Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Баламирзоев Назим Лиодинович
Должность: Ректор
Дата подписания: 01.06.2026 17:08:47
Уникальный программный ключ:
5cf0d6f89e80f49a334f6a4ba58e91f3526b9926

Приложение А

(обязательное к рабочей программе дисциплины)

Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Дагестанский государственный технический университет»

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине «Современные технологии разработки программного обеспечения»

Уровень образования	магистратура
Направление подготовки бакалавр/магистратура/специальность	09.04.03 Прикладная информатика
Профиль подготовки/специализация	Прикладная информатика в управлении финансами

Разработчик _____ Шахбанова И.К., к.э.н., доцент
подпись (ФИО уч. степень, уч. звание)

Фонд оценочных средств обсужден на заседании кафедры ЭБиТД от 14.10.2022 г., протокол № 2.

Зав. кафедрой _____ Шахбанова И.К., к.э.н., доцент
подпись (ФИО уч. степень, уч. звание)

г. Махачкала 2022.

СОДЕРЖАНИЕ

1. Область применения, цели и задачи фонда оценочных средств

Фонд оценочных средств (ФОС) является компонентом рабочей программы дисциплины «Современные технологии разработки программного обеспечения» и предназначен для оценки планируемых результатов обучения, сформированных у обучающихся по направлению подготовки 09.04.03 «Прикладная информатика». ФОС применяется при входном контроле, текущем контроле, рубежной аттестации и промежуточной аттестации в форме экзамена.

Цель ФОС - установить соответствие уровня подготовки обучающихся требованиям ОПОП ВО и рабочей программы дисциплины в части владения современными технологиями разработки ПО: анализ требований, проектирование архитектуры, разработка, тестирование, интеграция компонентов, CI/CD, контейнеризация, контроль качества и безопасная разработка.

Задачи ФОС:

- определить показатели и критерии оценивания компетенций УК-1, ОПК-7 и ПК-4;
- закрепить перечень оценочных средств для входного, текущего, рубежного и промежуточного контроля;
- установить шкалы оценивания результатов обучения;
- обеспечить проверку знаний, умений и навыков по жизненному циклу ПО, архитектуре, тестированию, DevOps, контейнеризации и безопасности;
- обеспечить сопоставимость результатов оценивания на очной и заочной формах обучения.

Официальные источники и стандарты, использованные при формировании оценочных средств:

- ФГОС ВО 09.04.03 «Прикладная информатика»
- Профессиональный стандарт 06.001 «Программист»
- Профессиональный стандарт 06.004 «Специалист по тестированию в области информационных технологий»
- ISO/IEC/IEEE 12207:2017
- ISO/IEC 25010:2023
- OWASP Top 10:2021
- GitHub Actions Documentation
- Docker Documentation
- Kubernetes Documentation
- OpenAPI Specification

2. Описание показателей и критериев оценивания компетенций, формируемых в процессе освоения дисциплины (модуля)

2.1. Перечень компетенций с указанием этапов их формирования в процессе освоения ОПОП

Перечень компетенций сформирован по рабочей программе дисциплины и ФГОС ВО по направлению подготовки 09.04.03 «Прикладная информатика».

Код компетенции	Компетенция	Код и наименование индикатора	Критерии оценивания	Наименование оценочного средства	Контролируемые разделы и темы
УК-1	Способен осуществлять критический анализ проблемных ситуаций на основе системного подхода, вырабатывать стратегию действий	УК-1.1. Знать методы системного и критического анализа проблемных ситуаций в разработке ПО, методы анализа требований, рисков и ограничений проекта.	Показывает знание методов анализа требований и проектных рисков; аргументирует выбор архитектурных и технологических решений.	Входной контроль, тест, кейс-задача, лабораторная работа, экзамен	Темы 1, 2, 3, 4, 9
УК-1		УК-1.2. Уметь применять системный подход при выборе технологий разработки, тестирования и развертывания ПО.	Сравнивает варианты жизненного цикла, декомпозирует задачу на требования, компоненты, тесты и этапы поставки.	Практическое задание, контрольная работа, проект	Темы 2, 3, 4, 5, 8

УК-1		УК-1.3. Владеть методиками постановки цели, выбора способа реализации и оценки результата разработки ПО.	Формирует стратегию решения задачи, выбирает инструменты, фиксирует критерии приемки и контролирует риски.	Индивидуальное задание, защита лабораторной работы	Темы 3, 4, 6, 9
ОПК-7	Способен использовать методы научных исследований и математического моделирования при проектировании и управлении информационными системами	ОПК-7.1. Знать процессы жизненного цикла ПО, методы моделирования требований, архитектуры и качества программных систем.	Оперирует понятиями жизненного цикла, требований, архитектуры, качества, тестирования, сопровождения и DevOps-практик.	Тест, устный опрос, экзамен	Темы 1, 2, 3, 4, 5
ОПК-7		ОПК-7.2. Уметь применять модели, нотации и инструменты проектирования ПО для описания архитектуры и	Создает диаграммы, спецификации API, модели данных, тестовые сценарии и карту поставки программного	Лабораторная работа, кейс-задача	Темы 3, 4, 5, 7

		процессов разработки.	продукта.		
ОПК-7		ОПК-7.3. Владеть инструментами моделирования, контроля качества и автоматизации жизненного цикла ПО.	Использует Git, CI/CD, контейнеризацию, статический анализ, unit-тестирование и документацию проекта.	Практическое задание, проект	Темы 5, 6, 7, 8, 9
ПК-4	Способность интегрировать компоненты и сервисы финансовых ИС	ПК-4.1. Знать архитектурные подходы к интеграции компонентов, сервисов, API и баз данных в прикладных информационных системах.	Понимает REST API, микросервисы, очереди сообщений, контейнерные среды, безопасность и журналирование интеграций.	Тест, лабораторная работа	Темы 3, 4, 5, 6
ПК-4		ПК-4.2. Уметь выполнять интеграцию программных компонентов и	Настраивает API-контракт, репозиторий, pipeline, тестовое окружение и документацию	Лабораторная работа, кейс-проект	Темы 5, 6, 7, 8

		сервисов с учетом требований технического задания.	интеграции.		
ПК-4		ПК-4.3. Владеть навыками практической разработки, тестирования, контейнеризации и развертывания программного продукта.	Представляет работающий прототип, результаты тестов, Dockerfile/Compose-файл, CI/CD-конфигурацию и README.	Итоговый проект, экзамен	Темы 6, 7, 8, 9

2.1.1. Этапы формирования компетенций

Этапы контроля распределены по входному, текущему, рубежному и промежуточному контролю. Итоговая проверка проводится в форме экзамена.

Код компетенции	Контролируемый этап	Вид учебной работы	Оценочное средство	Неделя / тема	Форма фиксации результата	Порог освоения
УК-1	Входной	Актуализация базовых знаний	Вопросы входного контроля	До темы 1	ответы в аудитории / тест	не менее 56%
УК-1	Текущий	Анализ требований и проблемной ситуации	Кейс-задача 1	Темы 1-3	аналитическая записка	не менее 56%
ОПК-7	Текущий	Моделирование архитектуры и процессов	Лабораторная работа 1-3	Темы 3-5	модель, диаграмма, спецификация	не менее 56%
ПК-4	Текущий	Интеграция компонентов	Лабораторная работа 4-6	Темы 5-7	репозиторий, код, тесты	не менее 56%
УК-1, ОПК-7, ПК-4	Рубежный	Контрольные работы	Аттестационные контрольные работы № 1-3	Темы 1-9	балльная ведомость	не менее 56 баллов
УК-1, ОПК-7, ПК-4	Промежуточный	Экзамен	Вопросы и практическое	1 семестр	экзаменационная ведомость	56-100 баллов

			задание			

2.2. Показатели уровня сформированности компетенций на этапах их формирования, описание шкал оценивания

Таблица 2 связывает показатели сформированности компетенций с контрольными заданиями и шкалами оценивания.

Уровень	Показатели сформированности	Типовые действия обучающегося	Оценочные средства	Компетенции
Высокий	86-100 баллов; полное и аргументированное выполнение задания	Проектирует архитектуру, описывает API, пишет тесты, настраивает CI/CD, объясняет выбор технологий и ограничения	экзамен, проект, лабораторные работы	УК-1, ОПК-7, ПК-4
Повышенный	71-85 баллов; выполнение задания с несущественными ошибками	Разрабатывает основные артефакты, допускает отдельные неточности в документации или тестовом покрытии	контрольная работа, лабораторная работа	УК-1, ОПК-7, ПК-4
Базовый	56-70 баллов; выполнение минимально достаточного набора требований	Описывает требования, реализует базовый прототип, использует Git и тестирование на уровне типового примера	тест, практическая задача	УК-1, ОПК-7, ПК-4
Недостаточный	0-55 баллов; результат не	Не формулирует требования, не объясняет	любой вид контроля	УК-1, ОПК-7, ПК-4

	подтверждает достижение индикаторов	архитектуру, не демонстрирует работоспособный код или тесты		

2.2.1. Описание шкал оценивания

В ФГБОУ ВО «Дагестанский государственный технический университет» применяется модульно-рейтинговая система оценки учебной деятельности. Для текущего контроля и промежуточной аттестации используется четырехуровневая шкала.

Шкала оценивания	**Баллы**	**Критерии оценивания**	**Типичные признаки ответа / работы**
Отлично	86-100	Полный высокий уровень сформированности компетенций	Обучающийся корректно раскрывает понятия жизненного цикла ПО, архитектурных стилей, тестирования, CI/CD, контейнеризации и безопасности; демонстрирует работающий результат и документирует решения.
Хорошо	71-85	Повышенный уровень сформированности компетенций	Ответ полный, но содержит отдельные неточности в терминологии, проектной документации, описании тестовых сценариев или pipeline.
Удовлетворительно	56-70	Базовый уровень сформированности компетенций	Обучающийся воспроизводит основные понятия, выполняет типовое

			практическое задание, но допускает ошибки в архитектурном обосновании, интеграции или тестировании.
Неудовлетворительно	0-55	Компетенции не сформированы	Обучающийся не выполняет существенную часть задания, не объясняет выбор технологий, не предоставляет проверяемый результат или не владеет основными понятиями дисциплины.

Критерии оценки практических заданий: работоспособность решения - до 30 баллов; корректность архитектуры и кода - до 25 баллов; тестирование - до 15 баллов; документация - до 15 баллов; защита решения - до 15 баллов.

3. Типовые контрольные задания, иные материалы и методические рекомендации, необходимые для оценки сформированности компетенций

3.1. Вопросы входного контроля

1. Понятие программного обеспечения и программной системы.
2. Жизненный цикл программного продукта: основные этапы.
3. Основные парадигмы программирования.
4. Назначение системы контроля версий Git.
5. Понятие ветки, коммита и pull request.
6. Реляционная база данных и первичный ключ.
7. HTTP-запрос: метод, URL, заголовки, тело.
8. REST API: ресурс и операция.
9. Назначение unit-тестов.

10. Различия между функциональными и нефункциональными требованиями.

11. Понятие программной архитектуры.

12. Назначение пользовательской истории в Agile-подходе.

13. Основные типы ошибок в программном коде.

14. Понятие непрерывной интеграции.

15. Назначение контейнеризации приложений.

16. Понятие зависимости в программном проекте.

17. Назначение README-файла в репозитории.

18. Понятие технического долга.

19. Назначение статического анализа кода.

20. Основные риски безопасности веб-приложений.

3.2. Вопросы текущего контроля знаний студентов

Аттестационная контрольная работа № 1

1. Модели жизненного цикла ПО: каскадная, итерационная, инкрементальная, Agile.

2. Процессы жизненного цикла ПО по ISO/IEC/IEEE 12207:2017.

3. Сбор и анализ требований к программному продукту.

4. Функциональные и нефункциональные требования: критерии разделения.

5. User story, acceptance criteria, backlog.

6. Моделирование предметной области: сущности, связи, ограничения.

7. Архитектурные стили: монолит, слоистая архитектура, микросервисы.

8. API-first подход и OpenAPI-спецификация.

9. Техническое задание и проектная документация.

10. Пример декомпозиции задачи разработки на этапы.

Аттестационная контрольная работа № 2

1. Git workflow: feature branch, merge request, code review.

2. Практики чистого кода и рефакторинга.

3. Unit-тестирование, интеграционное тестирование, end-to-end тестирование.

4. Test case, test suite, test report.

5. CI/CD pipeline: этапы build, test, package, deploy.

6. GitHub Actions или Azure Pipelines: структура workflow/pipeline.

7. Dockerfile: назначение основных инструкций FROM, WORKDIR, COPY, RUN, CMD.

8. Docker Compose: описание многокомпонентного приложения.

9. Управление зависимостями проекта.

10. Настройка среды разработки и тестового окружения.

Аттестационная контрольная работа № 3

1. Качество программного продукта по ISO/IEC 25010:2023.

2. Метрики качества кода и тестового покрытия.

3. Безопасность разработки: OWASP Top 10:2021.

4. Уязвимости Broken Access Control и Injection: причины и примеры.

5. Секреты, переменные окружения и управление конфигурацией.

6. Логирование, мониторинг и трассировка приложений.

7. Контейнерная оркестрация и Kubernetes.

8. DevOps и DevSecOps: различия в задачах и контролях.

9. Сопровождение и развитие программного продукта.

10. Технический долг и управление изменениями.

Типовые практические задания

1. Составить набор пользовательских историй и критериев приемки для сервиса учета заявок.

2. Построить архитектурную схему программного продукта: клиент, API, база данных, внешние сервисы.

3. Разработать OpenAPI-описание для двух ресурсов REST API.

4. Создать структуру Git-репозитория с README, ветками и pull request-моделью.

5. Написать unit-тесты для функции расчета комиссии или тарифа.

6. Подготовить Dockerfile для простого веб-приложения.

7. Собрать docker-compose.yml для приложения и базы данных.

8. Настроить CI/CD workflow, выполняющий установку зависимостей, тесты и сборку артефакта.

9. Провести code review фрагмента кода и оформить список замечаний.

10. Описать угрозы по двум пунктам OWASP Top 10:2021 и предложить меры контроля.

3.3. Вопросы для проверки остаточных знаний студентов

1. Какие этапы включает жизненный цикл программного продукта?

2. Какие артефакты подтверждают анализ требований?

3. Чем архитектурное решение отличается от технологического выбора?

4. Какие элементы содержит REST API-спецификация?

5. Какие действия выполняет разработчик при code review?

6. Какие тесты применяются для проверки бизнес-логики?

7. Какие операции автоматизируются в CI/CD pipeline?

8. Как контейнеризация влияет на воспроизводимость окружения?

9. Какие характеристики качества включает ISO/IEC 25010:2023?

10. Какие категории угроз входят в OWASP Top 10:2021?

11. Как фиксируются дефекты в процессе тестирования?

12. Какие документы подтверждают готовность программного продукта к поставке?

13. Какие данные должны быть в README проекта?

14. Как оценивается технический долг?

15. Какие критерии используются при приемке практического проекта?

3.4. Задания для промежуточной аттестации (зачета и/или экзамена)

Список вопросов для проведения экзамена

1. Понятие современных технологий разработки программного обеспечения.

2. Жизненный цикл ПО и его процессы по ISO/IEC/IEEE 12207:2017.

3. Каскадная, итерационная и инкрементальная модели разработки.

4. Agile-подходы в разработке ПО: Scrum, Kanban, Lean.

5. Сбор требований к программному продукту.

6. Функциональные требования: признаки, примеры, критерии приемки.

7. Нефункциональные требования: производительность, безопасность, надежность, сопровождаемость.

8. Пользовательские истории и backlog продукта.

9. Моделирование предметной области программной системы.

10. UML и C4 как средства описания программной архитектуры.

11. Монолитная архитектура: преимущества, ограничения, область применения.

12. Микросервисная архитектура: принципы, риски, условия применения.

13. Событийно-ориентированная архитектура и очереди сообщений.

14. Проектирование REST API.

15. OpenAPI Specification как стандарт описания HTTP API.

16. Базы данных в современных программных системах.

17. Реляционные и нереляционные хранилища данных.

18. ORM, миграции и контроль схемы базы данных.

19. Git как система контроля версий.

20. Git workflow: ветвление, merge request, pull request, code review.

21. Стандарты оформления кода и статический анализ.

22. Рефакторинг и управление техническим долгом.

23. Unit-тестирование: цели, структура, ограничения.

24. Интеграционное тестирование и тестирование API.

25. End-to-end тестирование пользовательских сценариев.

26. Тестовая документация: test case, test suite, defect report.

27. Профстандарт «Специалист по тестированию в области информационных технологий»: трудовые функции и связь с дисциплиной.

28. Профстандарт «Программист»: трудовые действия и связь с дисциплиной.

29. CI/CD: цели, этапы и инструменты.

30. GitHub Actions: workflow, job, step, runner.

31. Azure Pipelines: назначение и место в DevOps-процессе.

32. Docker: образ, контейнер, Dockerfile.

33. Docker Compose для локального окружения разработки.

34. Kubernetes как платформа оркестрации контейнеров.

35. Конфигурация приложения: переменные окружения, секреты, профили запуска.

36. Логирование и мониторинг в программных системах.

37. Трассировка запросов в распределенных приложениях.

38. Безопасная разработка ПО: основные принципы.

39. OWASP Top 10:2021 как документ по критическим рискам веб-приложений.

40. Broken Access Control: причины, последствия, меры контроля.

41. Cryptographic Failures: примеры и профилактика.

42. Injection: причины и способы предотвращения.

43. Insecure Design: связь с этапом проектирования.

44. Security Misconfiguration: ошибки конфигурации и контроль.

45. Vulnerable and Outdated Components: управление зависимостями.

46. Software and Data Integrity Failures: контроль целостности поставки.

47. Качество программного продукта по ISO/IEC 25010:2023.

48. Характеристики качества: functional suitability, performance efficiency, compatibility.

49. Характеристики качества: usability, reliability, security.

50. Характеристики качества: maintainability, portability, safety.

51. Документирование программного проекта.

52. README, CHANGELOG, API-документация и архитектурное описание.

53. DevOps и DevSecOps в разработке ПО.

54. Управление релизами и версиями программного продукта.

55. Поставка программного продукта: артефакт, образ, пакет, релиз.

56. Сопровождение программного продукта и обработка инцидентов.

57. Метрики разработки: lead time, deployment frequency, change failure rate, MTTR.

58. Критерии выбора технологии разработки для прикладной информационной системы.

59. Критерии приемки курсового практического проекта.

60. Перспективы развития технологий разработки ПО: платформенная инженерия, low-code, AI-assisted development.

Пример практического задания на экзамене

Разработать структуру проекта веб-сервиса учета заявок: описать требования, предложить архитектуру, подготовить перечень API-методов, указать тестовые сценарии, описать CI/CD pipeline и контейнерное окружение. В ответе должны быть представлены: 1) схема компонентов; 2) таблица требований; 3) фрагмент OpenAPI-описания или таблица эндпоинтов; 4) список тестов; 5) краткая инструкция по запуску.

Экзамен проводится в устной или письменной форме. Практическое задание может выполняться на компьютере или в письменной форме с описанием архитектуры, алгоритма и контрольных процедур.