**Федеральное государственное бюджетное образовательное учреждение высшего образования   
«Саратовский государственный технический университет имени Ю. А. Гагарина»**

**Кафедра прикладных информационных технологий**

**Отчёт по практической работе «Инструментальные средства управления задачами и багтрекинга».**

Выполнил студент б1-ИФСТ-31,

Песчанов Арсений Андреевич

Проверил

Старший преподаватель

кафедры «ПИТ»

Пиминов Дмитрий Алексеевич

Саратов, 2023

**Анализ существующих инструментальных средств управления задачами.**

1. **Asana:**

* Asana - это веб-приложение, которое основано на облачной архитектуре. Оно разработано на языке программирования Python и использует базу данных PostgreSQL.
* Функциональные возможности Asana включают создание проектов и задач, назначение ответственных за выполнение задач, установку сроков выполнения задач, отслеживание прогресса выполнения задач, обмен сообщениями и файлами внутри проекта, создание шаблонов проектов для повторного использования и многое другое (Рисунок 1).
* Ограничения использования Asana зависят от выбранного тарифного плана. Например, бесплатный тарифный план имеет ограничение на количество пользователей в команде до 15 человек, а также ограничения на доступ к расширенным функциям. Платные тарифы могут иметь другие ограничения, такие как ограничение на количество проектов или задач.
* Условия использования Asana определяются пользовательским соглашением и политикой конфиденциальности, которые можно найти на официальном сайте Asana.
* Asana также интегрируется с другими сервисами, такими как Google Drive, Dropbox, Slack, Zapier и многими другими.
* Asana предоставляет бесплатный тарифный план для небольших команд до 15 человек.

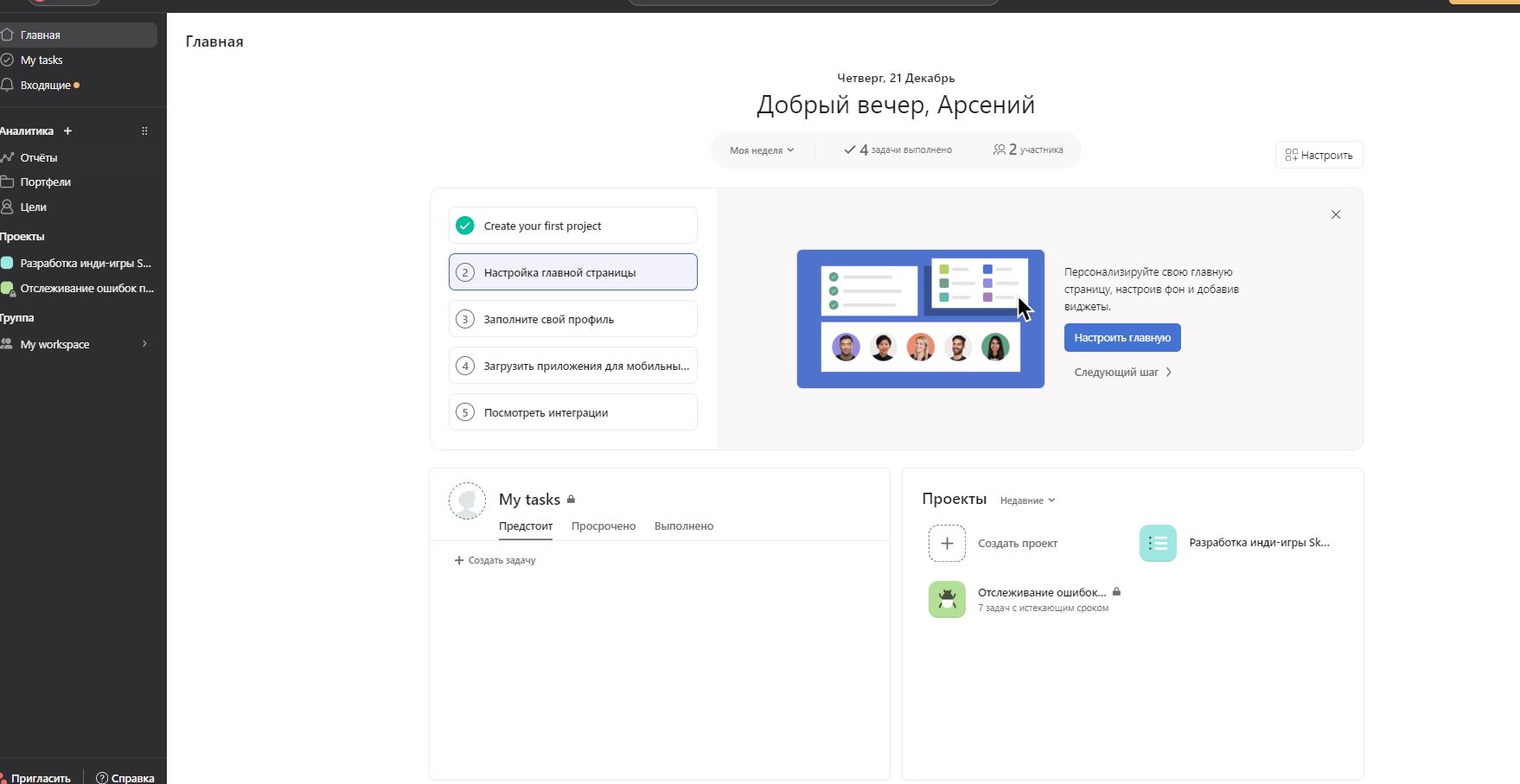


Рисунок 1. Интерфейс Asana

1. **Jira:**

* Jira - это инструментальное средство для управления проектами и задачами, которое разработано на языке программирования Java и использует базу данных MySQL.
* Функциональные возможности Jira включают создание проектов и задач, назначение ответственных за выполнение задач, установку сроков выполнения задач, отслеживание прогресса выполнения задач, создание диаграмм Ганта и отчетов о проекте, интеграцию с другими инструментами разработки программного обеспечения и многое другое (Рисунок 2).
* Ограничения использования Jira зависят от выбранного тарифного плана. Например, бесплатный тарифный план имеет ограничение на количество пользователей в команде (до 10 человек), а также ограничения на доступ к расширенным функциям. Платные тарифы могут иметь другие ограничения, такие как ограничение на количество проектов или задач.
* С помощью Jira вы можете создавать проекты и задачи, назначать ответственных за выполнение задач, устанавливать сроки выполнения задач, отслеживать прогресс выполнения задач, создавать диаграммы Ганта и отчеты о проекте, интегрировать с другими инструментами разработки программного обеспечения и многое другое.
* Jira имеет различные тарифные планы, которые могут иметь ограничения на количество пользователей, доступ к расширенным функциям и другие параметры. Выбор тарифного плана зависит от потребностей вашей команды и бюджета.
* Одним из основных преимуществ использования Jira является улучшение коммуникации и сотрудничества в команде. Каждый член команды может видеть свои задачи, сроки и прогресс выполнения, что позволяет более эффективно управлять проектами и достигать поставленных целей.
* Также Jira предоставляет возможность создания настраиваемых досок задач, которые можно использовать для отслеживания задач в реальном времени и быстрого реагирования на изменения в проекте.

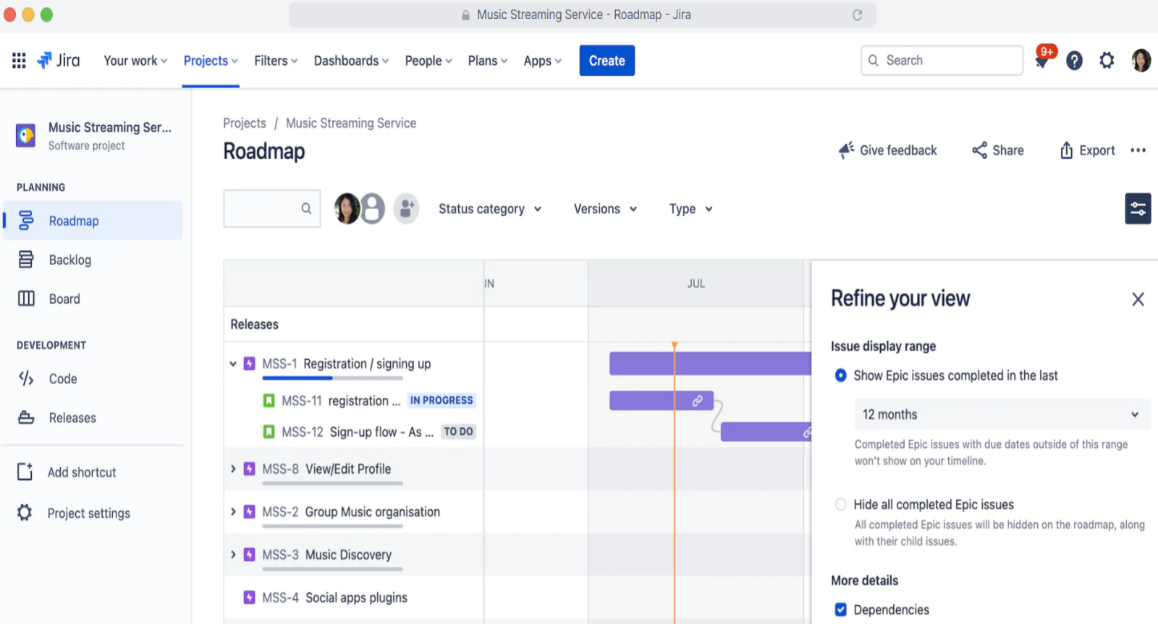


Рисунок 2. Интерфейс Jira

1. **Wrike:**

* Wrike - это инструментальное средство для управления проектами и задачами, которое также используется в различных отраслях, включая IT, финансы, производство и другие. Он позволяет командам эффективно организовывать и отслеживать задачи, сроки и прогресс выполнения проектов.
* Архитектура Wrike основана на облачных технологиях, что позволяет пользователям получить доступ к своим проектам и задачам из любого места и на любом устройстве. Wrike имеет простой и интуитивно понятный интерфейс, что делает его доступным для использования как для опытных пользователей, так и для новичков.
* Функциональные возможности Wrike включают создание проектов и задач, назначение ответственных за выполнение задач, установку сроков выполнения задач, отслеживание прогресса выполнения задач, создание диаграмм Ганта и отчетов о проекте, интеграцию с другими инструментами разработки программного обеспечения и многое другое.
* Ограничения использования Wrike зависят от тарифного плана, который вы выбираете. Некоторые функции могут быть недоступны в более дешевых тарифных планах, а некоторые тарифные планы могут иметь ограничения на количество пользователей или количество проектов.
  + - Условия использования Wrike также зависят от тарифного плана, который вы выбираете. Некоторые тарифные планы могут предоставлять более высокий уровень поддержки и возможности интеграции с другими инструментами, а некоторые тарифные планы могут иметь ограничения на количество пользователей или количество проектов.

1. **Вывод:**

В ходе проведённого анализа было выбрано инструментальные средства управления задачами и багтрекинга – Asana. Данное инструментальное средство предоставляет удобный интерфейс, как и остальные средства предоставляет обширный список инструментов. Но в отличии от других средств управления задачами, Asana имеет бесплатный 30 дневной доступ к абсолютно всему инструментарию, что позволяет новичкам объективно оценить и попробовать все возможности сервиса, более того - в системе Asana уже есть система багтрекинга, что освобождает разработчиков от поиска и подключения внешних систем багтрекинга.

**Практическая апробация инструментального средства управления задачами.**

1. Выбрать одну из систем управления задачами. Включить в отчёт чёткое обоснование своего выбора. Произвести необходимые действия по активации выбранной системы (Рисунок 3).

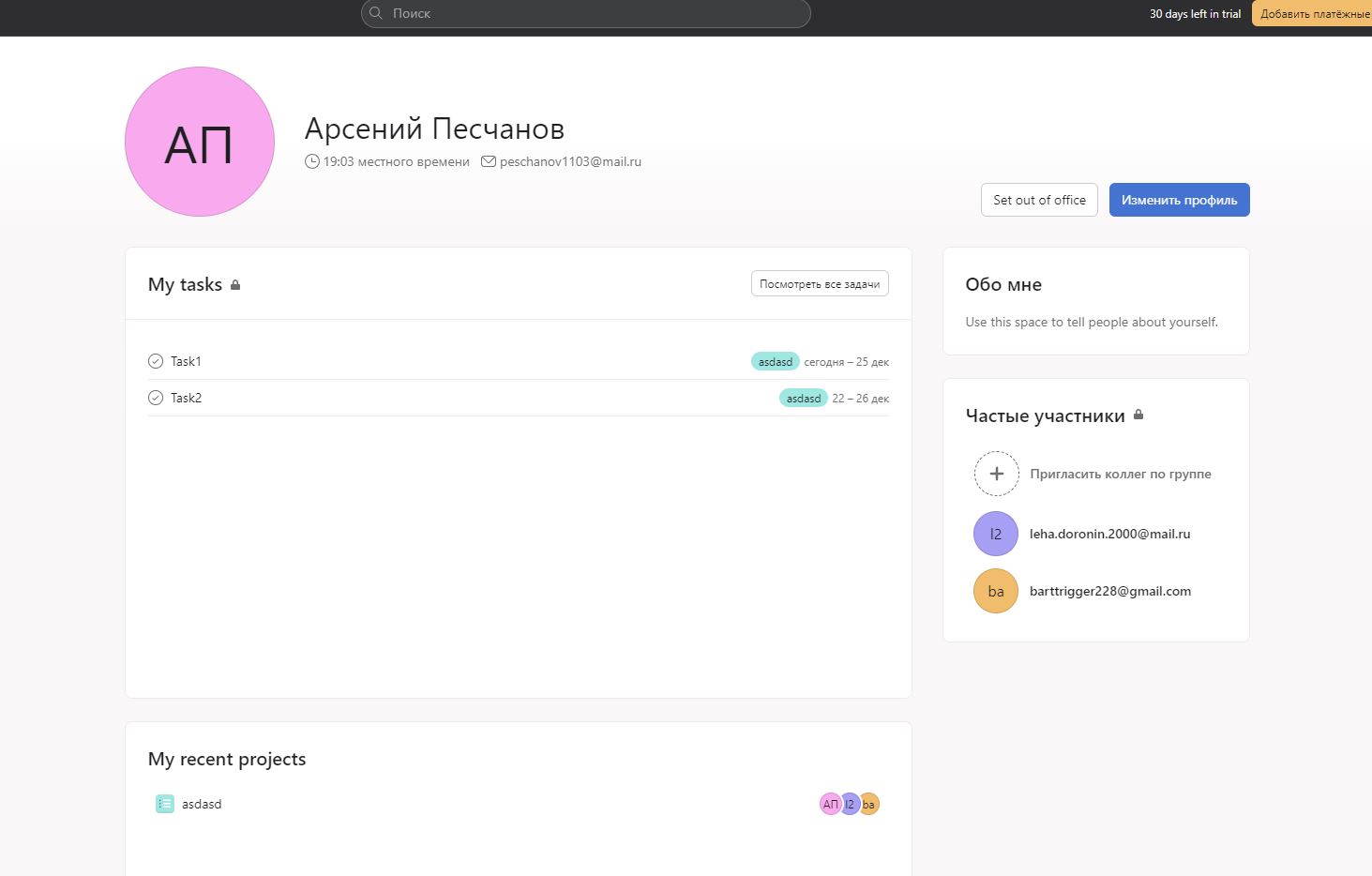


Рисунок 3. Регистрация в Asana

1. Завести проект в выбранной системе управления задачами. Рекомендуется завести не тестовый, а реальный проект, связанный, например, с разработкой программного обеспечения для курсовой или дипломной работы (Рисунок 4).

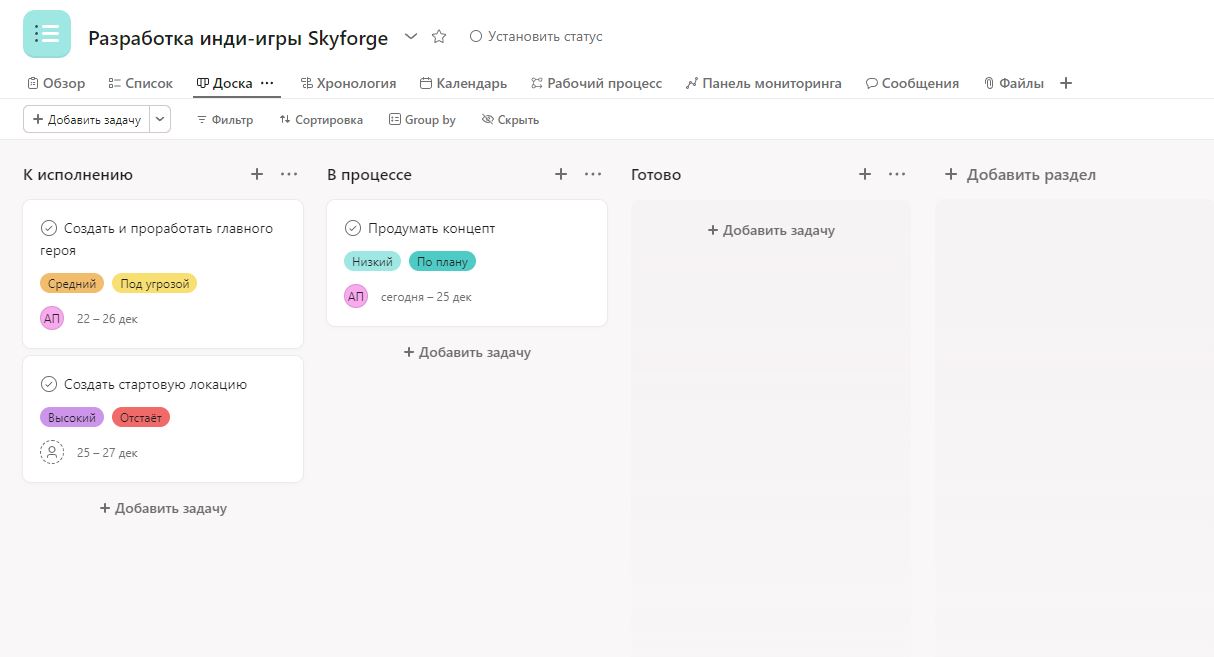


Рисунок 4. Созданный проект

При этом настроить подсистему отслеживания ошибок (багтрекинг). Багтрекинг может выполняться как подсистемой выбранного инструментального средства, так и внешней системой (Рисунок 5).

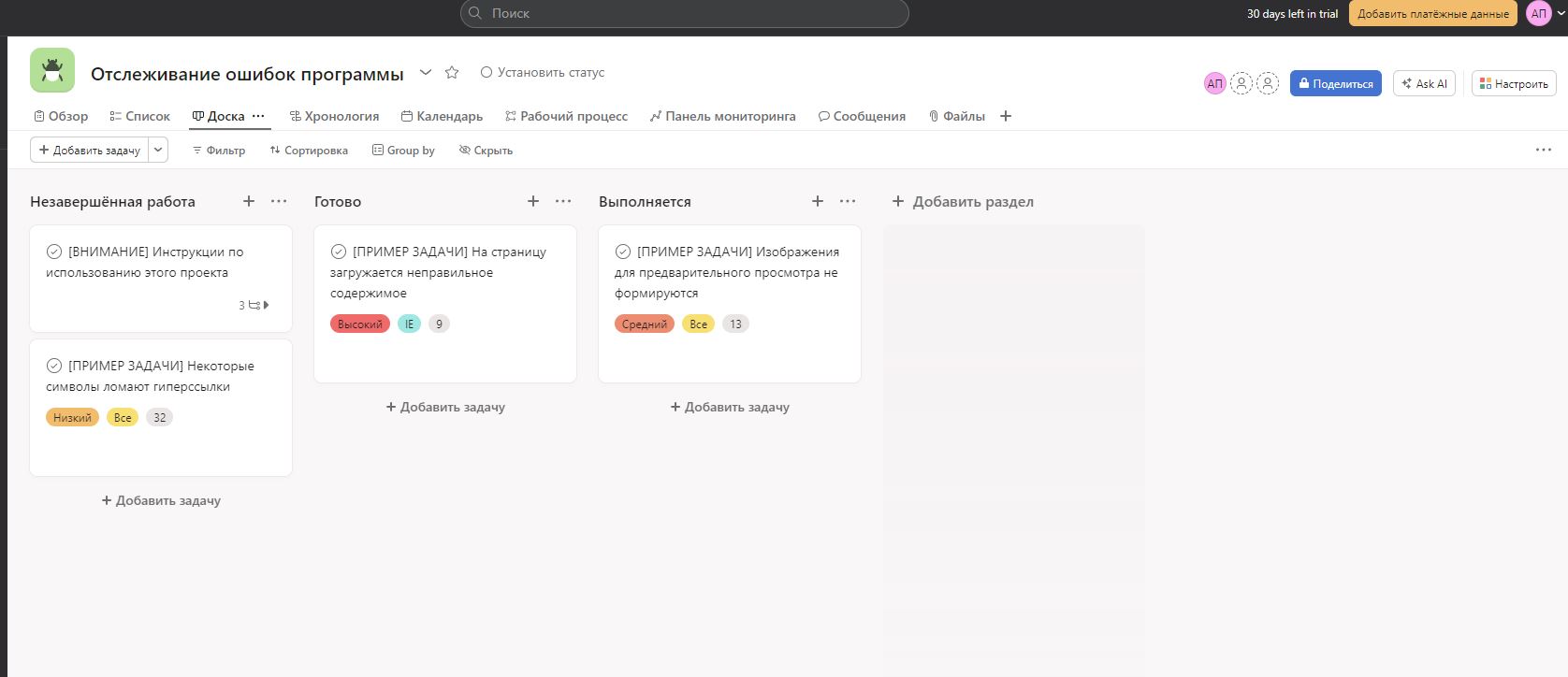


Рисунок 5. Система багтрекинга

1. Сформировать команду разработчиков, состоящую из 3-4 человек. Рекомендуется включить в команду не искусственных тестовых исполнителей, а своих одногруппников (Рисунок 6).

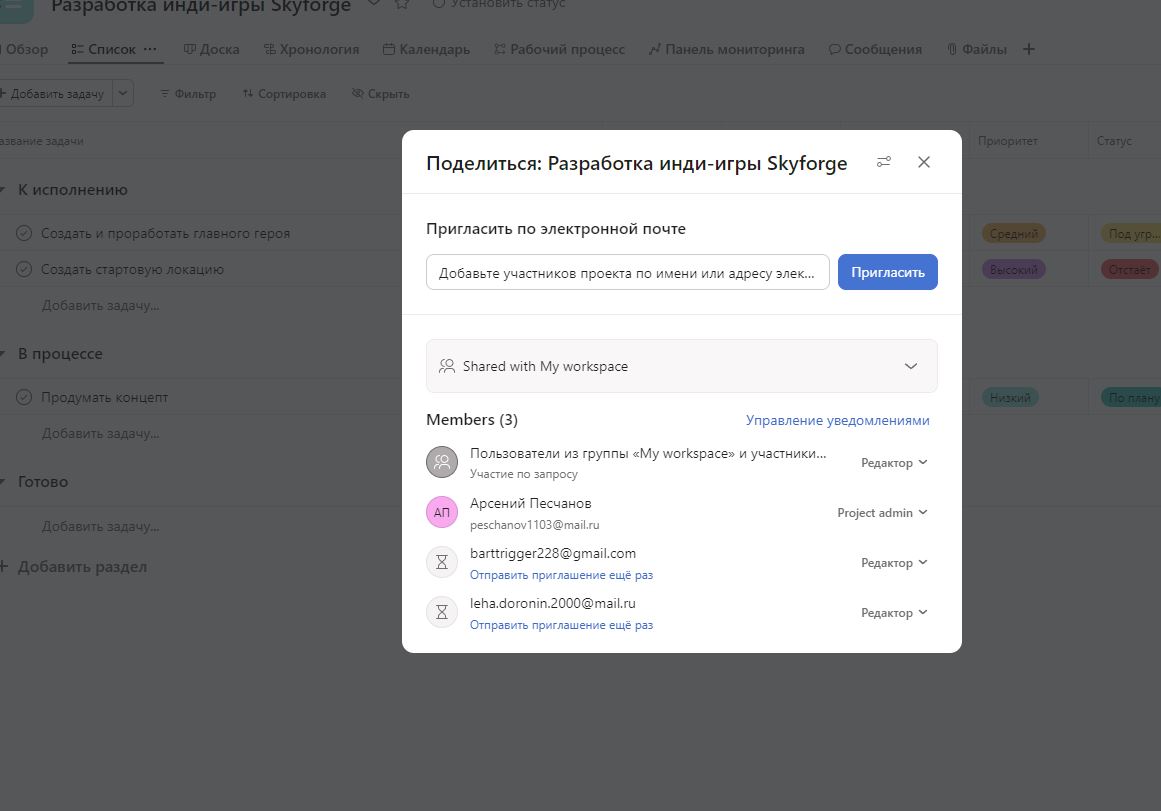


Рисунок 6. Добавление пользователей в проект

1. Разбить весь проект на 2-3 стадии, в каждой из которых сформировать пул из 7-10 задач. При этом для каждой задачи установить сроки выполнения и требуемые ресурсы (Рисунок 7).

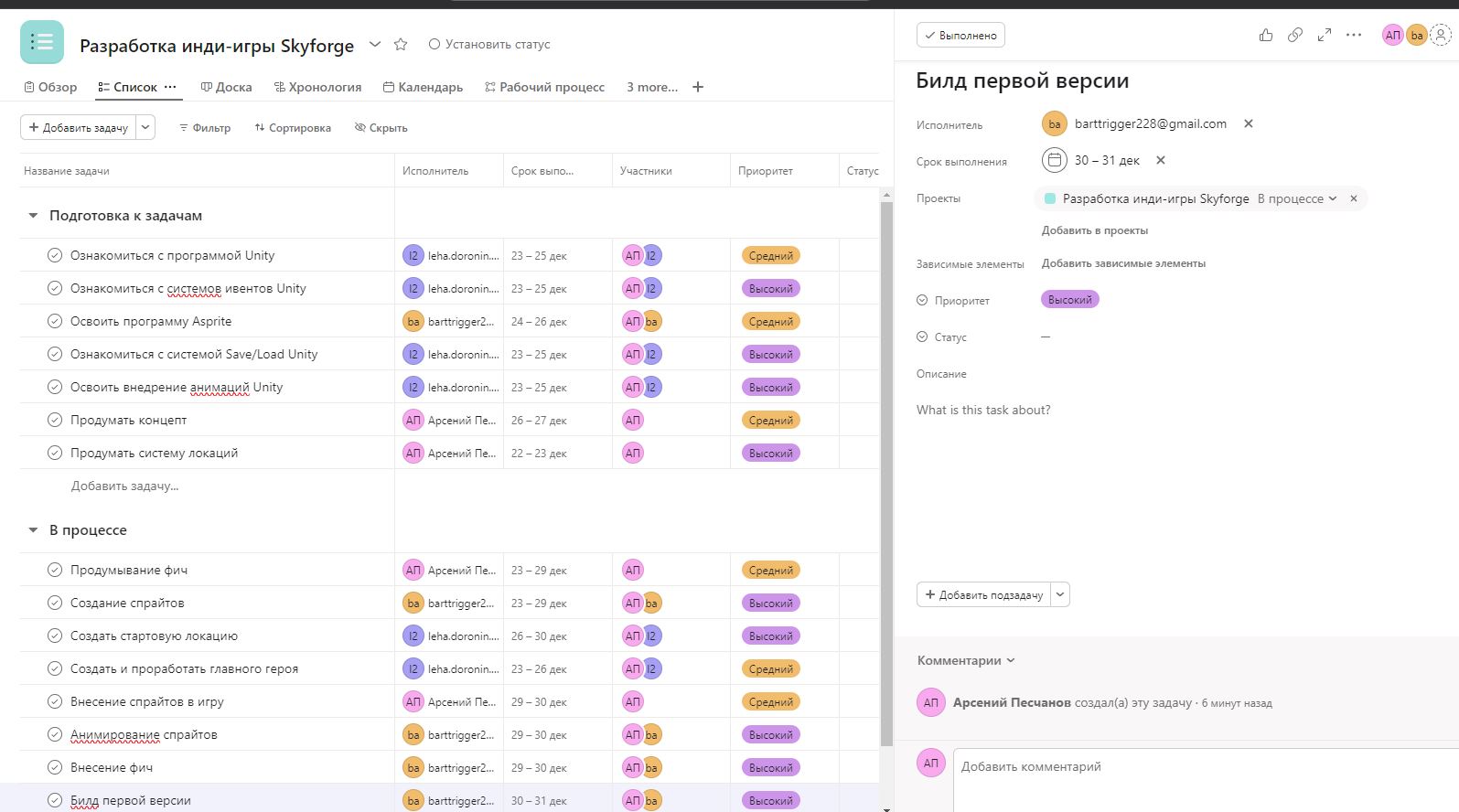


Рисунок 7. Разбиение проекта на стадии и пулы задач

1. Установить взаимосвязи задач и графически отобразить их, например, при помощи диаграммы Ганта(Рисунок 8).

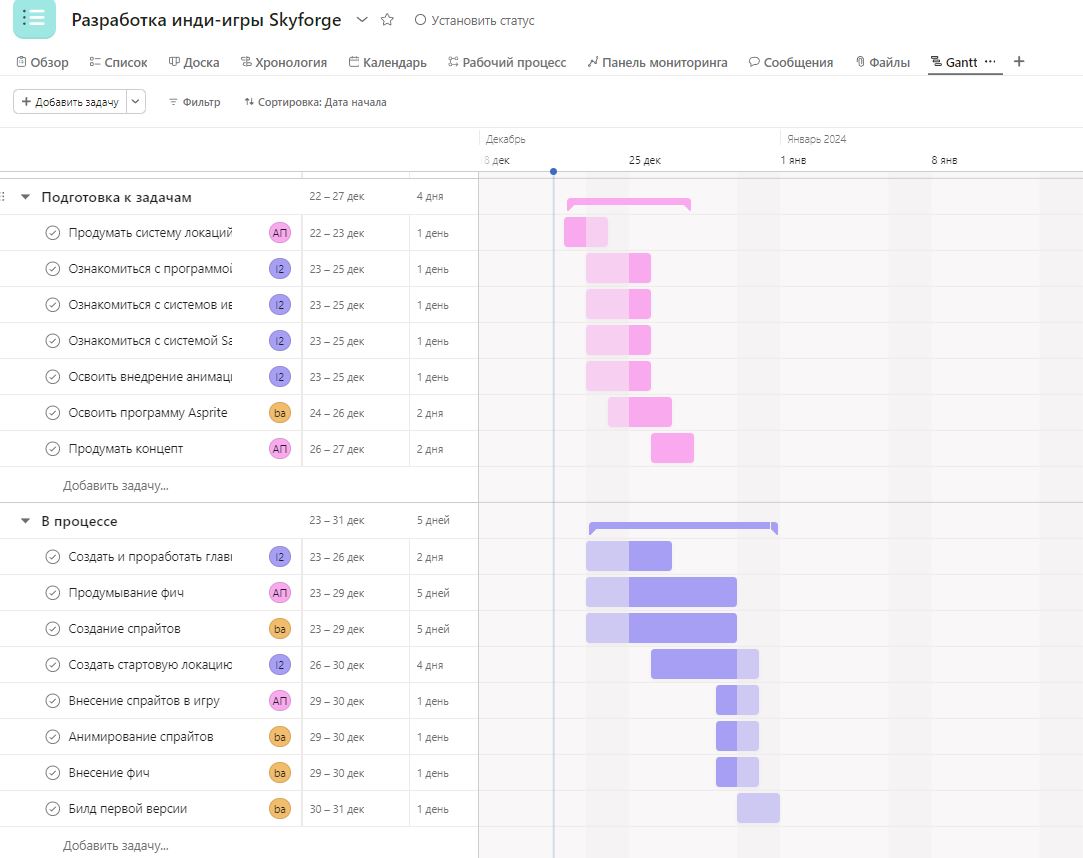


Рисунок 8. Диаграмма Ганта

1. Отобразить «критический путь» проекта (Рисунок 9).

Критический путь – это одна или несколько задач, влияющие на срок завершения проекта. Это кратчайший путь исполнения проекта, т. е. невозможно выполнить проект ранее, чем будут завершены работы по цепочке задач проекта, лежащих на критическом пути.

В основе метода лежит определение наиболее длительной последовательности задач от начала проекта до его окончания с учетом их взаимосвязи. Задачи, лежащие на критическом пути (критические задачи), имеют нулевой резерв времени выполнения, и, в случае изменения их длительности, изменяются сроки всего проекта. В процессе выполнения проекта критический путь проекта может меняться, так как при изменении длительности задач некоторые из них могут оказаться на критическом пути.

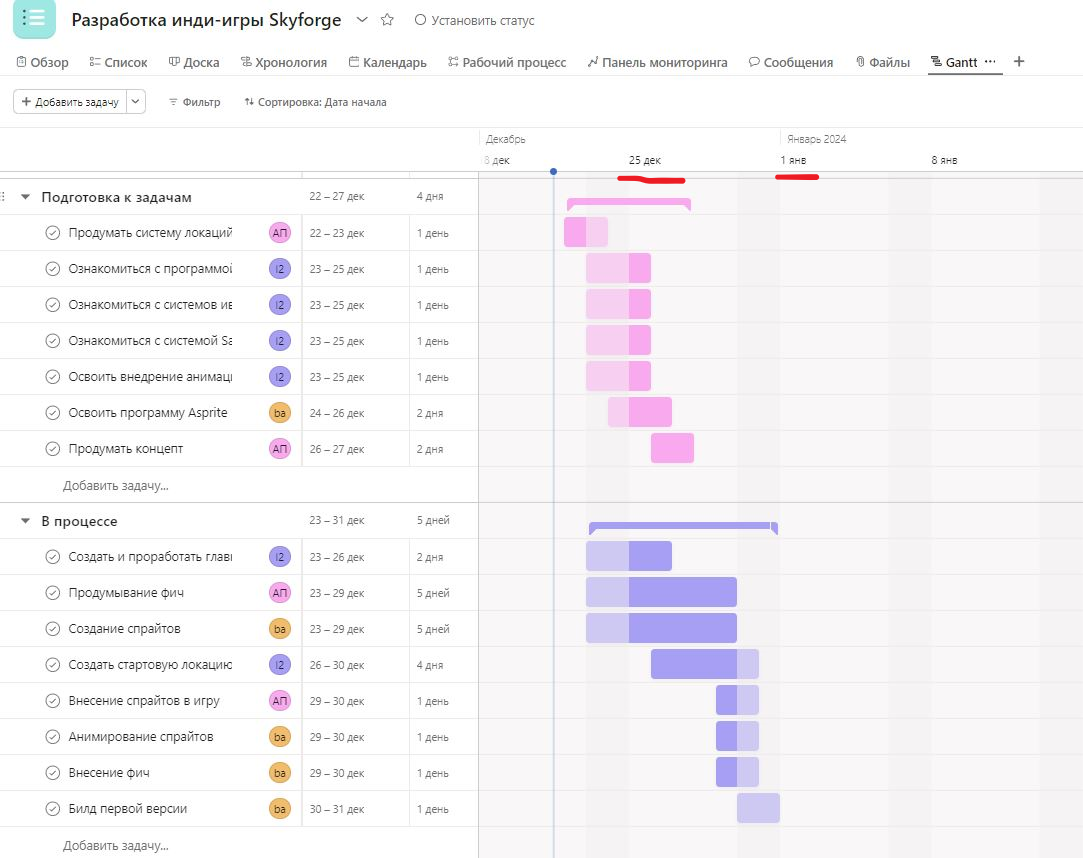


Рисунок 9. Критический путь проекта

1. Произвести назначение задач на конкретных исполнителей из команды, установив задание, сроки, исходные данные (Рисунок 10).

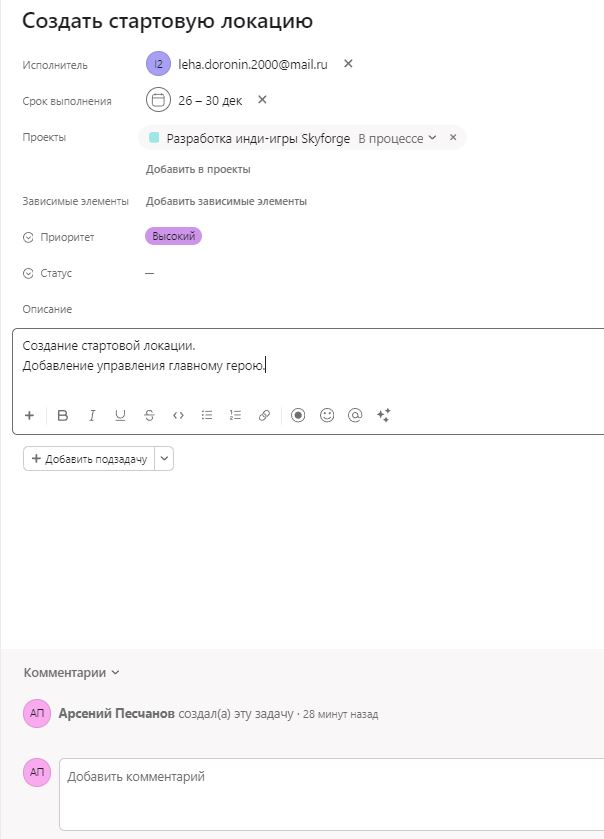


Рисунок 10. Назначение исполнителя на задачу

1. От лица конкретного исполнителя продемонстрировать получение задач, регистрацию фактов получения задач в системе и оповещение об этом менеджера.

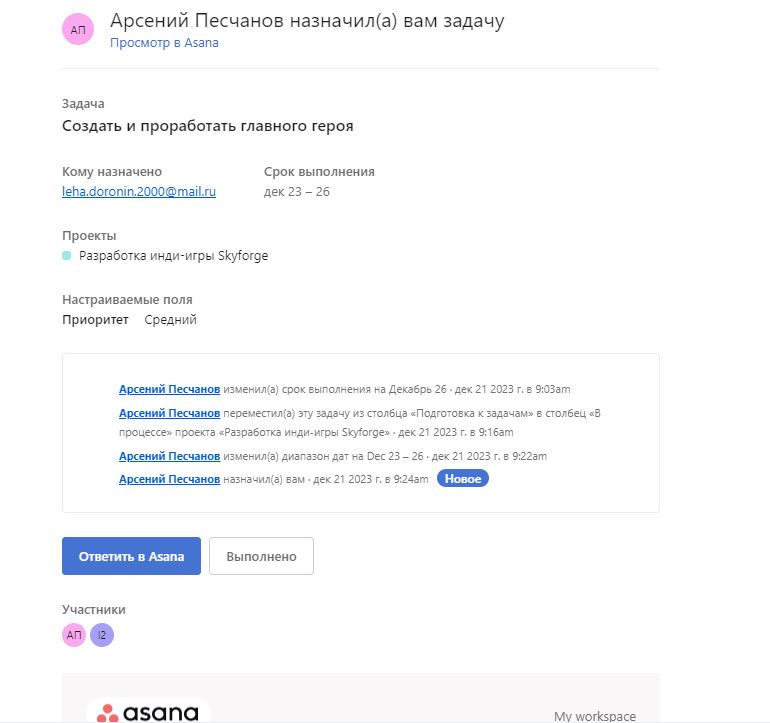


Рисунок 11. Получение задачи и оповещение

1. Промоделировать нахождения ошибки в разрабатываемой системе, её регистрацию и назначение на конкретного разработчика в багтрекинг-системе. Затем проэмулировать весь жизненный цикл деффекта с соответствующей регистрацией в системе (Рисунок 12).

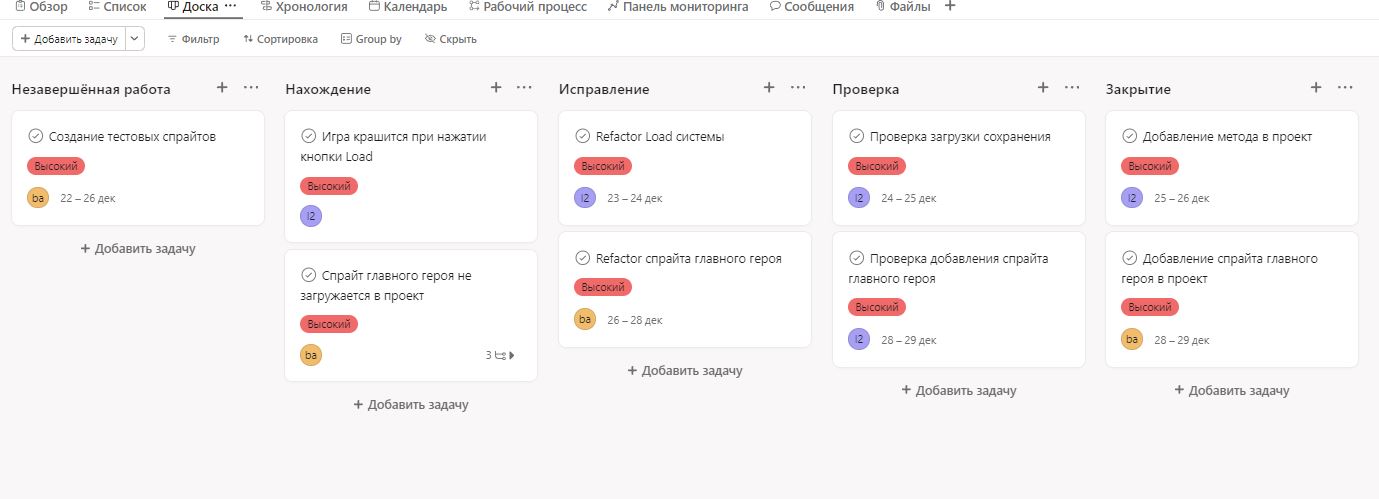


Рисунок 12. Нахождение ошибки и её жизненный цикл

1. Промоделировать ситуацию увеличения сроков выполнения задачи с соответствующей перестройкой плана исполнения проекта. При изменении сроков задач, не находящихся на критическом пути проекта – это может не влиять на сроки сдачи проекта. Однако если задача находится на критическом пути проекта, и её срок сдачи меняется – то и срок сдачи самого проекта может измениться (Рисунок 13).

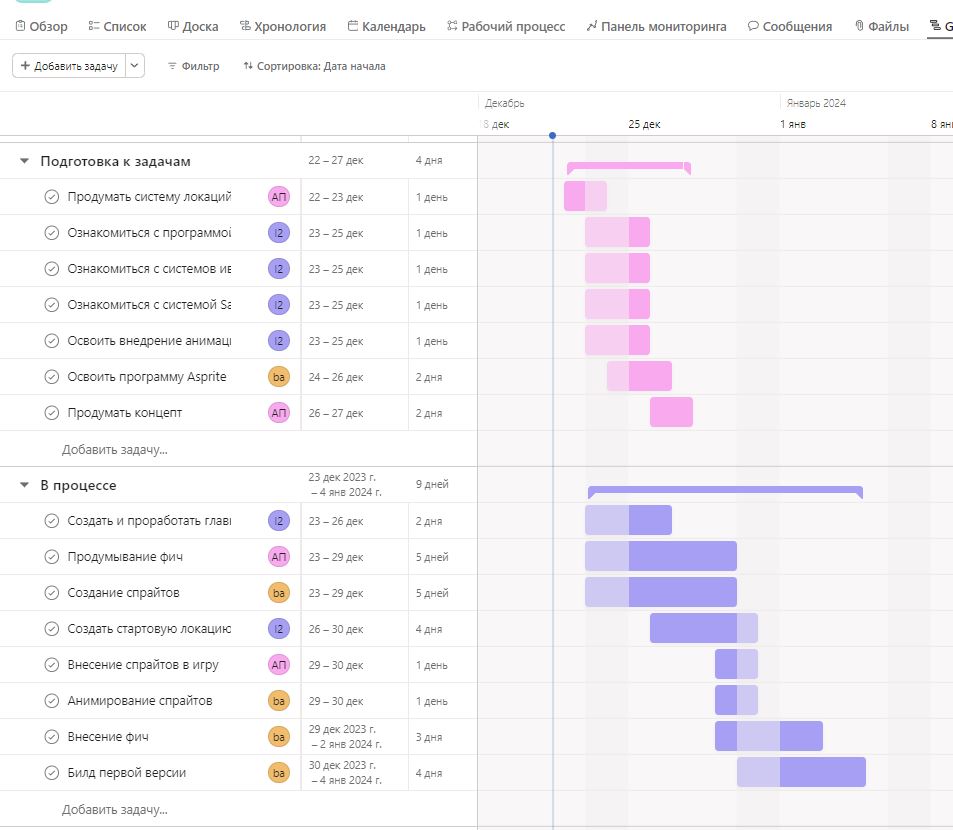


Рисунок 13. Изменение сроков выполнения задач

1. Смоделировать завершение всего проекта. При этом при помощи встроенных возможностей системы построить отчётность (Рисунок 14).

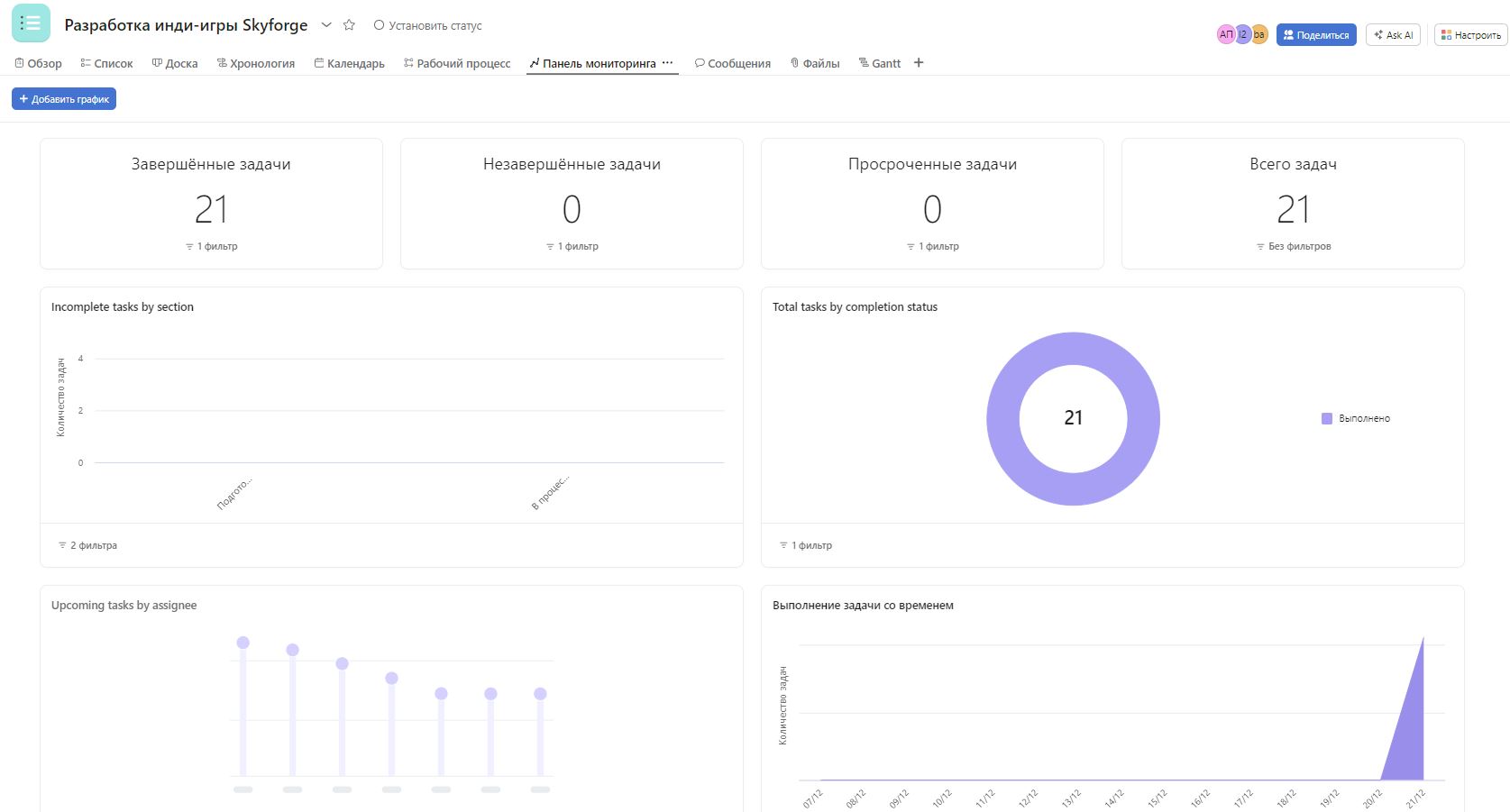


Рисунок 14. Отчетность