МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное
автономное образовательное
учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ПРАКТИЧЕСКИМ РАБОТАМ

«Управление программными проектами»

Направление подготовки 09.03.04 «Программная инженерия»
Направленность (профиль) «Разработка и сопровождение программного обеспечения»
Квалификация выпускника: бакалавр

Ставрополь

2026

Введение

Цель изучения дисциплины «Управление программными проектами» является формирование у студентов теоретических знаний и практических навыков в области управления проектами разработки программного обеспечения, включая планирование, организацию командной работы, контроль ресурсов и сроков, документирование и обеспечение качества, с учетом правовых норм и стандартов в сфере информационных технологий.

Задачами обучения являются:

- изучение методологий управления проектами (традиционные, гибкие, гибридные) и их применимости к различным типам IT-проектов;
- освоение инструментов планирования, оценки трудозатрат, управления рисками и контроля исполнения проектов;
- формирование навыков работы в команде, распределения ролей и эффективной коммуникации с заинтересованными сторонами;
- изучение современных систем контроля версий и инструментов управления задачами (Jira, Trello, GitLab);
- освоение методов документирования проекта, подготовки отчетности и презентации результатов;
- изучение правовых и этических аспектов управления IT-проектами, включая лицензирование, защиту интеллектуальной собственности и информационную безопасность.

1. Перечень планируемых результатов обучения по дисциплин (модулю), соотнесенных с планируемыми результатами освоения образовательной программы

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов

ПК-1 – Способен осуществлять управление программными проектами: планировать, контролировать сроки и ресурсы, применять системы контроля версий и инструменты управления задачами	ИД-1$_{ПК-1}$ Применяет методы декомпозиции работ (WBS), оценки трудоёмкости (PERT, Planning Poker) и календарного планирования (диаграмма Ганта, критический путь) для формирования плана проекта	Способен разрабатывать WBS и диаграмму Ганта, определять критический путь и распределять ресурсы
	ИД-2$_{ПК-1}$ Использует системы контроля версий (Git, Git Flow) и платформы совместной разработки (GitLab, GitHub) для организации командной работы	Способен применять Git для управления исходным кодом и документацией, использовать ветвление и слияние для организации командной работы
	ИД-3$_{ПК-1}$ Применяет инструменты управления задачами (Jira, Trello, YouTrack) для отслеживания прогресса (burndown chart), ведения бэклога и контроля исполнения	Способен использовать инструменты управления задачами (Jira, Trello, GitLab Issues) для постановки задач и контроля сроков выполнения
ПК-9 – Умеет оформлять техническую документацию, готовить презентации и отчеты, вести деловую коммуникацию на русском и иностранном языках в профессиональной среде	ИД-1$_{ПК-9}$ Оформляет техническую документацию (ТЗ, пояснительная записка, руководство пользователя, API-спецификация) в форматах Markdown, OpenAPI, DocBook в соответствии с ГОСТ	Способен разрабатывать и оформлять проектную документацию (устав проекта, ТЗ, план управления проектом) в соответствии со стандартами
	ИД-2$_{ПК-9}$ Готовит презентационные материалы и публично защищает результаты проектной деятельности, аргументированно отвечая на вопросы	Способен создавать презентационные материалы и проводить публичные выступления, защищая результаты проектной деятельности
ПК-10 – Готов соблюдать правовые нормы, стандарты и требования информационной безопасности, участвовать в разработке нормативной и технической документации, учитывать законодательство в области ИТ	ИД-1$_{ПК-10}$ Применяет основные правовые нормы (лицензирование ПО, авторское право, ФЗ-152, ФЗ-149) и стандарты ИБ (ГОСТ Р ИСО/МЭК 27001, ГОСТ Р 56545-2015) при разработке и эксплуатации ПО	Способен применять правовые нормы при выборе лицензий на разрабатываемое ПО, учитывать требования законодательства об авторском праве

Лабораторная работа № 1

Формирование команды и выбор идеи проекта

Цель: Сформировать команду и инициировать учебный проект.

Теоретическое обоснование

Успех любого IT-проекта на 80% определяется качеством команды и чёткостью идеи на старте. **Формирование команды** – это не просто объединение людей, а создание группы с взаимодополняющими компетенциями (аналитик, разработчик, тестировщик, менеджер). В теории управления проектами выделяют несколько стадий развития команды (модель Такмана): формирование (forming), бурление (storming), нормирование (norming), работа (performing) и расформирование (adjourning). На начальном этапе важно распределить роли, определить зоны ответственности и договориться о правилах коммуникации (например, ежедневные стендапы, использование мессенджеров, система контроля версий).

Генерация идей – ключевой процесс инициации. Применяются методы: мозговой штурм (не критиковать, количество важнее качества), метод SCAMPER (замена, комбинирование, адаптация, модификация, применение для другой цели, удаление, перестановка), метод фокальных объектов (перенос свойств случайных объектов на разрабатываемый продукт). Важно не только нагенерировать идеи, но и отсеять нежизнеспособные. Для этого используют критерии: актуальность (есть ли потребность?), реализуемость (техническая и ресурсная), уникальность (чем отличается от аналогов), потенциальная прибыль (даже в учебном проекте оценивают «ценность»). Выбранная идея оформляется как бизнес-идея, содержащая ценностное предложение: «Для кого? Какую проблему решаем? Какая выгода?».

После выбора идеи необходимо создать репозиторий в Git (GitHub, GitLab или Bitbucket). **Git** – распределённая система контроля версий, позволяющая вести историю изменений, работать с ветками, разрешать конфликты. **Файл README.md (формат Markdown)** – это визитная карточка проекта. Он должен

содержать: название, краткое описание (1–2 предложения), цели проекта, стек технологий (языки, фреймворки, базы данных), инструкцию по установке и запуску (даже если позже), список участников и ссылки на дополнительные документы. Хороший README повышает привлекательность проекта для потенциальных вкладчиков и облегчает старт новым участникам. Таким образом, первая лабораторная работа закладывает фундамент: команда + идея + инструментарий.

Методика и порядок выполнения работы

Задание 1. Сформировать команду (3–4 чел.), выбрать вариант (или предложить свой), провести мозговой штурм, утвердить идею, создать репозиторий на GitHub/GitLab, оформить README.md (название, цель, основные функции, технологии, участники).

Вариант	Название проекта	Краткое описание
1	Интернет-магазин книг	Платформа для продажи бумажных и электронных книг
2	Мобильное приложение «Фитнес-трекер»	Учёт тренировок, питания, интеграция со смарт-часами
3	Система управления задачами (аналог Trello)	Доски, списки, карточки, теги, сроки
4	Чат-бот для службы поддержки	Ответы на частые вопросы, эскалация оператору
5	Веб-сервис для онлайн-бронирования столиков в кафе	Календарь занятости, уведомления, отзывы
6	Приложение для изучения иностранных слов (карточки)	Интервальное повторение, статистика, тесты
7	Платформа для обмена студенческими конспектами	Загрузка, рейтинг, комментарии, поиск
8	Система мониторинга доступности веб-сайтов	Проверка HTTP, уведомления при сбоях, отчёты

9	Приложение «Калькулятор расходов семьи»	Категории, графики, бюджеты, мультипользовательский режим
10	Генератор расписания для вуза	Учёт преподавателей, аудиторий, групп, избегание конфликтов

Содержание отчета и его форма

1) Титульный лист с названием ЛР, ФИО участников, вариант.
2) Описание процесса формирования команды (роли, распределение обязанностей).
3) Результаты мозгового штурма (не менее 5 идей, оценка каждой).
4) Обоснование выбранной идеи (SWOT или матрица оценки).
5) Ссылка на репозиторий и скриншот README.md.
6) Выводы о готовности команды к работе.
7) Ответы на контрольные вопросы.

Вопросы для защиты работы

1. Какие роли обычно выделяют в команде разработки ПО?
2. В чём разница между «бизнес-идеей» и «продуктовым видением»?
3. Перечислите не менее трёх критериев выбора идеи проекта.
4. Для чего нужен README.md в репозитории?
5. Какие разделы обязательно должны быть в README.md учебного проекта?
6. Что такое «ценностное предложение» (value proposition)?
7. Назовите три антипаттерна командной работы.
8. Какие принципы лицензирования открытого ПО нужно учесть при выборе идеи?
9. Какой инструмент для ведения репозитория вы знаете, кроме Git?
10. Что такое «техническое задание» (ТЗ) и на каком этапе его разрабатывают?

Критерии оценивания

Параметр	Баллы
Чёткость описания процесса формирования команды	2
Качество анализа идей (не менее 5, обоснование выбора)	3
Полнота README.md (все разделы, грамотное оформление)	3
Наличие репозитория и корректной ссылки	1
Выводы и готовность к проекту	1

Лабораторная работа № 2

Разработка устава проекта и определение стейкхолдеров

Цель: Освоить этап инициации проекта.

Теоретическое обоснование

Устав проекта (Project Charter) – это документ высокого уровня, который формально авторизует проект и даёт менеджеру право использовать ресурсы. Согласно стандартам PMBOK (PMI), устав должен содержать: обоснование проекта (зачем?), измеримые цели (SMART), высокоуровневые требования, границы проекта (что входит, а что исключено), ключевые вехи, предварительную бюджетную рамку, основные риски, список стейкхолдеров, критерии успеха, а также ограничения и допущения. **Ограничения (constraints)** – это жёсткие рамки: фиксированный бюджет, жёсткий дедлайн, обязательные технологии. **Допущения (assumptions)** – это предположения, которые принимаются за истину без доказательств (например, «доступ к серверу будет предоставлен в течение недели»). Если допущение ошибочно, возникает риск.

Особое внимание уделяется целям. SMART-критерии: Specific (конкретная), Measurable (измеримая), Achievable (достижимая), Relevant (релевантная – согласованная с общей стратегией), Time-bound (ограниченная по времени). Пример не-SMART: «Сделать хороший интернет-магазин». SMART: «Запустить MVP интернет-магазина книг с функцией онлайн-оплаты через две

платёжные системы через 2 месяца, обеспечив конверсию не менее 2% на тестовой группе из 100 пользователей».

Стейкхолдеры (заинтересованные стороны) – это отдельные лица или организации, которые активно влияют на проект или на которых проект влияет. Для их идентификации используют методы: мозговой штурм, интервью, анализ документов. Затем строится матрица «власть – интерес» (Power/Interest Grid). Она делит стейкхолдеров на 4 группы:

1) Высокая власть, высокий интерес → ключевые игроки (необходимо активно вовлекать, часто отчитываться).
2) Высокая власть, низкий интерес → держать удовлетворёнными (информировать по существу).
3) Низкая власть, высокий интерес → информировать подробно (волонтёры, тестировщики).
4) Низкая власть, низкий интерес → минимальный мониторинг.

Для каждого стейкхолдера разрабатывается стратегия коммуникации (как часто, в каком формате, кто отвечает). Например, для заказчика (высокая власть, высокий интерес) – еженедельные демо-отчёты и встречи. Для технического специалиста из смежного отдела (низкая власть, высокий интерес) – рассылка протоколов совещаний. Грамотный анализ стейкхолдеров предотвращает конфликты и повышает вероятность поддержки проекта

Методика и порядок выполнения работы

Задание 1. Для выбранного варианта проекта (1–10) выполнить:

1. Сформулировать SMART-цели (минимум 3).
2. Определить границы (что входит/не входит в проект).
3. Выявить ограничения (бюджет, время, технологии) и допущения.
4. Составить список стейкхолдеров (не менее 8).
5. Построить матрицу «власть – интерес» и предложить план коммуникаций.

Пример для варианта 1 (Интернет-магазин книг): цели – запустить MVP через 2 месяца, обеспечить онлайн-оплату, интеграцию с 2 платёжными системами.

Содержание отчета и его форма

1) Титульный лист с названием ЛР, ФИО участников, вариант.
2) Устав проекта (текст документа 1–2 стр.).
3) Таблица стейкхолдеров (роль, интерес, власть, стратегия).
4) Матрица стейкхолдеров (графическое изображение).
5) Ссылка на коммиты в репозитории, где добавлены эти документы.
6) Выводы о ключевых заинтересованных лицах.
7) Ответы на контрольные вопросы.

Вопросы для защиты работы

1. Какие элементы обязательно входят в устав проекта?
2. Что такое SMART-критерии для цели?
3. В чём разница между ограничением и допущением?
4. Кто такие «скрытые стейкхолдеры»? Приведите пример.
5. Какие стратегии взаимодействия существуют для квадранта «высокая власть – низкий интерес»?
6. Зачем нужна матрица стейкхолдеров?
7. Какую информацию содержит раздел «границы проекта»?
8. Что такое «предположения рисков» в уставе?
9. Кто должен подписывать устав проекта в реальном бизнесе?
10. Назовите три типичных ограничения проекта.

Критерии оценивания

Параметр	Баллы
SMART-цели (реалистичность, измеримость)	2
Полнота устава (границы, ограничения,	3

допущения)	
Стейкхолдеры (не менее 8, разнообразие)	2
Качество матрицы и стратегий взаимодействия	2
Оформление, ссылка на репозиторий	1

Лабораторная работа № 3
Декомпозиция работ (WBS) и оценка трудоемкости

Цель и содержание работы: Научиться декомпозировать проект на задачи и оценивать их трудоемкость.

Теоретическое обоснование

Иерархическая структура работ (WBS, Work Breakdown Structure) – это фундамент всего планирования. WBS представляет собой ориентированную на результат декомпозицию проекта на более мелкие, управляемые компоненты – пакеты работ (work packages). Принципы построения WBS:

1. Правило 100% – WBS включает все работы, необходимые для завершения проекта, и только их.
2. Отсутствие перекрытий – каждый элемент WBS должен быть уникальным, не пересекаться с другим.
3. Декомпозиция до уровня, позволяющего надёжно оценить трудоёмкость (обычно 8–80 человеко-часов).
4. Результат-ориентированность – не действия (написать код), а результаты (модуль авторизации).

WBS может быть представлена в виде диаграммы (дерева) или таблицы с иерархическими номерами (1.1, 1.2, 1.2.1 и т.д.). Каждый пакет работ должен иметь чёткого ответственного. Для IT-проектов типичная декомпозиция: анализ требований → проектирование → разработка (по модулям) → тестирование → документирование → внедрение. Глубина WBS для учебного проекта – 3–4 уровня, 20–30 пакетов работ.

Оценка трудоёмкости – одна из самых сложных задач. Методы:

- Экспертный опрос – опрашиваются специалисты, усредняются оценки.
- Аналоговый – на основе похожих проектов.
- Параметрический – использует математические модели (например, COCOMO).
- Покер планирования (Planning Poker) – agile-метод: каждый член команды независимо даёт оценку в относительных единицах (story points) или часах, затем оценки обсуждаются и достигается консенсус. Этот метод снижает когнитивные искажения (например, эффект якорения).

Важно различать трудоёмкость (человеко-часы) и длительность (календарные дни). При одной и той же трудоёмкости длительность может быть меньше, если параллельно работают несколько человек, но из-за коммуникационных потерь (закон Брукса) добавление новых людей может замедлить проект. Результаты оценки фиксируются в таблице: ID, название, ответственный, трудоёмкость, предшественники. Правильно построенная WBS и реалистичная оценка – залог успешного календарного планирования.

Методика и порядок выполнения работы

Задание 1. Для своего варианта проекта:

1. Построить WBS в виде диаграммы (не менее 20 задач на 3–4 уровнях).
2. Провести оценку трудоемкости каждой задачи методом Planning Poker (имитация: каждый участник дает оценку, затем усреднение).
3. Заполнить таблицу с колонками: ID, Название задачи, Исполнитель (роль), Трудоёмкость (чел.-часы), Предшественники.
4. Указать общую трудоёмкость проекта в человеко-часах.

Примечание: для вариантов 1–10 типовые работы: анализ, дизайн, разработка модулей, тестирование, документирование.

Содержание отчета и его форма

1) Титульный лист с названием ЛР, ФИО участников, вариант.
2) Диаграмма WBS (можно в виде текстовой иерархии или рисунка).
3) Таблица трудоёмкости с указанием метода оценки.
4) Расчёт общей трудоёмкости.
5) Выводы о наиболее трудоёмких задачах и рисках переоценки/недооценки.
6) Ответы на контрольные вопросы.

Вопросы для защиты работы

1. Что такое «пакет работ» (work package) в WBS?
2. Какие ошибки чаще всего допускают при построении WBS?
3. В чём суть метода Planning Poker?
4. Как пересчитать трудоёмкость в календарные дни при заданной численности команды?
5. Что такое «аналоговая оценка»?
6. Зачем нужен словарь WBS (WBS dictionary)?
7. Назовите три способа представления WBS (форматы).
8. Почему трудоёмкость тестирования часто недооценивается?
9. Как учесть «потери на коммуникацию» при оценке?
10. Что такое «точка невозврата» в декомпозиции?

Критерии оценивания

Параметр	Баллы
Корректность WBS (принципы 100%, непересекаемость)	3
Детализация (≥20 задач, ≥3 уровней)	2
Обоснованность оценок трудоёмкости (покер, логика)	3

Полнота таблицы (исполнители, предшественники)	1
Анализ и выводы	1

Лабораторная работа 4

Календарное планирование и диаграмма Ганта

Цель: Освоить инструменты календарного планирования.

Теоретическое обоснование

Календарное планирование превращает WBS и оценки в расписание. **Основной инструмент** – диаграмма Ганта (Gantt chart), предложенная Генри Гантом в 1910-х годах. Она представляет собой горизонтальные полосы, где каждая полоса – задача, её длина пропорциональна длительности, а положение по оси времени – датам начала и окончания. Диаграмма Ганта наглядно показывает параллельность работ, последовательность, вехи (**milestones** – ключевые события, обычно нулевой длительности, обозначаются ромбом или звездой).

Для построения расписания необходимо определить логические зависимости между задачами. Типы зависимостей (связей) в сетевом планировании:

- Finish-to-Start (FS) – задача B начинается после завершения A (самый частый тип).
- Start-to-Start (SS) – B начинается, когда начинается A (например, написание тестов параллельно с кодом).
- Finish-to-Finish (FF) – B заканчивается вместе с A (например, окончание проверки документации вместе с её написанием).
- Start-to-Finish (SF) – B заканчивается, когда A начинается (редкий, используется в расписаниях смен).

Кроме того, учитываются задержки (lag) и опережения (lead). После построения сетевого графика (или диаграммы Ганта с линиями связей)

вычисляется критический путь – самая длинная цепочка задач с нулевым резервом времени. Задачи на критическом пути не имеют задержек; их задержка отодвигает весь проект. Менеджер должен следить именно за этими задачами. Остальные задачи имеют свободный и полный резерв – время, на которое их можно задержать без срыва финального срока.

Назначение ресурсов – ещё один важный шаг. Каждой задаче назначаются исполнители (или роли). При этом рассчитывается загрузка ресурсов (resource loading) – процент времени каждого участника, занятого в проекте. Если загрузка превышает 100% (например, один разработчик назначен на две задачи, идущие параллельно), возникает перегрузка. Её устраняют с помощью ресурсного левеллинга (resource leveling): сдвиг задач, добавление ресурсов, изменение зависимостей. Инструменты: MS Project, GanttProject, Jira (плагин BigGantt), LibreOffice Calc (с макросами). Результат – реалистичное расписание, согласованное с командой.

Методика и порядок выполнения работы

Задание 1. На основе WBS из ЛР3:

1. Определить логические связи между задачами (4 типа зависимостей).
2. Назначить ресурсы (участников команды) с учётом их ролей и доступности (например, 20 часов в неделю).
3. Построить диаграмму Ганта (в любом инструменте).
4. Рассчитать критический путь (выделить красным).
5. Проанализировать загрузку каждого участника (таблица перегрузок).
6. Предложить способ выравнивания ресурсов (ресурсный левелинг).

Содержание отчета и его форма

1) Скриншот/файл диаграммы Ганта.
2) Таблица зависимостей задач (ID задачи, тип зависимости, ID предшественника).
3) Расчёт критического пути (перечислить задачи).

4) Анализ загрузки ресурсов (график или таблица).
5) Выводы о реалистичности расписания.
6) Ответы на контрольные вопросы.

Вопросы для защиты работы

1. Какие типы зависимостей задач существуют? Приведите пример «старт-финиш».
2. Что такое «критический путь»? Как его найти на диаграмме Ганта?
3. Чем отличается диаграмма Ганта от сетевого графика?
4. Что означает «отрицательный резерв времени»?
5. Как влияет назначение ресурса с частичной занятостью на длительность?
6. Что такое «веха» (milestone) и как её обозначают?
7. Назовите три инструмента для построения диаграммы Ганта.
8. Как учесть выходные и праздники при планировании?
9. Что такое «сжатие проекта» (crashing)?
10. Какой недостаток классической диаграммы Ганта (без линий связи)?

Критерии оценивания

Параметр	Баллы
Корректность логических связей	2
Полнота диаграммы (все задачи из WBS, сроки)	2
Правильное определение критического пути	3
Анализ загрузки ресурсов и предложения по левеллингу	2
Оформление, наличие вех	1

Лабораторная работа № 5

Управление рисками в проекте

Цель работы: Научиться идентифицировать и анализировать риски.

Теоретическое обоснование

Риск – это неопределённое событие или условие, которое в случае наступления оказывает положительное или отрицательное влияние на цели проекта. Управление рисками (Risk Management) по PMBOK включает 6 процессов: планирование управления рисками, идентификация, качественный анализ, количественный анализ, планирование реагирования, мониторинг и контроль. В рамках лабораторной работы мы фокусируемся на идентификации, анализе и реагировании.

Идентификация рисков должна быть всесторонней. Методы: мозговой штурм, метод Дельфи, анализ контрольного списка (риск-чеклист), диаграммы причинно-следственных связей (Исикава), анализ допущений. Для IT-проектов типовые категории рисков:

- Технические (сложность интеграции, недостаточная производительность, отказ оборудования).
- Организационные (текучка кадров, конфликты в команде, смена приоритетов).
- Внешние (изменение законодательства, действия конкурентов, форс-мажоры).
- Управленческие (нереалистичные сроки, слабый контроль, плохая коммуникация).

Качественный анализ оценивает вероятность (обычно шкала 1–5) и влияние (на срок, бюджет, качество). Результат – матрица «Вероятность – Влияние», где риски ранжируются на красные (высокий приоритет), жёлтые и зелёные. Количественный анализ (не всегда выполняется в учебном проекте) использует методы: дерево решений, моделирование Монте-Карло, анализ чувствительности. Он даёт числовую оценку (например, «вероятность

превышения бюджета на 20% составляет 15%»).

Планирование реагирования на риски включает четыре основные стратегии для негативных рисков:

- Избежание – устранение причины (например, отказ от нестабильной технологии).
- Минимизация (смягчение) – снижение вероятности или последствий (например, дополнительное тестирование).
- Передача – перекладывание на третью сторону (страхование, аутсорсинг).
- Принятие – пассивное (ничего не делать) или активное (создать резерв времени/бюджета).

Для позитивных рисков (возможностей) – использование, разделение, усиление, принятие. Все данные фиксируются в реестре рисков (Risk Register) – таблице с полями: ID, описание, категория, вероятность, влияние, приоритет, план реагирования, ответственный, статус. Реестр регулярно обновляется в ходе проекта.

Методика и порядок выполнения работы

Задание 1. Для своего проекта:

1. Выявить не менее 10 рисков (технических, организационных, внешних).
2. Оценить вероятность (1–5) и влияние (1–5).
3. Построить матрицу «Вероятность – Влияние» (разделить на зоны: красная, жёлтая, зелёная).
4. Для 3-х наиболее опасных рисков разработать план реагирования с конкретными действиями.
5. Заполнить реестр рисков.

Содержание отчета и его форма

1) Титульный лист с названием ЛР, ФИО участников, вариант.
2) Реестр рисков (таблица на 10+ строк).
3) Матрица «вероятность–влияние» (график или цветовая сетка).
4) Подробный план реагирования для топ-3 рисков.
5) Рекомендации по мониторингу рисков в ходе спринтов
6) Ответы на контрольные вопросы.

Вопросы для защиты работы

1. Чем отличается риск от проблемы (issue)?
2. Какие методы количественного анализа рисков вы знаете?
3. Что такое «триггер риска»? Приведите пример.
4. В каком случае применяют стратегию «передача риска»?
5. Что такое «резерв на непредвиденные обстоятельства» (контингент)?
6. Назовите три типичных риска IT-проектов.
7. Как часто нужно пересматривать реестр рисков?
8. Что такое «остаточный риск»?
9. Какие ошибки бывают при оценке вероятности?
10. Что входит в план реагирования на риск?

Критерии оценивания

Параметр	Баллы
Количество и разнообразие рисков (≥10)	2
Корректность оценки вероятности/влияния	2
Качественная матрица и приоритизация	2
Реалистичность планов реагирования	3
Оформление реестра	1

Лабораторная работа № 6

Настройка системы управления задачами (Jira/Trello)

Цель: Освоить инструменты для отслеживания задач.

Теоретическое обоснование

Системы управления задачами (Issue Tracking Systems) – это инструменты, обеспечивающие прозрачность, подотчётность и координацию в проекте. Они позволяют превратить WBS в набор задач (issues), назначить исполнителей, установить приоритеты и сроки, отслеживать статус. Наиболее популярны: Jira (де-факто стандарт для Scrum/ Kanban), Trello (простая канбан-доска), Asana, YouTrack, ClickUp.

Jira (от Atlassian) поддерживает как классическое управление проектами, так и гибкие методологии. Ключевые понятия:

- Проект – контейнер для задач.
- Типы задач (Issue Types): Epic (крупная функция), Story (пользовательская история), Task (задача), Subtask (подзадача), Bug (дефект).
- Статусы (статусы) и Workflow (жизненный цикл задачи): например, Open → In Progress → In Review → Done. Переходы между статусами могут быть автоматическими (при коммите с ключом задачи).
- Приоритеты: Highest, High, Medium, Low, Lowest.
- Компоненты (Components) – логические части системы (например, «База данных», «API», «Фронтенд»).
- Метки (Labels) – свободные теги для поиска.
- Спринты (Scrum) – итерации фиксированной длины.
- Доска (Board) – визуализация потока задач (Kanban или Scrum).

Trello – упрощённый аналог, использующий доски, списки (например, «To Do», «Doing», «Done») и карточки. В карточку можно добавить чек-листы, метки, дату, вложения. Power-ups (например, плагин для диаграммы сгорания) расширяют функционал. Для учебных проектов Trello часто бывает достаточным.

Настройка системы начинается с создания проекта и доски. Затем в соответствии с WBS создаются задачи (каждая задача – отдельный issue). Важно

установить связи между задачами (блокирует/блокируется, связана с эпиком). Для Scrum-досок задачи оцениваются в story points (относительная сложность). Для Kanban – ограничивается количество задач в столбце «В работе» (WIP limit) для выявления узких мест. Правильно настроенная система управления задачами становится единым источником правды о ходе работ, позволяет генерировать отчёты (burndown, velocity, cumulative flow) и существенно снижает хаос

.

Методика и порядок выполнения работы

Задание 1. 10 индивидуальных заданий по вариантам проектов.

Общее требование для всех вариантов:

1. В отчёте представить скриншоты доски, карточки задачи, списка задач, настроек workflow (для Jira) или автоматизаций (для Trello).
2. Ссылка на проект (с доступом для проверяющего).
3. Выводы о том, насколько выбранный инструмент и конфигурация подходят для вашего типа проекта.

Вариант	**Название проекта**	**Индивидуальное задание (дополнительные условия)**
1	Интернет-магазин книг	Настроить доску с колонками: «Бэклог», «Анализ», «Разработка», «Тестирование», «Готово». Создать эпики: «Каталог», «Корзина», «Оплата», «Личный кабинет». Добавить метки: «фронтенд», «бэкенд», «база данных».
2	Мобильное приложение «Фитнес-трекер»	Настроить Scrum-доску со спринтами длиной 2 недели. Создать компоненты: «Тренировки», «Питание», «Статистика», «Уведомления». Для каждой задачи указать story points (числа Фибоначчи: 1,2,3,5,8).
3	Система управления задачами (аналог Trello)	Настроить Kanban-доску с WIP-ограничениями: «В работе» – 3 задачи, «На проверке» – 2 задачи. Добавить поле «Приоритет» (Critical, High, Medium, Low). Создать автоматизацию: при переходе в «Готово» – переместить

		в архив.
4	Чат-бот для службы поддержки	Настроить доску с колонками: «Идеи», «Планирование», «Разработка бота», «Интеграция с API», «Тестирование», «Документирование», «Запущено». Добавить теги: «intent», «webhook», «logging».
5	Веб-сервис для онлайн-бронирования столиков в кафе	Создать проект в Jira. Настроить workflow для задач типа Bug: Open → In Progress → In Review → Done, но для Story: Open → Analysis → Dev → Testing → Done. Добавить компонент «Интеграция с Google Calendar».
6	Приложение для изучения иностранных слов (карточки)	Использовать Trello. Создать 4 списка: «Backlog», «Next sprint», «In progress», «Done». В карточках добавить чек-листы (например, «дизайн карточки», «алгоритм интервального повторения»). Назначить членов команды на карточки.
7	Платформа для обмена студенческими конспектами	Настроить доску с метками: «высокий приоритет», «средний», «низкий». Создать 3 эпика: «Загрузка файлов», «Рейтинг и комментарии», «Поиск и фильтры». Добавить пользовательское поле «Оценка трудоёмкости в часах» (числовое).
8	Система мониторинга доступности веб-сайтов	Настроить Scrum-доску в Jira. Создать версии (releases): «Alpha», «Beta», «Release». Привязать задачи к версиям. Настроить бёрндаун-чарт. Добавить задачу-баг с типом «Сбой мониторинга» и приоритетом Highest.
9	Приложение «Калькулятор расходов семьи»	Настроить Kanban-доску с колонками: «Требования», «Дизайн», «Фронтенд», «Бэкенд», «Тестирование», «Готово». Добавить WIP-лимит: «Тестирование» – не более 2 задач. Использовать метки: «отчеты», «графики», «мультипользователь».

10	Генератор расписания для вуза	Создать проект в Trello, добавить power-up «Calendar» для отображения дедлайнов. Настроить автоматизацию (Butler): если задача находится в колонке «Сделать» более 3 дней, то добавлять метку «просрочено». Создать не менее 15 задач на основе WBS.

Содержание отчета и его форма

1) Титульный лист с названием ЛР, ФИО участников, вариант.
2) Выбор инструмента. Обоснование выбора: Jira или Trello (сравнение, почему выбран именно этот инструмент для вашего проекта).
3) Настройка проекта и доски. Скриншот созданного проекта (название, тип проекта). Скриншот доски с указанием выбранной методологии (Scrum или Kanban). Описание колонок (статусов) и их назначения. Если настроены WIP-ограничения – указать их.
4) Настройка workflow (жизненного цикла задачи). Схема или текстовое описание переходов между статусами. Скриншот настроек workflow (для Jira) или правил автоматизации Butler (для Trello). Пример одного правила автоматизации (если настроено).
5) Создание задач. Таблица с перечнем созданных задач (не менее 15) в соответствии с WBS из ЛР3: | ID задачи | Название | Тип (Story/Task/Bug) | Приоритет | Исполнитель | Story points / часы | Дедлайн | Связь с эпиком |. Скриншот списка задач (backlog или доска).
6) Детальная карточка задачи. Скриншот одной (любой) карточки с заполненными всеми полями: описание, чек-листы, комментарии, вложения, исполнитель, сроки, теги, компоненты
7) Использование дополнительных возможностей. Перечень использованных меток (labels) и компонентов (components). Скриншоты настроек эпиков (если есть). Скриншоты отчётов (burndown chart, cumulative flow diagram – для Jira).
8) Анализ и выводы. Насколько удобно настроенная система отражает

реальные задачи проекта? Какие сложности возникли при настройке? Что можно улучшить?

9) Ответы на контрольные вопросы.

Вопросы для защиты работы

1. Какие типы задач (issue types) существуют в Jira и для чего каждый используется?
2. В чём принципиальное отличие доски Scrum от доски Kanban?
3. Как в Trello реализовать оценку трудоёмкости, если нет встроенной функции story points?
4. Что такое WIP-лимит и для чего он нужен в Kanban?
5. Как в Jira настроить автоматический переход задачи из статуса «In Progress» в «Done» при коммите с определённым комментарием?
6. Какие метрики и отчёты можно получить из Jira для анализа производительности команды?
7. Что такое «эпик» (epic) и как он связан с пользовательскими историями?
8. Перечислите не менее трёх способов группировки задач в Jira/Trello (компоненты, метки, версии и т.д.).
9. Как в Trello с помощью power-up «Calendar» визуализировать дедлайны задач?
10. Какие права доступа к проекту можно назначить в Jira для ролей «администратор», «разработчик», «наблюдатель»?

Критерии оценивания

Параметр	Баллы
Обоснование выбора инструмента (логичность, учёт типа проекта)	1
Корректность настройки доски и методологии (соответствие Scrum/Kanban, наличие колонок, WIP при необходимости)	2
Полнота и соответствие WBS (создано не менее 15 задач, они отражают WBS из ЛР3)	2
Качество заполнения карточек (приоритет, исполнитель,	2

оценка, дедлайн, описание – заполнены осмысленно)	
Настройка workflow или автоматизаций (наличие переходов, скриншоты, хотя бы одно правило)	1
Использование меток/компонентов/эпиков (не менее двух видов группировки)	1
Оформление отчёта (структура, скриншоты, ссылка, грамотность)	1
Примечание: *Если студент использовал Trello вместо Jira, критерии применяются с учётом возможностей Trello (например, автоматизации через Butler, power-ups вместо встроенных workflow). Отсутствие какого-либо элемента, не поддерживаемого инструментом, не снижает оценку, но студент должен указать это в отчёте.*	

Лабораторная работа № 7

Scrum-симуляция: проведение спринта

Цель: Применить на практике Scrum-церемонии.

Теоретическое обоснование

Scrum – это фреймворк для управления сложными продуктами, основанный на итеративной разработке. Он определяет роли (Product Owner, Scrum Master, команда разработки), артефакты (Product Backlog, Sprint Backlog, Increment) и события (Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective). **Цель Scrum** – обеспечить гибкость, быструю обратную связь и непрерывное улучшение.

Спринт – это итерация фиксированной длительности (обычно 1–4 недели), в конце которой создаётся готовый, пригодный к использованию инкремент продукта. Длительность спринта не меняется в течение проекта.

Планирование спринта (Sprint Planning, до 8 часов для 4-недельного спринта) отвечает на вопросы: что может быть доставлено в этом спринте? как будет выполняться работа? Команда выбирает задачи из Product Backlog (обычно ранжированного Product Owner'ом), формулирует цель спринта (Sprint Goal) – краткое описание того, зачем нужен спринт, и создаёт Sprint Backlog. Задачи оцениваются в часах или стори поинтах.

Ежедневный Scrum (Daily Stand-up, 15 минут) проводится в одно и то же время. Каждый участник отвечает на три вопроса:

1. Что я сделал вчера, чтобы помочь команде достичь цели спринта?
2. Что я сделаю сегодня?
3. Вижу ли я какие-либо препятствия (блокеры)?

Scrum Master отвечает за устранение препятствий. Daily Scrum не является отчётом перед начальством, а служит для синхронизации.

Обзор спринта (Sprint Review) проводится в конце спринта. Команда демонстрирует заинтересованным лицам (стейкхолдерам) готовый инкремент. Обсуждается, что было сделано, какие изменения в продукте произошли, корректируется Product Backlog. Это не презентация отчёта, а интерактивная

встреча, дающая обратную связь.

Ретроспектива спринта (Sprint Retrospective) – внутреннее событие для команды, где анализируется процесс: что прошло хорошо, что плохо, какие улучшения внедрить в следующем спринте. Форматы: «Start, Stop, Continue», «4Ls» (Liked, Learned, Lacked, Longed for), анализ диаграммы выгорания задач. Результат – конкретный план действий по улучшению (не более 2–3 пунктов).

Проведение полного цикла Scrum-церемоний (даже в симуляции на 1 неделю) даёт студентам бесценный опыт самоорганизации и адаптации. После спринта оформляется отчёт, содержащий бэклог спринта, burndown chart, результаты ретроспективы.

Методика и порядок выполнения работы

Задание 1. Для своего проекта (имитация спринта длительностью 1 календарная неделя):

1. Провести планирование спринта: выбрать задачи из бэклога на 40 часов работы команды, сформулировать цель спринта.
2. Организовать **ежедневные стендапы** (5 дней – по 15 мин) – протокол (дата, кто участвовал, что сделано, планы, блокеры).
3. В конце спринта – **обзор**: подготовить демо работающего фрагмента (можно макет, код, документацию).
4. **Ретроспектива**: выявить 3 проблемы и 3 улучшения, оформить план действий.
5. Оформить **отчёт по спринту** (см. требования).

Содержание отчета и его форма

1) Титульный лист с названием ЛР, ФИО участников, вариант.
2) Цель спринта и список задач с оценками (бэклог спринта).
3) Протоколы ежедневных стендапов (таблица с датами).
4) Burndown chart (фактический vs идеальный).
5) Отчёт об обзоре (что продемонстрировано, обратная связь).
6) Результаты ретроспективы (таблица «Проблемы – Улучшения – Действия»).

7) Выводы о работе команды в спринте.
8) Ответы на контрольные вопросы.

Вопросы для защиты работы

1. Какие три вопроса задаются на ежедневном стендапе?
2. Кто может отменять спринт и по каким причинам?
3. Чем отличается обзор спринта от ретроспективы?
4. Что такое «бэклог спринта» и кто его заполняет?
5. Как определяется «готовая работа» (Definition of Done)?
6. Роль Scrum-мастера – чем он отличается от тимлида?
7. Что такое «выгорание задач» (burndown chart) и как его построить?
8. Как часто должен встречаться владелец продукта с командой?
9. Можно ли добавить задачу в спринт после его старта?
10. Назовите два антипаттерна ретроспективы.

Критерии оценивания

Параметр	Баллы
Качество планирования (цель, реалистичность)	2
Ведение стендапов (полнота протоколов)	2
Демо и обзор (наличие работающего результата)	2
Ретроспектива (глубина анализа, конкретные действия)	3
Burndown chart и выводы	1

Лабораторная работа № 8

Разработка технической документации проекта

Цель: Освоить навыки документирования проекта.

Теоретическое обоснование

Техническая документация – это неотъемлемая часть жизненного цикла ПО. Она обеспечивает передачу знаний, упрощает поддержку, снижает риски потери информации при смене команды. В зависимости от аудитории документация делится на:

- Техническое задание (ТЗ) – для заказчика и разработчиков, фиксирует требования.
- Архитектурная документация – для разработчиков и архитекторов.
- Пользовательская документация – для конечных пользователей.
- Документация по развёртыванию и администрированию – для DevOps и системных администраторов.

Техническое задание по ГОСТ 34.602 содержит разделы: общие сведения (название, организация-заказчик), назначение и цели создания системы, требования к системе (функциональные, нефункциональные – надёжность, безопасность, производительность), требования к информационной совместимости, состав и параметры технических средств, требования к документированию, стадии и этапы разработки, порядок контроля и приёмки. Альтернативный подход – User Stories в Agile. User Story описывает потребность пользователя на естественном языке: «Как [роль], я хочу [действие], чтобы [ценность]». Каждая история сопровождается критериями приёмки (Acceptance Criteria) – конкретными, проверяемыми условиями, которые определяют, когда история выполнена. Пример: «Как покупатель, я хочу фильтровать книги по жанру, чтобы быстрее находить интересующие меня произведения». Критерии: фильтр отображается на странице каталога; при выборе жанра список книг обновляется без перезагрузки страницы; если книг нет – показывается сообщение.

Архитектурная документация описывает высокоуровневую структуру системы. Популярный язык – UML (Unified Modeling Language) или метод C4 (Context, Containers, Components, Code). На начальном этапе достаточно диаграммы

компонентов (Component diagram) или диаграммы контейнеров C4, показывающей основные сервисы (веб-сервер, БД, кэш, внешние API). К диаграмме прилагается текстовое описание архитектурных решений (например, «клиент-сервер с REST API», «микросервисная архитектура», «использование очередей сообщений RabbitMQ»).

Пользовательская документация (руководство пользователя) пишется простым языком, содержит: инструкцию по установке (если требуется), регистрацию/вход, описание основных функций с иллюстрациями (скриншотами), описание типичных ошибок и их решений (FAQ). Руководство может быть в формате PDF, встроенной справкой веб-приложения или Wiki в репозитории. Хорошая документация – это инвестиция, которая многократно окупается в будущем.

Методика и порядок выполнения работы

Задание 1. Для своего варианта:

1. Составить ТЗ (структура: назначение, требования к функционалу, требования к надёжности, состав и параметры).
2. Либо написать 10 User Stories с Acceptance Criteria (формат: «Как <роль>, я хочу <действие>, чтобы <ценность>»).
3. Описать архитектуру (диаграмма компонентов или C4 level 1-2) – текстовое описание + рисунок.
4. Написать пользовательское руководство (установка, первый запуск, основные функции, экранные формы).
5. Поместить все документы в репозиторий (папка /docs).

Содержание отчета и его форма

1) Титульный лист с названием ЛР, ФИО участников, вариант.
2) Техническое задание / User Stories (текст 2–3 стр.).
3) Диаграмма архитектуры (скриншот или код PlantUML).
4) Пользовательское руководство (1–2 стр. с картинками).
5) Ссылка на папку /docs в репозитории.
6) Ответы на контрольные вопросы.

Вопросы для защиты работы

1. Какие разделы обязательно содержит ТЗ по ГОСТ?
2. Что такое «критерии приемки» (Acceptance Criteria)?
3. В чём разница между функциональными и нефункциональными требованиями?
4. Какие виды диаграмм UML чаще всего используют для архитектуры ПО?
5. Что такое «архитектурное решение» и как его документировать?
6. Для кого предназначено пользовательское руководство?
7. Какие инструменты помогают генерировать документацию из кода (например, Doxygen)?
8. Что такое «ReadTheDocs» и как он связан с Sphinx?
9. Почему важно версионировать документацию вместе с кодом?
10. Какие требования предъявляют к руководству администратора?

Критерии оценивания

Параметр	Баллы
Полнота и корректность ТЗ / User Stories	3
Качество архитектурного описания (диаграмма + пояснения)	3
Пользовательское руководство (понятность, полнота)	3
Структура и размещение в репозитории	1

Лабораторная работа № 9

Презентация и защита проекта

Цель: Представить результаты управления проектом.

Теоретическое обоснование

Защита проекта – это не просто демонстрация продукта, а демонстрация управленческой компетенции. Ключевой элемент – план-фактный анализ (сравнение запланированных и фактических показателей). Он позволяет оценить точность планирования, выявить проблемные места и извлечь уроки. Анализируются три основных параметра:

- Сроки (плановая длительность vs фактическая).
- Трудоёмкость (план в человеко-часах vs факт).
- Бюджет (если применимо, в учебном проекте – условно).

Отклонения могут быть положительными (сделали раньше/меньше) и отрицательными (опоздание/перерасход). Важно не просто констатировать факт, но и объяснить причины. Причины следует связывать с рисками, зафиксированными в ЛР5: например, «риск недооценки сложности интеграции API сбылся, поэтому задача “Подключение платёжного шлюза” заняла на 8 часов больше». Также анализируются изменения требований (scope creep), проблемы в команде (болезнь, конфликт), технические трудности.

Метрики, используемые в план-фактном анализе:

- Индекс выполнения сроков SPI = Earned Value / Planned Value. SPI < 1 – отставание.
- Индекс выполнения стоимости CPI (если есть бюджет).
- Процент завершённых задач по отношению к плану.

Для наглядности строят совмещённую диаграмму Ганта, где плановые полосы отображаются одним цветом, а фактические даты начала/окончания – другим. Также полезны диаграмма накопленного потока (cumulative flow diagram) из Jira и диаграмма выгорания задач (burndown) для спринтов.

Итоговый отчёт (10–15 страниц) должен включать:

- Краткое описание продукта и команды.
- Все артефакты (устав, WBS, диаграмму Ганта, реестр рисков,

документацию).

- План-фактный анализ в табличной форме.
- Выводы: что удалось, что не удалось, почему.
- Рекомендации для будущих проектов (Lessons Learned).

Презентация на 5–7 слайдов должна быть сжатой, визуально насыщенной (графики, диаграммы, скриншоты), содержать ключевые цифры и не перегружать текстом. Доклад – 5–7 минут. Вопросы преподавателя и аудитории могут касаться как управленческих решений (почему выбрали такую оценку?), так и технических деталей. Умение аргументированно отвечать и признавать ошибки (с анализом причин) – признак зрелого проектного менеджера. Защита завершает цикл управления проектом и даёт ценный опыт ретроспективы всей работы

.

Методика и порядок выполнения работы

Задание 1. Для своего варианта подготовить **итоговый отчёт** и **презентацию**:

1. Собрать все артефакты из ЛР1–ЛР8.
2. Провести план-фактный анализ:
    a. Запланированная трудоёмкость (из ЛР3) vs фактическая (по данным Jira/Trello).
    b. Запланированный календарь (ЛР4) vs фактическая длительность.
    c. Отклонения по каждой крупной задаче (в %).
3. Выявить причины отклонений (связать с рисками из ЛР5).
4. Оценить качество результата (соответствие критериям приёмки из ЛР8).
5. Подготовить презентацию на 5–7 слайдов.
6. Провести публичную защиту (доклад + ответы на вопросы)

Содержание отчета и его форма

1) Титульный лист с названием ЛР, ФИО участников, вариант.
2) Итоговый отчёт (10–15 стр.):
    a. Краткое описание продукта.

b. План-фактная таблица (задача, план (часы), факт (часы), отклонение, причина).
c. Диаграмма Ганта с фактическими сроками (наложением).
d. Анализ рисков, которые сбылись.
e. Выводы и рекомендации.

3) Презентация (ссылка на файл или встроенные слайды).
4) Ответы на контрольные вопросы.

Вопросы для защиты работы

1. Что такое «план-фактный анализ» и зачем он нужен?
2. Как рассчитать индекс выполнения сроков (SPI)?
3. Какие типичные причины перерасхода времени в студенческих проектах?
4. Как измерить качество продукта, если нет реальных пользователей?
5. Что такое «накопленный поток задач» (cumulative flow diagram)?
6. Какие метрики можно вывести из Jira для оценки производительности?
7. Как презентовать отклонения, не создавая негативное впечатление?
8. Какие разделы обязательно должны быть в итоговом отчёте?
9. Что такое «уроки проекта» (lessons learned) и как их оформлять?
10. Назовите три распространённых ошибки при защите проекта.

Критерии оценивания

Параметр	Баллы
Полнота план-фактного анализа (все задачи, отклонения)	3
Глубина причин отклонений (ссылки на риски, события)	2
Качество презентации (структура, наглядность, лаконичность)	2
Устный доклад (соблюдение регламента, чёткость)	2
Ответы на вопросы (аргументированность, знание деталей)	1

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное
автономное образовательное
учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К САМОСТОЯТЕЛЬНОЙ РАБОТЕ

«Управление программными проектами»

Направление подготовки 09.03.04 «Программная инженерия» Направленность (профиль) «Разработка и сопровождение программного обеспечения»
Квалификация выпускника: бакалавр

Ставрополь

2026

ВВЕДЕНИЕ

Цель изучения дисциплины «Управление программными проектами» является формирование у студентов теоретических знаний и практических навыков в области управления проектами разработки программного обеспечения, включая планирование, организацию командной работы, контроль ресурсов и сроков, документирование и обеспечение качества, с учетом правовых норм и стандартов в сфере информационных технологий.

Задачами обучения являются:

изучение методологий управления проектами (традиционные, гибкие, гибридные) и их применимости к различным типам IT-проектов;

освоение инструментов планирования, оценки трудозатрат, управления рисками и контроля исполнения проектов;

формирование навыков работы в команде, распределения ролей и эффективной коммуникации с заинтересованными сторонами;

изучение современных систем контроля версий и инструментов управления задачами (Jira, Trello, GitLab);

освоение методов документирования проекта, подготовки отчетности и презентации результатов;

изучение правовых и этических аспектов управления IT-проектами, включая лицензирование, защиту интеллектуальной собственности и информационную безопасность.

В ходе изучения дисциплины «Управление программными проектами» у обучающегося формируются набор следующих компетенций:

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов

ПК-1 – Способен осуществлять управление программными проектами: планировать, контролировать сроки и ресурсы, применять системы контроля версий и инструменты управления задачами	ИД-1$_{ПК-1}$ Применяет методы декомпозиции работ (WBS), оценки трудоёмкости (PERT, Planning Poker) и календарного планирования (диаграмма Ганта, критический путь) для формирования плана проекта	Способен разрабатывать WBS и диаграмму Ганта, определять критический путь и распределять ресурсы
	ИД-2$_{ПК-1}$ Использует системы контроля версий (Git, Git Flow) и платформы совместной разработки (GitLab, GitHub) для организации командной работы	Способен применять Git для управления исходным кодом и документацией, использовать ветвление и слияние для организации командной работы
	ИД-3$_{ПК-1}$ Применяет инструменты управления задачами (Jira, Trello, YouTrack) для отслеживания прогресса (burndown chart), ведения бэклога и контроля исполнения	Способен использовать инструменты управления задачами (Jira, Trello, GitLab Issues) для постановки задач и контроля сроков выполнения
ПК-9 – Умеет оформлять техническую документацию, готовить презентации и отчеты, вести деловую коммуникацию на русском и иностранном языках в профессиональной среде	ИД-1$_{ПК-9}$ Оформляет техническую документацию (ТЗ, пояснительная записка, руководство пользователя, API-спецификация) в форматах Markdown, OpenAPI, DocBook в соответствии с ГОСТ	Способен разрабатывать и оформлять проектную документацию (устав проекта, ТЗ, план управления проектом) в соответствии со стандартами
	ИД-2$_{ПК-9}$ Готовит презентационные материалы и публично защищает результаты проектной деятельности, аргументированно отвечая на вопросы	Способен создавать презентационные материалы и проводить публичные выступления, защищая результаты проектной деятельности
ПК-10 – Готов соблюдать правовые нормы, стандарты и требования информационной безопасности, участвовать в разработке нормативной и технической документации, учитывать законодательство в области ИТ	ИД-1$_{ПК-10}$ Применяет основные правовые нормы (лицензирование ПО, авторское право, ФЗ-152, ФЗ-149) и стандарты ИБ (ГОСТ Р ИСО/МЭК 27001, ГОСТ Р 56545-2015) при разработке и эксплуатации ПО	Способен применять правовые нормы при выборе лицензий на разрабатываемое ПО, учитывать требования законодательства об авторском праве

1. ОРГАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Самостоятельная работа студентов является важнейшим условием формирования научного способа познания. Она проводится накануне каждого лабораторного занятия и включает изучение необходимого для выполнения лабораторных работ и подготовки к практическим занятиям теоретического материала, а также подготовки отчетов по выполненным лабораторным и практическим занятиям. Подготовленный материал оформляется в виде отчета. Самостоятельные занятия (СЗ) являются одной из активных форм обучения. Самостоятельные занятия по дисциплине Управление программными проектами» имеют целью: - закрепить и углубить знания, полученные студентами на лекциях и в процессе лабораторных и практических занятий; - привить практические навыки в оформлении организационных, распорядительных и информационно-справочных документов. Самостоятельные занятия проводятся в специализированных аудиториях, оборудованных средствами вычислительной техники и возможностью пользования Internet-ресурсами, в библиотеке. Предлагаемые методические рекомендации содержат информацию для студентов, необходимую при подготовке и проведении лабораторных занятий по дисциплине «Управление программными проектами».

2. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ

Самостоятельная работа по дисциплине «Управление программными проектами» состоит из двух частей:

1. Самостоятельное изучение вопросов;
2. Самостоятельная разработка проекта на базе умений и навыков, полученных на лабораторных работах.

3. СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

3.1. Темы для самостоятельного изучения

1. Правовые и этические аспекты управления IT-проектами.
2. Качество в программных проектах.
3. Документирование и отчетность в проекте.

В процессе самостоятельного изучения вопросов курса студент изучает предложенные их по рекомендуемому преподавателем списку литературы и предоставляет преподавателю краткий конспект, по которому в дальнейшем проводится индивидуальное собеседование.

На зачетной неделе студент должен предоставить готовый проект.

4. ОРГАНИЗАЦИЯ КОНТРОЛЯ ЗНАНИЙ, ПОЛУЧЕННЫХ ПРИ ВЫПОЛНЕНИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

В процессе собеседования и защиты самостоятельного проекта:
Оценка «отлично» выставляется студенту, если теоретическое содержание вопросов самостоятельного изучения освоено полностью, без пробелов; исчерпывающе, последовательно, четко и логически стройно излагает материал; свободно справляется с задачами, вопросами и другими видами применения знаний; использует в ответе дополнительный материал все предусмотренные программой задания выполнены, качество их выполнения оценено числом баллов, близким к максимальному; анализирует полученные результаты; проявляет самостоятельность при выполнении заданий.

Оценка «хорошо» выставляется студенту, если теоретическое и практическое содержание вопросов самостоятельной работы освоено полностью, необходимые практические компетенции в основном сформированы, все предусмотренные программой обучения учебные задания выполнены, качество их выполнения достаточно высокое. Студент твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопрос.

Оценка «удовлетворительно» выставляется студенту, если теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, большинство предусмотренных программой заданий выполнено, но в них имеются ошибки, при ответе на поставленный вопрос студент допускает неточности, недостаточно правильные формулировки, наблюдаются нарушения логической последовательности в изложении программного материала. Практические навыки освоены частично.

Оценка «неудовлетворительно» выставляется студенту, если он не знает значительной части программного материала, допускает существенные ошибки, неуверенно, с большими затруднениями выполняет практические работы, необходимые практические компетенции не сформированы, большинство предусмотренных программой обучения учебных заданий не выполнено, качество их выполнения оценено числом баллов, близким к минимальному.

5. РЕКОМЕНДАЦИИ ПО РАБОТЕ С ЛИТЕРАТУРОЙ И ИСТОЧНИКАМИ

5.1 Рекомендации студентам по организации работы с литературой и источниками Изучение литературы и источников необходимо начинать с прочтения соответствующих глав учебных изданий, учебных пособий или литературы, рекомендованной в качестве основной или дополнительной по дисциплине «Документирование и отчетность в проекте», которые прямо или косвенно относятся к отрабатываемой теме. Изучая литературу и источники, студенту рекомендуется вести краткий конспект. Однако не следует переписывать все содержание изучаемой темы, нужно выписывать лишь основные идеи и главные на ваш взгляд мысли. В отдельных случаях, когда встречаются важные определения, понятия, необходимый фактический материал и примеры, статистическая информация, имеющие отношение к изучаемой теме, студенту следует выписать их в виде цитат с полным указанием библиографических источников. Конспектирование рекомендуемой литературы и источников необходимо вести с распределением собранных материалов по отдельным главам и параграфам согласно учебно-тематическому плану. Необходимо выписывать все выходные данные по используемой литературе и источникам. Важным этапом при работе с рекомендуемой литературой иисточниками является изучение законодательных и нормативных актов федерального, регионального, местного и ведомственного уровней. При изучении Указов Президента РФ, Законов и Кодексов РФ, постановлений, положений, рекомендаций и т.д., студент должен выяснить все изменения и дополнения, которые могли быть внесены после их выхода в свет. Основой технологии интенсификации обучения на платформе цифровых образовательных технологий являются учебно-иллюстрационные материалы (опорный конспект) по дисциплине «Управление программными проектами».