Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Гриб Владислав Валерьевич
Должность: Ректор
Дата подписания: 06.11.2026 16:26:53
Уникальный программный ключ:
637517d24e103c3db032acf37e879d98ec1c5bb2f5eb89c29abfcd7f43985447

Образовательное частное учреждение высшего образования

«МОСКОВСКИЙ УНИВЕРСИТЕТ ИМЕНИ А.С. ГРИБОЕДОВА»

(ИМПЭ им. А.С. Грибоедова)

ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
(МЕЖДУНАРОДНЫЙ ДИПЛОМАТИЧЕСКИЙ КОЛЛЕДЖ)

УТВЕРЖДАЮ
Директор Международного
дипломатического колледжа
_____ Н.И. Платонова
«16» февраля 2026г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ
Разработка и интеграция модулей программного обеспечения

Специальность
09.02.11 Разработка и управление программным обеспечением

Квалификация выпускника:
Программист

на базе основного общего образования

Форма обучения: очная

Москва

Фонд оценочных средств для профессионального модуля «Разработка и интеграция модулей программного обеспечения». Специальность 09.02.11 Разработка и управление программным обеспечением /– М.: ИМПЭ им. А. С. Грибоедова.

Фонд оценочных средств является неотъемлемой частью рабочей программы профессионального модуля.

Организация-разработчик: Образовательное частное учреждение высшего образования «Московский университет имени А. С. Грибоедова»

Директор Международного дипломатического колледжа_____/Н.И. Платонова

1. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ, ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

Настоящий Фонд оценочных средств (ФОС) для профессионального модуля «Разработка, администрирование и защита баз данных» является неотъемлемым приложением для профессионального модуля «Разработка, администрирование и защита баз данных». На данный ФОС распространяются все реквизиты утверждения, представленные в РПД по данной дисциплине.

2. ПЕРЕЧЕНЬ ОЦЕНОЧНЫХ СРЕДСТВ

Для определения качества освоения обучающимися учебного материала по дисциплине используются следующие оценочные средства:

№ п/п	Оценочное средство	Краткая характеристика оценочного средства	Представление оценочного средства в ФОС
1	Тестирование	Вид контроля, позволяющий оценить изученный теоретический материал.	Вопросы для проведения тестирования
2	Практические задания	Вид контроля, позволяющий оценить умение обучающегося применять осваиваемую компетенцию в практических ситуациях и при решении производственных задач	Задания к практическому занятию
3	Контрольная работа	Вид контроля, позволяющий определить результат освоения компетенций по дисциплине в рамках рассматриваемой темы, оцениваемый с помощью соответствующих индикаторов достижения компетенций	Задания контрольной работы
4	Самостоятельная работа	Вид контроля, позволяющий оценить проработку теоретического материала, изучение рекомендуемой литературы, выполнение практико-ориентированных заданий (заполнение таблиц, проведение сравнительного анализа, составление схем и др.), решение практических задач, создание презентаций, написание рефератов, подборку нормативного и иного материала и выполнение других заданий	Задания самостоятельной работы
5	Курсовая работа	Вид контроля, позволяющий выявить степень владения базовыми знаниями, умениями и навыками, необходимыми для обучения, и определить уровень владения новым материалом	Индивидуальные задания (темы) для курсовой работы
6	Зачет/Зачет с оценкой/Экзамен	Вид контроля, позволяющий выявить степень овладения знаниями, умениями и навыками, необходимыми для дальнейшего освоения образовательной программы подготовки	Вопросы для подготовки к зачету/зачету с оценкой/экзамену

3.ТИПОВЫЕ КОНТРОЛЬНЫЕ ЗАДАНИЯ ИЛИ ИНЫЕ МАТЕРИАЛЫ, НЕОБХОДИМЫЕ ДЛЯ ОЦЕНКИ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

Темы для подготовки к промежуточной аттестации

МДК 02.01. Разработка программных модулей

1. Принципы проектирования программных модулей.
2. Методы анализа требований для определения функциональности модуля.
3. Принципы высококачественных модулей: сильная связность, слабое зацепление, информационное скрытие.
4. Методы декомпозиции сложной задачи на подзадачи и модули.
5. Определение ответственности и границ модуля.
6. Разработка спецификаций модуля: формальные и неформальные подходы.
7. Проектирование интерфейсов модуля (API) и контрактов.
8. Использование предусловий и постусловий в спецификациях.
9. Оптимизация взаимодействия между модулями для сопровождаемости, тестируемости и переиспользования.
10. Применение диаграмм компонентов (UML) для визуализации модульной структуры.
11. Основные принципы ООП: инкапсуляция, наследование, полиморфизм, абстракция.
12. Практическое применение принципов ООП в проектировании модулей.
13. Базовые отношения между классами: ассоциация, агрегация, композиция.
14. Критерии выбора между отношениями «has-a» (агрегация/композиция) и «is-a» (наследование).
15. Абстрактные классы и интерфейсы: сравнительный анализ и применение.
16. Контрактно-ориентированное программирование.
17. Принципы проектирования SOLID.
18. Применение диаграмм классов UML для визуализации и проектирования.
19. Модульная архитектура: принципы, преимущества и ограничения.
20. Примеры архитектурных стилей: монолитные и микросервисные приложения.
21. Системы контроля версий: принципы работы Git и branching-стратегии.
22. Системы сборки и менеджеры зависимостей: Maven, Gradle, NuGet.
23. Использование сторонних библиотек: оценка, внедрение, управление версиями.
24. Создание и публикация собственных библиотек: сборка, версионирование, распространение.
25. Архитектурные шаблоны высокого уровня: MVC, MVP, MVVM (концептуальное сравнение).
26. Классификация паттернов проектирования: порождающие, структурные, поведенческие.
27. Взаимосвязь паттернов проектирования с отношениями между классами (наследование, композиция и др.).
28. Модели асинхронного программирования: Callback, Promises/Futures, async/await.
29. Управление потоками: пулы потоков (Thread Pool) и задачи (Task).
30. Механизмы отмены асинхронных и длительных операций.

31. Синхронизация доступа к общим ресурсам: блокировки, семафоры, мьютексы.
32. Проблемы параллельного программирования: состояние гонки (race condition) и взаимная блокировка (deadlock).
33. Паттерны параллельного программирования: Producer-Consumer, Fork-Join и другие.
34. Применение лямбда-выражений и функциональных интерфейсов в асинхронных операциях.
35. Оценка сложности алгоритмов: понятие асимптотической оценки.
36. Большие О-нотации: анализ временной и пространственной сложности.
37. Анализ алгоритмов: худший, лучший и средний случаи.
38. Амортизационный анализ сложности операций.
39. Базовые структуры данных: массивы (статические и динамические).
40. Линейные структуры данных: связные списки, стек, очередь, дек.
41. Алгоритмы поиска: линейный и бинарный поиск, их применение.
42. Алгоритмы сортировки: от простых (пузырьковая, вставками) до эффективных (быстрая, слиянием).
43. Хеш-таблицы: принцип работы, хеш-функции, разрешение коллизий.
44. Древовидные структуры: бинарные деревья, деревья поиска, сбалансированные деревья (AVL, красно-черные).
45. Графы: представление (матрица, список смежности), алгоритмы обхода (DFS, BFS). 46. Расширенные алгоритмы на графах: поиск кратчайшего пути (Дейкстра, Флойд 47. Алгоритмы построения минимального остовного дерева: Крускала и Прима.
48. Методы разработки алгоритмов: "разделяй и властвуй", жадные алгоритмы, динамическое программирование.
49. Алгоритмы работы со строками: поиск подстроки (Кнута-Морриса-Пратта, БойераМура).
50. Специализированные структуры данных: B-деревья, B+-деревья, бинарные кучи и приоритетные очереди.

МДК.02.02 Поддержка и тестирование программных модулей

1. Внутреннее и внешнее качество ПО: сравнительный анализ.
2. Стандарт ISO/IEC 25010: Модель качества программного обеспечения.
3. Практическое применение и интерпретация метрик цикломатической сложности.
4. Оценка уровня связности модуля: от функциональной до случайной.
5. Оценка уровня зацепления модулей: от зацепления по данным до зацепления по содержимому.
6. Динамические метрики и их связь с надежностью и производительностью системы.
7. Принцип единственной ответственности как основа высокой связности.
8. Принцип инверсии зависимостей как основа слабого зацепления.
9. Инструменты оценки качества: от линтеров до платформ типа SonarQube.
10. Паттерны рефакторинга как средство устранения «запахов кода».
11. Профилирование как метод поиска «узких мест» в модуле.
12. Классификация дефектов по степени критичности и приоритету исправления.
13. Отладка как научный метод: гипотеза, эксперимент, анализ.
14. Современные возможности IDE для отладки (условные точки, трассировка).

15. Поиск и локализация «плавающих» дефектов.
16. Стратегии локализации ошибок: бинарный поиск, анализ симптомов.
17. Работа с системами отслеживания ошибок: жизненный цикл дефекта.
18. Процесс управления инцидентами и проблемами.
19. Архитектурные подходы к обработке ошибок и исключений.
20. Построение иерархии пользовательских исключений.
21. Шаблоны обработки исключений: специфичный перехват, проброс, трансформация.
22. Логирование в отладке: выбор уровня детализации и точек вставки логов.
23. Корреляция логов и трассировок для анализа распределенных сбоев.
24. Влияние механизма исключений на производительность.
25. Best practices документирования и воспроизведения дефектов.
26. Планирование тестирования: оценка усилий, определение стратегии.
27. Модульное тестирование: использование mock-объектов и заглушек.
28. Подходы к интеграционному тестированию: Big Bang, incremental, sandwich.
29. Системное тестирование: проверка нефункциональных требований.
30. Приемочное тестирование пользователем (UAT) и операционное (OAT).
31. Техники тестирования черного ящика: таблицы решений, анализ причинноследственных связей.
32. Критерии покрытия кода в тестировании белого ящика.
33. Проектирование тестов для нефункциональных требований.
34. Инструменты для тестирования безопасности (OWASP ZAP, Burp Suite).
35. Методологии нагрузочного тестирования: планирование, выполнение, анализ.
36. Тестирование совместимости на различных платформах.
37. Экономическое обоснование автоматизации тестирования (ROI).
38. Паттерны проектирования автоматизированных тестов (Page Object, Screenplay).
39. Интеграция тестов в CI/CD: пайплайны, gate, отчетность.
40. Пирамида тестирования как оптимальная стратегия автоматизации.
41. Методы разработки тестовых случаев: таблицы решений, диаграммы переходов состояний.
42. Анализ рисков как основа для определения приоритетов тестирования.
43. Инструменты для тестирования API (Postman, REST Assured, SoapUI).
44. Стратегии тестирования контейнеризованных и serverless-приложений.
45. Особенности тестирования в Agile и DevOps-методологиях.
46. Виды работ по сопровождению ПО: корректирующее, адаптивное, совершенствующее, профилактическое.
47. Ревью кода: формальные и неформальные подходы, инструменты (GitHub PR, Gerrit).
48. Стандарты внутренней документации: соглашения об именовании, комментарии, архитектурные решения (ADR).
49. Создание и поддержание тестовой документации: тест-план, тест-кейс, чек-лист, отчет.
50. Современные подходы к документации: living documentation, doc-as-code, автоматическая генерация.

МДК.02.03 Технология разработки программных модулей

1. Системы статического анализа кода: принцип работы, цели и примеры инструментов.
2. Интеграция статических анализаторов в процесс разработки: проверка стиля, качества, уязвимостей и соблюдения принципов.
3. Инструменты UML для проектирования: классификация (визуальные редакторы, генераторы кода) и примеры.
4. Современные среды разработки как платформы для анализа: встроенные рефакторинги, визуализаторы зависимостей, подсказки по сложности.
5. Интерактивные среды для анализа алгоритмов: назначение, возможности для обучения и отладки.
6. Библиотеки для работы со сложными структурами данных: обзор и выбор для конкретных задач.
7. Профайлеры для оценки алгоритмической сложности и производительности в runtime: принцип профилирования, типы профайлеров (инструментирующие, семплирующие).
8. ORM-фреймворки и микро-ORM: сравнительный анализ подходов, примеры.
9. Инструменты для построения, анализа и оптимизации SQL-запросов: встроенные в IDE, независимые клиенты, анализаторы планов выполнения.
10. Системы управления миграциями базы данных: принцип версионирования схемы, обзор инструментов.
11. Профайлеры запросов к БД и APM-системы: мониторинг производительности на уровне запросов и транзакций.
12. Обзор GUI-фреймворков для создания декстопных и веб-приложений: критерии выбора, семейства.
13. Концепция привязки данных и шаблон MVVM: архитектурные преимущества, реализация в различных фреймворках.
14. Средства стилизации UI: от CSS-препроцессоров и CSS-in-JS для веба до систем ресурсов и стилей в нативных фреймворках.
15. Технологии создания API для межмодульного взаимодействия: сравнительный анализ REST, gRPC, GraphQL по критериям производительности, типизации и гибкости.
16. Инструменты для автоматического документирования API: подходы на основе кода и аннотаций.
17. Клиенты для тестирования API на этапе разработки и интеграции: функционал и использование.
18. Фреймворки для модульного тестирования GUI: подходы к изоляции и симуляции интерфейса.
19. Инструменты для автоматизации End-to-End (E2E) сценариев для GUI: классификация.
20. Методы тестирования асинхронных и событийно-ориентированных интерфейсов: моки таймеров, симуляторы событий, тестирование с ожиданиями.
21. Статические анализаторы безопасности кода: интеграция в CI/CD для поиска уязвимостей.
22. Принципы безопасного кодирования для модулей: OWASP рекомендации, защита от инъекций, работа с чувствительными данными.

23. Системы управления зависимостями и сборкой для разных языков: общая логика работы.
24. Стратегии управления версиями модулей и библиотек: семантическое версионирование (SemVer), совместимость, dependency hell.
25. Инструменты контейнеризации для изоляции среды выполнения модуля: Docker как стандарт, создание и использование Docker-образов.
26. Принципы CI/CD применительно к модульной разработке: цели непрерывной интеграции и доставки для независимых модулей.
27. Обзор платформ для реализации CI/CD пайплайнов: облачные и self-hosted решения.
28. Этапы типового CI/CD пайплайна для модуля: от коммита к артефакту.
29. Публикация артефактов: использование артефакт-репозиториев.
30. Мониторинг и observability модулей в production: APM-системы, сбор метрик, логирование и распределенная трассировка.

Темы для самостоятельной работы обучающихся

МДК 02.01. Разработка программных модулей

1. Архитектурные подходы к доступу к данным: Data Access Layer (DAL).
2. Паттерн Repository: абстракция над хранилищем данных.
3. Паттерн Unit of Work: управление транзакциями и целостностью.
4. Сравнение подходов Active Record и Data Mapper.
5. Архитектурный паттерн CQRS (Command Query Responsibility Segregation).
6. Технологии доступа к данным: ADO.NET, Entity Framework Core, Dapper, другие ORM.
7. Управление подключениями к БД: connection strings, пулы соединений, мониторинг.
8. Конфигурация ORM: маппинг сущностей (Fluent API, атрибуты).
9. Реализация CRUD-операций и выполнение запросов.
10. Управление транзакциями: уровни изоляции, распределенные транзакции.
11. Стратегии управления конкурентным доступом: оптимистическая и пессимистическая блокировка.
12. Оптимизация работы с БД: проблема N+1 и методы ее решения.
13. Индексы в базах данных: создание, анализ, покрывающие индексы.
14. Анализ планов выполнения запросов (Query Execution Plans).
15. Кэширование данных: кэширование запросов первого и второго уровня.
16. Принципы безопасного программирования для модулей данных.
17. Безопасность на уровне БД: принцип наименьших привилегий, шифрование данных.
18. Обеспечение конфиденциальности: маскирование данных в логах и отладочной информации.
19. Аудит изменений данных: механизмы отслеживания «кто, когда, что изменил».
20. Управление миграциями базы данных: инструменты и стратегии деплоя.
21. Виды пользовательских интерфейсов: командная строка, графический, речевой.
22. Принципы и этапы разработки графического пользовательского интерфейса (GUI).
23. Технологии и инструменты для разработки GUI.
24. Компоненты GUI: типы элементов управления, компоновка (layout).
25. Событийно-ориентированное программирование в GUI: события и обработчики.

26. Работа с окнами: создание, управление жизненным циклом, взаимодействие между окнами.
27. Валидация пользовательского ввода: методы, регулярные выражения, обратная связь.
28. Многопоточность в GUI-приложениях: проблемы и паттерны (например, BackgroundWorker).
29. Основы UX/UI дизайна: теория цвета, работа со шрифтами, стилизация.
30. Визуализация данных в GUI: построение графиков и диаграмм с использованием специализированных библиотек.
31. Основные нефункциональные требования: безопасность, производительность, масштабируемость.
32. Принципы безопасного программирования на уровне модулей.
33. Криптографические примитивы и их применение в модулях.
34. Методология оптимизации производительности: профилирование, анализ узких мест.
35. Алгоритмическая оптимизация: выбор структур данных, учет кэш-локальности.
36. Оптимизационные техники: мемоизация, кэширование, ленивые вычисления.
37. Оптимизация операций ввода-вывода и сетевого взаимодействия.
38. Принципы проектирования масштабируемых систем: горизонтальное и вертикальное масштабирование.
39. Паттерны масштабируемости: шардирование, репликация, кэширование.
40. Принципы построения отказоустойчивых систем: idempotency, retry, circuit breaker.
41. Системы мониторинга и метрики: Prometheus, Grafana, бизнес-метрики.
42. Централизованное логирование: стеки ELK, Loki, анализ логов.
43. Распределенная трассировка (Distributed Tracing): OpenTelemetry, Jaeger.
44. Проактивный мониторинг и алертинг.
45. Принципы энергоэффективного («зеленого») программирования.
46. Управление затратами в облачных средах: оптимизация использования ресурсов.
47. Современные облачные практики: serverless, containers, infrastructure as code.
48. Стратегии развертывания: blue-green deployments, canary releases, zero-downtime миграции.
49. Наблюдаемость как ключевая характеристика современных распределенных систем.
50. Управление конфигурацией и секретами в облачных и гибридных средах.

МДК.02.02 Поддержка и тестирование программных модулей

1. Определение и сущность качества программного модуля.
2. Международные стандарты качества программного обеспечения (ISO 25010 и др.).
3. Статические метрики объема кода: количество строк кода и его ограничения.
4. Метрика цикломатической сложности Маккейба.
5. Метрики связности и их классификация.
6. Метрики зацепления и их классификация.
7. Динамические метрики качества: покрытие кода тестами, частота отказов, время отклика.
8. Принципы проектирования, влияющие на качество модулей (SOLID, DRY, KISS).
9. Инструменты статического анализа кода (линтеры) и их роль.
10. Понятие «запахов кода» и их связь с качеством.

11. Роль рефакторинга в поддержании высокого качества кодовой базы.
12. Инструменты профилирования для анализа производительности модуля.
13. Разграничение понятий: ошибка, дефект, сбой.
14. Основные типы программных ошибок: синтаксические, логические, ошибки времени выполнения.
15. Процесс и философия отладки программного обеспечения.
16. Функциональные возможности современных инструментов отладки (дебаггеров).
17. Техники пошаговой отладки: точки останова, шаг за шагом, просмотр состояния.
18. Стратегии поиска ошибок: «разделяй и властвуй», обратное прослеживание.
19. Системы отслеживания ошибок (Bug Tracking Systems): назначение и базовые функции.
20. Документирование процесса отладки и отчетности об ошибках.
21. Роль и классификация исключительных ситуаций в программировании.
22. Различия между проверяемыми (checked) и непроверяемыми (unchecked) исключениями.
23. Синтаксические конструкции для управления исключениями (try-catch-finally, throw/throws).
24. Стратегии работы с исключениями: перехват, проброс, трансляция, логирование.
25. Использование логирования (logging) как инструмента отладки и анализа.
26. Цели и основные этапы процесса тестирования программного обеспечения.
27. Сравнение ручного и автоматизированного тестирования.
28. Модульное тестирование: цели и объекты тестирования.
29. Интеграционное тестирование: цели и подходы.
30. Системное тестирование и его место в жизненном цикле.
31. Приемочное тестирование и его формы.
32. Метод тестирования черного ящика (Black-box Testing): принципы и техники.
33. Метод тестирования белого ящика (White-box Testing): принципы и критерии покрытия.
34. Виды нефункционального тестирования: производительность, безопасность, удобство использования.
35. Тестирование производительности: основные виды.
36. Тестирование удобства использования и его критерии.
37. Нагрузочное и стресс-тестирование: сравнительная характеристика.
38. Преимущества и недостатки автоматизации тестирования.
39. Инструменты для автоматизации функционального тестирования (Selenium, TestComplete).
40. Интеграция автоматизированных тестов в процесс разработки (CI/CD).
41. Методы разработки тестовых случаев: анализ граничных значений, эквивалентное разделение.
42. Критерии покрытия тестами: операторы, ветви, условия, пути.
43. Выбор инструментов для разработки и выполнения тестов.
44. Особенности тестирования мобильных и облачных приложений.
45. Тестирование микросервисной архитектуры: специфические задачи и подходы.
46. Работы, выполняемые на этапе сопровождения программного обеспечения.
47. Процесс ревьюирования кода как практика повышения качества.
48. Стандарты разработки и оформления документации на ПО.
49. Виды тестовой документации: планы, спецификации, сценарии, отчеты.

50. Инструменты для создания технической документации и комментирования кода.

МДК.02.03 Технология разработки программных модулей

1. Настройка и кастомизация правил статического анализатора кода для проверки принципов SOLID и code smells.
2. Автоматическая генерация UML-диаграмм (классов, последовательности) из существующего кода и реверс-инжиниринг.
3. Сравнение возможностей рефакторинга в различных IDE и их влияние на производительность.
4. Анализ алгоритмов с помощью Big O на практике: использование профайлера для верификации теоретической оценки.
5. Реализация нестандартных структур данных с использованием базовых библиотек языка.
6. Сравнение производительности микро-ORM и нативного SQL в высоконагруженных сценариях: написание и профилирование бенчмарков.
7. Продвинутые техники оптимизации SQL-запросов с использованием инструментов анализа планов выполнении.
8. Разработка и отладка сложных миграций БД с поддержкой отката в распределенной системе.
9. Выбор GUI-фреймворка для кроссплатформенного десктопного приложения: сравнительный анализ Qt, .NET MAUI, Electron.
10. Реализация паттерна MVVM «с нуля» и сравнение со встроенной поддержкой во фреймворках.
11. Создание системы дизайн-токенов и тем оформления для переиспользуемых UIкомпонентов.
12. Проектирование API-шлюза для агрегации данных из нескольких микросервисов с использованием GraphQL.
13. Автоматизация поддержки актуальности Swagger-документации на основе аннотаций в коде.
14. Разработка коллекции автотестов для REST API и их интеграция в CI.
15. Тестирование компонентов с использованием библиотек: подходы к мокированию и проверке состояния.
16. Написание устойчивых E2E-тестов с использованием паттерна Page Object.
17. Стратегии тестирования WebSocket-соединений и real-time обновлений интерфейса.
18. Интеграция визуального регрессионного тестирования в пайплайн для UI-модулей.
19. Проведение аудита безопасности зависимостей проекта помощью OWASP DependencyCheck или Snyk.
20. Реализация модуля с соблюдением принципов безопасности: хеширование паролей, безопасные заголовки HTTP, валидация входных данных.
21. Разрешение конфликтов версий зависимостей в крупном монорепозитории с множеством модулей.
22. Создание и публикация пакета (библиотеки) в публичный реестр с соблюдением SemVer.

23. Оптимизация Docker-образа для production-модуля: многоэтапная сборка, уменьшение размера, безопасность.
24. Проектирование отказоустойчивого CI/CD пайплайна для модуля с этапами кэширования зависимостей и параллельным выполнением тестов.
25. Автоматизация развертывания (CD) модуля в облачную среду из артефактрепозитория.
26. Сбор и анализ пользовательских метрик производительности модуля в Prometheus.
27. Настройка алертинга на основе метрик и логов для критически важных модулей системы.
28. Практика «Infrastructure as Code» для модуля: описание среды развертывания.
29. Реализация стратегии Blue-Green Deployment или Canary Release для модуля с минимальным временем простоя.
30. Организация мониторинга технического долга модуля через метрики качества кода и регулярный аудит.

Примерные варианты тестовых оценочных заданий (ТОЗ) для контрольного рубежа в рамках текущего контроля

МДК 02.01. Разработка программных модулей

Тестовое задание	Оцениваемые компетенции
ВОПРОСЫ ОТКРЫТОГО ТИПА	
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Степень взаимозависимости и связанности элементов внутри одного модуля называется _____. +: связность	ПК 2.1.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Мера зависимости одного модуля от внутреннего устройства других модулей называется _____. +: зацепление	ПК 2.1.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Принцип, при котором детали реализации модуля скрыты от внешнего мира и доступен только четко определенный интерфейс, называется _____. +: инкапсуляция	ПК 2.1.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Набор из пяти принципов объектно-ориентированного проектирования, способствующих созданию понятного и гибкого кода, известен как принципы _____. +: SOLID	ПК 2.1.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Принцип, гласящий, что класс должен иметь только одну причину для изменения, называется принципом _____. +: единственной ответственности	ПК 2.1.

(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Паттерн проектирования, который определяет семейство алгоритмов, инкапсулирует каждый из них и делает их взаимозаменяемыми, называется _____. +: Стратегия	ПК 2.1.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Структура данных, в которой добавление и удаление элементов происходит по принципу "последним пришел – первым вышел" (LIFO), называется _____. +: стек	ПК 2.1.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Эффективная структура данных для поиска, вставки и удаления, основанная на хеш-функции, называется _____. +: хеш-таблица	ПК 2.1.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Рекурсивный метод проектирования алгоритмов, разбивающий задачу на подзадачи того же типа, называется _____. +: "разделяй и властвуй"	ПК 2.1.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Метод проектирования алгоритмов, при котором задача разбивается на перекрывающиеся подзадачи, результаты которых запоминаются и повторно используются, называется _____. +: динамическое программирование	ПК 2.1.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Паттерн слоя доступа к данным, который абстрагирует набор объектов и операции над ними, имитируя коллекцию в памяти, называется _____. +: Репозиторий	ПК 2.2.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Паттерн для группировки набора операций с базой данных в единую транзакцию, чтобы все они либо завершились успешно, либо были отменены, называется _____. +: Единица работы	ПК 2.2.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Архитектурный подход, разделяющий операции чтения данных (Query) и операции изменения данных (Command), называется _____. +: CQRS	ПК 2.2.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Технология, которая автоматически связывает объекты приложения с записями в реляционной базе данных, называется _____. +: ORM (Object-Relational Mapping)	ПК 2.2.

(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Механизм, при котором изменения в графическом интерфейсе автоматически отражают изменения в объекте данных, и наоборот, называется _____. +: привязка данных	ПК 2.2.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Состояние, при котором два или более потока одновременно обращаются к общим данным и как минимум один из них изменяет их, называется _____. +: состояние гонки	ПК 2.2.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Ситуация, в которой два или более потока взаимно блокируют друг друга, ожидая освобождения ресурсов, называется _____. +: взаимная блокировка	ПК 2.2.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Объект, используемый для синхронизации потоков и ограничивающий количество потоков, которые могут одновременно получить доступ к ресурсу, называется _____. +: семафор	ПК 2.2.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Принцип безопасного программирования, согласно которому пользователю или процессу должны предоставляться минимально	ПК 2.2.

необходимые привилегии, называется принципом _____. +: наименьших привилегий	
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Стратегия развертывания, при которой работают два идентичных производственных окружения ("синее" и "зеленое"), а трафик переключается между ними, называется _____ развертывание. +: сине-зеленое	ПК 2.2.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Архитектурный шаблон, производный от MVC, где роль контроллера играет Посредник (Presenter), тесно связанный с Представлением, называется _____. +: MVP	ПК 2.2.
ВОПРОСЫ ЗАКРЫТОГО ТИПА	
Какой принцип SOLID гласит: "Программные сущности должны быть открыты для расширения, но закрыты для модификации"? +: Принцип открытости/закрытости -: Принцип единственной ответственности -: Принцип подстановки Барбары Лисков -: Принцип инверсии зависимостей	ПК 2.1.

Какой принцип объектно-ориентированного программирования позволяет трактовать объекты разных классов как объекты общего родительского класса? +: Полиморфизм -: Инкапсуляция -: Абстракция -: Наследование	ПК 2.2.
Какой тип связи между классами предполагает, что один объект является частью другого, но может существовать самостоятельно? +: Агрегация -: Композиция -: Наследование -: Реализация	ПК 2.2.
Какая структура данных работает по принципу "первым пришел, первым вышел" (FIFO)? -: Стек (Stack) -: Куча (Heap) +: Очередь (Queue) -: Связный список (Linked List)	ПК 2.2.
Какой архитектурный шаблон четко разделяет ответственность за данные, отображение и логику управления? -: MVP (Model-View-Presenter) -: MVVM (Model-View-ViewModel) +: MVC (Model-View-Controller) -: Сервис-ориентированная архитектура (SOA)	ПК 2.2.
Какая уязвимость возникает при выполнении непроверенных пользовательских данных в качестве части команды СУБД? -: XSS (Межсайтовый скриптинг)	ПК 2.1.
+: SQL-инъекция -: Race condition (Состояние гонки) -: Переполнение буфера	
Какой инструмент используется для описания и визуализации статической структуры классов в системе? -: Диаграмма последовательности UML -: Диаграмма развертывания UML +: Диаграмма классов UML -: Диаграмма деятельности UML	ПК 2.2.

Какое отношение между классами в UML является наиболее сильным, где часть не может существовать без целого? +: Композиция -: Агрегация -: Ассоциация -: Зависимость	ПК 2.2.
Какой паттерн слоя доступа к данным используется для группировки набора операций в одну бизнес-транзакцию, чтобы все они либо завершились успешно, либо были отменены? -: Репозиторий (Repository) -: Data Mapper +: Единица работы (Unit of Work) -: Активная запись (Active Record)	ПК 2.2.

МДК.02.02 Поддержка и тестирование программных модулей

Тестовое задание	Оцениваемые компетенции
ВОПРОСЫ ОТКРЫТОГО ТИПА	
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Нормативный стандарт, определяющий модель качества программного обеспечения, известен как _____. +: ISO 25010	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Метрика, измеряющая количество линейно независимых путей в графе потока управления программы, называется _____. +: цикломатическая сложность	ПК 2.4
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Мера зависимости одного модуля от деталей реализации других модулей называется _____. +: зацепление	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Динамическая метрика, показывающая, какая часть исходного кода была выполнена во время тестов, называется _____. +: покрытие кода	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Процесс поиска, анализа и устранения причин дефектов в программном	ПК 2.4.
обеспечении называется _____. +: отладка	

(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Механизм обработки аномальных или исключительных ситуаций во время выполнения программы называется _____. +: исключение	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Результат ошибки программиста, приводящий к некорректному состоянию программы, называется _____. +: дефект	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Программная система для регистрации, отслеживания статуса и управления жизненным циклом дефектов называется _____. +: система отслеживания ошибок	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Вид тестирования, при котором тестировщик не имеет доступа к исходному коду и внутренней структуре системы, называется тестированием _____. +: черного ящика	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Вид тестирования, оценивающий удобство и простоту использования интерфейса системы для конечного пользователя, называется _____. +: тестирование удобства использования	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Процесс проверки кода другими разработчиками на предмет соответствия стандартам и выявления дефектов называется _____. +: ревью кода	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Процесс изменения внутренней структуры кода без изменения его внешнего поведения с целью улучшения читаемости и сопровождаемости называется _____. +: рефакторинг	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Метод разработки тестовых случаев, при котором входные данные разбиваются на классы эквивалентности, называется _____. +: эквивалентное разделение	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Техника тестирования, при которой тестовые данные выбираются на границах допустимых диапазонов, называется _____. +: анализ граничных значений	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Вид тестирования, имитирующий ожидаемую или пиковую нагрузку на систему для оценки ее поведения, называется _____. +: нагрузочное тестирование	ПК 2.4.

(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Критерий покрытия тестами, требующий выполнения каждой ветви (как true, так и false) каждого условия в коде, называется покрытием _____. +: условий	ПК 2.4.

(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Формальное описание шагов, данных и ожидаемых результатов для проверки конкретной функциональности, называется _____. +: тестовый случай	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Документ, фиксирующий обнаруженный дефект с описанием шагов для его воспроизведения, называется _____. +: отчет об ошибке	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Стратегия поиска ошибки, при которой программа исследуется от точки сбоя назад к его причине, называется _____. +: обратное прослеживание	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Принцип, согласно которому модуль должен быть открыт для расширения, но закрыт для модификации, называется принципом _____. +: открытости/закрытости	ПК 2.4.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Метод оптимизации, при котором результаты дорогостоящих вызовов функций сохраняются и переиспользуются, называется _____. +: мемоизация	ПК 2.4.
ВОПРОСЫ ЗАКРЫТОГО ТИПА	
Какой нормативный стандарт определяет модель качества программного обеспечения, включающую характеристики функциональности, надежности и сопровождаемости? -: ISO 9001 -: IEEE 829 +: ISO 25010 -: SWEBOK	ПК 2.4.
Какая метрика измеряет количество линейно независимых путей через исходный код модуля? -: Количество строк кода (SLOC) +: Цикломатическая сложность (Cyclomatic Complexity) -: Глубина наследования (Depth of Inheritance) -: Коэффициент связности (Cohesion)	ПК 2.4.

Как называется мера зависимости одного модуля от внутреннего устройства других модулей? -: Связность -: Инкапсуляция +: Зацепление -: Сложность	ПК 2.4.
Как называется процесс локализации, анализа и устранения причин дефектов в программе? -: Тестирование -: Верификация -: Валидация +: Отладка	ПК 2.4.
Как называется результат ошибки программиста, приводящий к некорректному состоянию программы? +: Дефект -: Отказ -: Инцидент -: Сбой	ПК 2.4.
Как называется механизм обработки аномальных ситуаций во время выполнения программы? -: Ассерция +: Исключение -: Код возврата -: Логирование	ПК 2.4.
Как называется вид тестирования, при котором тестировщик не имеет доступа к исходному коду системы? -: Тестирование белого ящика (White-box) +: Тестирование черного ящика (Black-box) -: Модульное тестирование (Unit Testing) -: Статическое тестирование (Static Testing)	ПК 2.4.
Как называется процесс проверки кода другими разработчиками на предмет соответствия стандартам? -: Рефакторинг -: Инспекция +: Ревью кода -: Валидация кода	ПК 2.4.

Как называется тестирование, имитирующее ожидаемую или пиковую нагрузку на систему? -: Стресс-тестирование (Stress Testing) -: Объемное тестирование (Volume Testing) +: Нагрузочное тестирование (Load Testing) -: Тестирование стабильности (Soak Testing)	ПК 2.4.

МДК.02.03 Технология разработки программных модулей

Тестовое задание	Оцениваемые компетенции
ВОПРОСЫ ОТКРЫТОГО ТИПА	
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Инструмент, автоматически проверяющий стиль кода, выявляющий потенциальные ошибки и уязвимости, называется _____ анализатор кода. +: статический	ОК 01
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Фреймворк, который автоматически отображает объекты приложения на записи в реляционной базе данных, называется _____. +: ORM	ОК 01
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Архитектурный шаблон, в котором Модель Представления	ОК 01

(ViewModel) связывает Модель данных с Представлением через привязку данных, называется _____. +: MVVM	
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Программный интерфейс, использующий протокол HTTP и стандартные методы (GET, POST) для взаимодействия между модулями, называется _____ API. +: REST	ОК 01
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Практика автоматической сборки и тестирования кода при каждом изменении в репозитории называется Непрерывная _____. +: интеграция	ОК 02
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Инструмент для визуального проектирования, позволяющий создавать диаграммы классов и последовательностей, называется _____ редактор. +: UML	ОК 02

(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Практика описания серверной инфраструктуры (серверы, сети) с помощью конфигурационных файлов называется «Инфраструктура как _____». +: код	ОК 02
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Инструмент для отслеживания производительности приложения в реальном времени, показывающий медленные запросы и узкие места, называется _____ система. +: APM	ОК 02
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Технология, которая автоматически связывает свойства данных в коде с элементами пользовательского интерфейса, называется _____ данных. +: привязка	ОК 02
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Специализированные анализаторы, ищущие в коде известные уязвимости (например, из списка OWASP), называются _____ анализаторы безопасности. +: статические	ПК 2.1.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Принцип, при котором код, данные и конфигурация модуля упаковываются вместе в единый развертываемый образ, называется _____. +: контейнеризация	ПК 2.1.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Подход к разработке, объединяющий практики разработки и эксплуатации для повышения скорости выпуска релизов, называется _____. +: DevOps	ПК 2.1.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ)	ПК 2.2.

Инструмент для автоматического форматирования кода в соответствии с заданным стилем называется _____. +: линтер	
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Протокол и система для отслеживания изменений в исходном коде называется система контроля _____. +: версий	ПК 2.2.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Программа, которая транслирует исходный код в машинный код или промежуточное представление, называется _____. +: компилятор	ПК 2.2.

(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Процесс улучшения внутренней структуры кода без изменения его внешнего поведения называется _____. +: рефакторинг	ПК 2.2.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Специальная программа для пошагового выполнения кода, установки точек останова и просмотра переменных, называется _____. +: отладчик	ПК 2.2.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Язык запросов для работы с данными в реляционных базах данных называется _____. +: SQL	ПК 2.2.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Фундаментальный протокол глобальной сети, отвечающий за адресацию и маршрутизацию пакетов данных между компьютерами, называется _____. +: IP	ПК 2.3.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Система для хранения собранных артефактов (библиотек, Dockerобразов) называется _____ репозиторий. +: артефакт	ПК 2.3.
(ВПИСАТЬ ПРАВИЛЬНЫЙ ОТВЕТ) Инструмент, который выполняет написанные тестовые сценарии, имитируя действия пользователя в браузере, называется _____ для E2E-тестирования. +: Selenium	ПК 2.3.
ВОПРОСЫ ЗАКРЫТОГО ТИПА	
Какой инструмент является стандартом де-факто для автоматизации тестирования веб-интерфейсов? -: TestComplete -: JMeter -: Postman +: Selenium	ПК 2.4.
Какой архитектурный шаблон предполагает наличие Модели Представления (ViewModel) для связывания данных и интерфейса? -: Model-View-Controller (MVC) -: Model-View-Presenter (MVP) +: Model-View-ViewModel (MVVM) -: Service-Oriented Architecture (SOA)	ПК 2.4.

Какой популярный клиент используется разработчиками для ручного тестирования и документирования API? -: Swagger UI -: cURL +: Postman -: Jenkins	ПК 2.4.
Какой инструмент для контейнеризации позволяет упаковать приложение со всеми зависимостями в переносимый образ? -: Kubernetes +: Docker -: Vagrant -: Terraform	ПК 2.4.
Какой инструмент используется для визуального проектирования диаграмм классов, компонентов и последовательностей? +: UML-редактор -: Система контроля версий -: Интегрированная среда разработки -: Профайлер	ПК 2.4.
Какой тип анализа кода выполняется без запуска программы для поиска потенциальных ошибок и проблем с качеством? -: Динамический анализ (Dynamic Analysis) +: Статический анализ (Static Analysis) -: Профилирование (Profiling) -: Модульное тестирование (Unit Testing)	ПК 2.4.
Какой подход к разработке объединяет практики разработки (Dev) и эксплуатации (Ops) для ускорения выпуска релизов? -: Agile -: Scrum +: DevOps -: Waterfall	ПК 2.4.
Какой принцип ООП позволяет классу-наследнику использовать или переопределять методы и свойства родительского класса? -: Инкапсуляция -: Полиморфизм +: Наследование -: Абстракция	ПК 2.4.

Какой формат используется gRPC по умолчанию для сериализации структурированных данных? -: JSON -: XML +: Protocol Buffers (protobuf) -: YAML	ПК 2.4.

Шкала оценивания тестовых заданий

Данный вид контроля, рассчитан на выявление уровня усвоения теоретического и практического материала в рамках изучения дисциплин. Критерии оценивания теста: 1 вопрос = 1 балл.

Оценка «5»	30-27 Баллов
Оценка «4»	26-23 Баллов
Оценка «3»	22-18 Баллов
Оценка «2»	0-17 Баллов