаотично и у команды был один источник правды.

Бизнес-аналитик

Отвечает за: перевод бизнес-потребностей в понятные требования.

- описывает бизнес-процессы и пользовательские сценарии;
- формулирует функциональные требования и критерии приёмки;
- выявляет ограничения, риски и зависимости;
- согласовывает требования с заказчиком и командой.

Системный аналитик

Отвечает за: техническую детализацию требований и стыковку систем.

- проектирует модели данных и контракты API;
- описывает интеграции и последовательности вызовов;
- прорабатывает нефункциональные требования: производительность, безопасность, отказоустойчивость;
- помогает команде и заказчику говорить на одном техническом языке.

В небольших командах бизнес- и системный аналитик часто один человек. В сложных интеграционных проектах их лучше разделять: бизнес-логика и техническая архитектура требуют разного фокуса.

UX/UI-дизайнер

Отвечает за: пользовательский опыт и визуальную целостность продукта.

- проводит исследования и интервью с пользователями;
- строит сценарии, прототипы и макеты;
- следит за удобством, доступностью и дизайн-системой;
- вместе с аналитиком и разработчиками проверяет реализуемость решений.

Хороший дизайнер не просто «рисует красиво»: он отвечает за то, чтобы пользователь понимал продукт и мог решить свою задачу.

Управление и доставка

Project Manager

Отвечает за: сроки, объём, бюджет, риски и коммуникацию по проекту.

- строит план работ и контролирует его выполнение;
- управляет зависимостями и изменениями объёма;
- снимает организационные блокеры;
- держит в курсе заказчика и стейкхолдеров.

Project Manager отвечает на вопрос «когда и в каких рамках». Он не должен подменять Product Manager в вопросах ценности и приоритетов.

Delivery Manager

Отвечает за: предсказуемость поставки и процесс доставки ценности от идеи до релиза.

- выстраивает поток задач и контролирует узкие места;
- координирует релизы и зависимости между командами;
- следит за метриками delivery: lead time, throughput, стабильность релизов;
- помогает командам договариваться о правилах работы.

Scrum Master / Agile Coach

Отвечает за: процесс, фасилитацию и устранение препятствий, а не за содержание продукта.

- помогает команде соблюдать договорённости и улучшать процесс;
- проводит обучение по Scrum, Kanban и другим подходам;
- фасилитирует ретроспективы и планирование;
- работает с конфликтами и командной динамикой.

Engineering Manager

Отвечает за: людей, команду и инженерную культуру.

- нанимает, развивает и оценивает инженеров;
- отвечает за здоровье команды и уровень выгорания;
- помогает выстраивать инженерные стандарты и процессы;
- работает с Tech Lead над техническим развитием команды.

Технический контур

СТО

Отвечает за: техническую стратегию компании и инженерную функцию в целом.

- определяет технологический вектор и ключевые архитектурные решения;
- отвечает за build vs buy, выбор платформ и технические риски;
- формирует инженерную культуру и стандарты;
- представляет техническую функцию на уровне бизнеса.

В небольшой компании СТО может писать код и лидировать разработку. В крупной — всё больше занимается стратегией, людьми и взаимодействием с бизнесом.

Архитектор

Отвечает за: целостность архитектуры и техническую реализуемость требований.

- проектирует границы систем, компонентов и интеграций;
- принимает решения по стеку, данным и инфраструктуре;
- следит за нефункциональными требованиями;
- фиксирует решения в архитектурных документах и ADR.

Tech Lead / Team Lead

Отвечает за: техническое качество и техническое лидерство внутри команды.

- принимает ключевые технические решения в рамках команды;
- участвует в декомпозиции и оценке задач;
- проводит код-ревью и развивает инженеров;
- следит за качеством, тестируемостью и поддерживаемостью кода.

Tech Lead не обязан быть менеджером по людям. Часто это отдельная роль: один человек отвечает за технологию, другой — за команду и карьеру сотрудников.

Backend-разработчик

Отвечает за: серверную логику, API, данные и интеграции.

- реализует бизнес-логику и контракты API;
- проектирует и оптимизирует работу с базой данных;
- пишет тесты и участвует в код-ревью;
- следит за производительностью, безопасностью и надёжностью сервисов.

Frontend-разработчик

Отвечает за: интерфейс, взаимодействие с пользователем и связку с API.

- реализует макеты и компоненты интерфейса;
- отвечает за производительность и доступность UI;
- работает с состоянием приложения и ошибками;
- взаимодействует с дизайнером и backend-разработчиком.

Мобильный разработчик

Отвечает за: нативные или кроссплатформенные приложения.

- реализует экраны, навигацию и локальное хранение данных;
- интегрируется с API и push-уведомлениями;

- учитывает особенности платформ и требования магазинов приложений;
- следит за производительностью и стабильностью приложения.

Fullstack-разработчик

Отвечает за: сквозную реализацию функций — от интерфейса до серверной части.

Особенно полезен в небольших командах и на ранних стадиях продукта. Важно, чтобы широта не превращалась в поверхностность: сложные части всё равно должны проходить ревью профильных специалистов.

QA-инженер

Отвечает за: качество продукта и проверку того, что он соответствует требованиям.

- строит стратегию тестирования и тест-планы;
- проводит ручное и автоматизированное тестирование;
- проверяет регрессии и критические сценарии перед релизом;
- участвует в приёмке и помогает команде предотвращать дефекты.

QA не «отвечает за качество в одиночку»: качество — общая ответственность команды, а QA задаёт систему проверок и помогает её поддерживать.

DevOps-инженер

Отвечает за: инфраструктуру, CI/CD и воспроизводимость окружений.

- автоматизирует сборку, тестирование и доставку;
- управляет инфраструктурой как кодом;
- настраивает окружения и секреты;
- помогает команде быстрее и безопаснее выпускать изменения.

SRE-инженер

Отвечает за: надёжность и наблюдаемость production-систем.

- определяет SLO/SLI и следит за ними;
- участвует в инцидентах и постмортемах;
- занимается ёмкостью, отказоустойчивостью и деградацией;
- уменьшает операционную нагрузку через автоматизацию.

Инженер по безопасности

Отвечает за: безопасность продукта и процессов разработки.

- проводит threat modeling и аудит кода;
- внедряет безопасные практики в SDLC;
- отвечает за управление уязвимостями и реакцию на инциденты;

- консультирует команды по аутентификации, данным и доступам.

Data-инженер и аналитик данных

Отвечают за: данные, отчётность и качество аналитики.

- Data-инженер строит пайплайны, хранилища и витрины данных;
- аналитик данных формулирует метрики, проверяет гипотезы и готовит отчёты;
- вместе они обеспечивают команду достоверными данными для решений.

ML-инженер

Отвечает за: внедрение моделей машинного обучения в продукт.

- готовит данные и признаки;
- обучает, оценивает и оптимизирует модели;
- внедряет модели в production и следит за деградацией;
- работает с Product Manager над метриками качества.

Технический писатель

Отвечает за: документацию, которой действительно пользуются.

- описывает API, пользовательские инструкции и внутренние процессы;
- поддерживает базу знаний и release notes;
- помогает команде формулировать понятные тексты в интерфейсе;
- снижает нагрузку на поддержку и разработчиков.

Поддержка и Customer Success

Отвечают за: помощь пользователям и обратную связь.

- обрабатывают обращения и инциденты;
- помогают пользователям освоить продукт;
- собирают повторяющиеся проблемы и передают их в продукт;
- участвуют в приоритизации исправлений и улучшений.

Тимлид, сеньор и другие: роли и уровни

В ИТ часто смешивают роль и грейд. «Сеньор» — это уровень компетенции, а «тимлид» — роль в команде. Один и тот же человек может быть Senior Backend Developer и Tech Lead, а может быть сильным Senior без управления людьми.

Грейды: Junior, Middle, Senior

- **Junior** — выполняет понятные задачи по чек-листу, нуждается в ревью и наставничестве. Отвечает за качество своей работы, но пока не за решения всей команды.
- **Middle** — самостоятельно решает типовые задачи, понимает контекст продукта, может предложить

решение и оценить риски. Поддержка нужна в сложных архитектурных вопросах.

- **Senior** — берёт сложные задачи, видит последствия решений, умеет декомпозировать работу, проводить ревью и обучать коллег. Отвечает за техническое качество в своей зоне.

Грейд — это про масштаб задач и степень самостоятельности, а не про количество лет. В разных компаниях границы отличаются, поэтому на собеседовании стоит уточнять, что именно означает Senior или Middle в конкретной команде.

Тимлид и техлид

Тимлид (Team Lead) — роль лидера команды. Он отвечает за людей, процесс и результат: помогает с приоритетами, снимает блокеры, проводит встречи, развивает сотрудников и следит за атмосферой. Тимлид не обязан быть самым сильным программистом, но должен понимать работу команды.

Техлид (Tech Lead) — роль технического лидера. Он отвечает за технические решения, архитектуру в рамках команды, качество кода, ревью и техническое развитие. Часто техлид не управляет людьми напрямую.

В небольшой команде роли тимлида и техлида может совмещать один человек. В крупной их лучше разделять: люди и технология требуют разного внимания.

Кто ещё бывает

- **Lead** — общий уровень ведущего специалиста; может быть Lead Developer, Lead QA или Team Lead. Уточняйте, что именно входит в роль.
- **Staff / Principal** — высокие технические грейды без обязательного управления людьми. Отвечают за сложные технические направления, стандарты и влияние на несколько команд.
- **Engineering Manager** — управляет людьми и инженерной командой: найм, развитие, оценка, здоровье команды. Может не быть самым сильным в технологиях.
- **Head of Engineering / Director of Engineering** — отвечает за несколько команд или инженерную функцию целиком: стратегия, процессы, бюджет, руководители.
- **Архитектор** — техническая роль, а не грейд: отвечает за целостность архитектуры и ключевые технические решения.

Как это связано с ролями из статьи

Роль отвечает на вопрос «что человек делает», грейд — «насколько сложные задачи он решает и насколько самостоятелен». Поэтому формулировки могут звучать так:

- Junior Backend Developer — разработчик начального уровня;
- Senior Frontend Developer — опытный разработчик без управления командой;
- Senior Team Lead — опытный специалист, который также ведёт команду;
- Staff Engineer — сильный инженер, влияющий на несколько команд;
- Engineering Manager — руководитель инженерной команды.

При найме и построении карьерного трека полезно разделять эти две оси. Иначе возникают споры: «почему новый сотрудник сеньор, но не тимлид?» или «почему тимлид не решает архитектурные вопросы?». Ответ прост: это разные измерения ответственности.

Как роли взаимодействуют между собой

Роли не работают изолированно. Основные точки взаимодействия удобно рассматривать по этапам жизненного цикла продукта.

1. Discovery: понять задачу

Product Manager, бизнес-аналитик, дизайнер и Tech Lead вместе проверяют проблему, пользователей и ограничения. На выходе — понимание ценности, критериев успеха и предварительная оценка реализуемости.

2. Проектирование: превратить идею в решение

Аналитик и дизайнер прорабатывают сценарии, системный аналитик и архитектор — интеграции и данные, Tech Lead и разработчики — техническую реализацию. Важно не переходить к разработке, пока не ясны критерии приёмки и границы задачи.

3. Планирование: договориться о работе

Product Owner и Tech Lead декомпозируют задачи, QA помогает определить стратегию проверки, DevOps — учесть окружения и релизные ограничения. Команда оценивает объём и фиксирует Definition of Ready.

4. Разработка: сделать и проверить

Разработчики реализуют задачи, дизайнер проверяет интерфейс, QA — сценарии и регрессии, аналитик отвечает на вопросы по требованиям. Ежедневные синки помогают быстро выявлять блокеры, а код-ревью — удерживать качество.

5. Релиз: вывести в production

DevOps и SRE обеспечивают доставку и наблюдаемость, QA — финальную проверку, Product Owner — приёмку, Project Manager или Delivery Manager — коммуникацию и координацию. После релиза важно проверить метрики и собрать обратную связь.

6. Поддержка: учиться на результате

Поддержка и Customer Success собирают обращения, аналитик данных проверяет метрики, Product Manager превращает выводы в новые гипотезы. Так замыкается цикл: продукт развивается на основе реального использования, а не только первоначального плана.

RACI и договорённости

Чтобы взаимодействие не превращалось в бесконечные согласования, полезно явно фиксировать ответственность. Один из простых инструментов — RACI:

- **Responsible** — кто выполняет работу;
- **Accountable** — кто принимает финальное решение и отвечает за результат;

- **Consulted** — кого нужно спросить до решения;
- **Informed** — кого нужно уведомить после.

Достаточно договориться о RACI для ключевых процессов: приоритизация бэклога, приёмка задачи, релиз, инцидент, изменение архитектуры. Это снимает споры «кто здесь главный» лучше, чем любая оргсхема.

Типовые составы команд

Небольшой стартап

Product Manager или основатель, дизайнер, 2–4 разработчика, QA и DevOps по совместительству. Роли часто совмещают, но важно сохранять явного владельца продукта и технического решения.

Продуктовая команда

Product Manager, Product Owner или аналитик, дизайнер, Tech Lead, 5–7 разработчиков, QA, DevOps. Такой состав уже позволяет вести несколько независимых потоков задач и выпускать релизы предсказуемо.

Крупная компания

Несколько продуктовых команд, архитектурный комитет, платформенная команда, SRE, безопасность, данные и поддержка. Здесь особенно важны общие стандарты, прозрачные интерфейсы между командами и единые правила принятия решений.

Частые ошибки

- **Смешивать Product Manager и Project Manager.** Первый отвечает за ценность и «что делать», второй — за сроки и «как успеть».
- **Считать QA единственным ответственным за качество.** Качество начинается с требований, дизайна и кода, а не с финального тестирования.
- **Оставлять аналитика «переводчиком» без права влиять на решение.** Тогда требования превращаются в пересказ, а не в проработанное решение.
- **Делать Tech Lead одновременно архитектором, тимлидом и менеджером по людям.** В небольшой команде это возможно, но зоны внимания начнут конфликтовать.
- **Не фиксировать решения.** Без ADR, критериев приёмки и договорённостей команда снова и снова обсуждает одно и то же.
- **Забывать про поддержку и данные.** Без обратной связи от пользователей и аналитики продукт развивается вслепую.

Короткий чек-лист

1. Выпишите роли и зоны ответственности для своей команды.
2. Определите, кто принимает решения по продукту, технологии и релизу.
3. Зафиксируйте Definition of Ready и Definition of Done.
4. Договоритесь о ритуалах: планирование, синки, демо, ретро, постмортемы.
5. Ведите единый бэклог и документируйте архитектурные решения.
6. Регулярно проверяйте, не размылись ли роли при росте команды.

Сильная ИТ-команда — это не команда из «универсальных солдат», а система, где у каждого есть понятная зона ответственности и возможность влиять на общий результат. Чем яснее роли и договорённости, тем меньше энергии уходит на координацию и тем больше — на создание продукта.

Источник: <https://cms.dolinais.ru/account/articles/1/95>