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

УТВЕРЖДАЮ
И.о. директора института
перспективной инженерии,
канд. техн. наук, доцент
А.А. Порохня

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ
ДЛЯ ПРОВЕДЕНИЯ ГОСУДАРСТВЕННОЙ ИТОГОВОЙ АТТЕСТАЦИИ

Направление подготовки	09.03.04 Программная инженерия
Направленность (профиль)	Разработка и сопровождение программного обеспечения
Форма обучения	Очная
Год начала обучения	2026 г.

Реализуется в 8 семестре

СОГЛАСОВАНО:
Представитель работодателя
ООО «Стилсофт»
_____И.В. Свистунов

РАЗРАБОТАНО:
Зав. межинститутской базовой кафедрой
_____Новикова Е.Н.

Предисловие

Назначение. Фонд оценочных средств для государственной итоговой аттестации предназначен для установления уровня профессиональной подготовки выпускников направления подготовки 09.03.04 «Программная инженерия», направленность (профиль) «Инженерия сложных программных систем», оценки качества освоения ОП ВО и степени обладания выпускниками необходимыми универсальными, общепрофессиональными и профессиональными компетенциями (УК-1, УК-2, УК-3, УК- 4, УК-5, УК-6, УК-7, УК-8, УК-9, УК-10, ОПК-1; ОПК-2; ОПК-3; ОПК-4; ОПК-5,ОПК-6; ОПК-7,ОПК-8; ПК-1; ПК-2; ПК-3; ПК-4; ПК-5; ПК-6; ПК-7; ПК-8; ПК-9; ПК-10).

2. Фонд оценочных средств является приложением к программе государственной итоговой аттестации.

3. Разработчик: Новикова Е. Н., заведующий межинститутской базовой кафедрой

4. Проведена экспертиза ФОС.

Члены экспертной группы:

Председатель: Новикова Елена Николаевна, зав. межинститутской базовой кафедрой

Члены комиссии: Николаев Евгений Иванович, доцент департаменты цифровых и робототехнических систем и электроники.

Винокурский Дмитрий Леонидович, доцент департаменты цифровых и робототехнических систем и электроники.

(Ф.И.О., должность)

Представитель организации-работодателя Свистунов Игорь Викторович, ООО «Стилсофт»

(Ф.И.О., должность)

Экспертное заключение фонд оценочных средств позволяет определить уровень сформированности компетенций и может быть использован при итоговой аттестации студентов направления 09.03.04 «Программная инженерия», профиль «Инженерия сложных программных систем».

5. Срок действия ФОС: на срок реализации образовательной программы.

1. Описание показателей и критериев оценивания компетенций, а также шкалоценивания

Оценка «отлично» (5 баллов)

Выставляется студенту, глубоко и прочно усвоившему программный материал по дисциплинам «Базы данных», «Программная инженерия», «Конструирование программного обеспечения» и «Алгоритмы и структуры данных», исчерпывающе, последовательно, грамотно и логично его излагающему, в ответе которого тесно связывается теория с практикой разработки программного обеспечения.

При этом студент:

- не затрудняется с ответом при видоизменении задания, свободно справляется с задачами по проектированию БД (нормализация, SQL-запросы, индексы), анализу сложности алгоритмов (O-нотация), выбору архитектурных паттернов (GoF, Clean Architecture, CQRS, Saga) и настройке CI/CD;
- демонстрирует знание и понимание полного жизненного цикла разработки ПО (от анализа требований до сопровождения), владеет методологиями Agile (Scrum, Kanban) и инструментами управления проектами (Jira, Trello, Git Flow);
- находит, обрабатывает и анализирует информацию из официальной документации фреймворков (Spring, Hibernate, Django), научно-технических статей (IEEE, ACM) и профессиональных сообществ (Habr, GitHub);
- использует спецификации, стандарты, правила и рекомендации в профессиональной области: ГОСТ 34, ISO/IEC 12207, OpenAPI/Swagger, рекомендации по безопасности OWASP Top 10;
- способен готовить обзоры технической литературы и электронных образовательных ресурсов для профессиональной деятельности;
- способен общаться в устной и письменной форме на русском языке и иностранном (английском) языке, используя профессиональную терминологию (SQL, ORM, JWT, CI/CD, latency, throughput);
- анализирует рынок программно-технических средств, информационных продуктов и услуг для решения прикладных задач, обосновывает выбор стека технологий (языки, фреймворки, СУБД, облачные платформы);
- способен формировать требования к информационным системам на основе анализа предметной области, идентифицировать, классифицировать и описывать проблемы, находить методы и подходы к их решению (в том числе с использованием алгоритмов машинного обучения и анализа данных);
- способен проектировать и разрабатывать программные продукты с применением современных архитектурных стилей (Clean Architecture, микросервисы, событийно-ориентированная архитектура), паттернов проектирования GoF и принципов SOLID;
- владеет навыками тестирования (unit, интеграционное, нагрузочное), профилирования и отладки кода, настройки CI/CD пайплайнов (GitHub Actions, GitLab CI);
- демонстрирует умение работать с базами данных на уровне проектирования схем (нормализация до 3НФ), оптимизации запросов (индексы, оконные функции) и миграций (Liquibase, Flyway);
- свободно применяет алгоритмы и структуры данных для решения практических задач (сортировки, поиск, обход графов, динамическое программирование), анализирует временную и ёмкостную сложность.

Оценка «хорошо» (4 балла)

Выставляется студенту, твёрдо знающему программный материал, грамотно и по существу излагающему его, не допускающему существенных неточностей в ответе на вопрос, правильно применяющему теоретические положения при решении практических вопросов и задач, владеющему необходимыми знаниями и приёмами их выполнения.

При этом студент:

- демонстрирует хорошие знания учебной литературы и нормативных актов, обладает навыками анализа источников, знает основные проблемы дисциплин, умеет устанавливать основные причинно-следственные связи;

- знает и понимает предметную область и профессию, но испытывает затруднения в некоторых трудовых функциях (например, в выборе оптимального архитектурного стиля или в настройке сложных CI/CD пайплайнов);
- способен находить, обрабатывать и анализировать информацию из ограниченного круга источников (основная учебная литература, документация);
- способен использовать спецификации, стандарты, правила и рекомендации в профессиональной области не в полном объёме (например, знает OpenAPI, но не использует аннотации для документирования);
- способен готовить обзоры научной литературы и электронных ресурсов для профессиональной деятельности для решения узкого круга задач;
- способен общаться в устной и письменной форме на русском языке и иностранном языке, но имеет недостаточный уровень профессиональной терминологии;
- способен анализировать рынок программно-технических средств, информационных продуктов и услуг для решения прикладных задач не в полном объёме;
- формирует требования к информационным системам на основе анализа предметной области, идентифицирует, классифицирует и описывает проблемы, находит методы и подходы к решению узкого круга задач;
- способен проектировать и разрабатывать программные продукты, но испытывает проблемы с проектной документацией (оформление ТЗ, архитектурных ADR, API-спецификаций);
- владеет навыками тестирования и работы с БД, но допускает незначительные ошибки при оптимизации запросов или при настройке нагрузочного тестирования;
- применяет алгоритмы и структуры данных, но может испытывать затруднения при анализе сложности нестандартных алгоритмов.

Оценка «удовлетворительно» (3 балла)

Выставляется студенту, который имеет знания только основного материала, но не усвоил его деталей, допускает неточности, недостаточно правильные формулировки, испытывает затруднения в применении нормативных актов и стандартов.

При этом студент:

- знает и понимает предметную область, но не в полном объёме профессию (путает этапы жизненного цикла ПО, методологии Agile и Waterfall);
- способен находить, обрабатывать информацию из разных источников, но навыки анализа не развиты (не может выделить ключевые требования из технического задания);
- способен использовать некоторые спецификации, стандарты, правила и рекомендации в профессиональной области (например, знает о существовании ГОСТ, но не применяет их на практике);
- способен готовить обзоры научной литературы и электронных ресурсов, но испытывает затруднения в её использовании для профессиональной деятельности;
- способен общаться в устной и письменной форме на русском языке, но испытывает затруднения при общении на иностранном языке;
- способен анализировать рынок программно-технических средств, информационных продуктов и услуг, но испытывает затруднения при использовании результатов анализа для решения прикладных задач;
- способен формировать требования к информационным системам на основе анализа предметной области, но испытывает проблемы с идентификацией и описанием методов и подходов к их решению;
- способен проектировать и разрабатывать программные продукты не на всех этапах жизненного цикла (например, может написать код, но не умеет его тестировать или документировать);
- имеет базовые навыки работы с БД (простые SELECT-запросы), но не владеет нормализацией и оптимизацией;
- знает основные алгоритмы и структуры данных, но не умеет оценивать их сложность и выбирать подходящие для конкретной задачи;
- имеет представление о тестировании, но не может написать unit-тесты или настроить CI/CD.

Оценка «неудовлетворительно» (2 балла)

Выставляется студенту, который демонстрирует слабое знание содержания дисциплин, плохо ориентируется в основных понятиях, знает значительную часть программного материала на очень низком уровне, допускает существенные ошибки, неуверенно, с большим затруднением показывает практические знания.

При этом студент:

- слабо владеет законодательным материалом и стандартами, при определении причинно-следственных связей испытывает серьёзные затруднения;
- слабо ориентируется в предметной области и с трудом понимает профессию (не может объяснить разницу между программированием и программной инженерией);
- испытывает серьёзные затруднения при сборе, обработке и анализе информации из разных источников, а также при использовании спецификаций, стандартов, правил и рекомендаций в профессиональной области;
- при подготовке обзоров научной литературы и электронных ресурсов для профессиональной деятельности допускает серьёзные ошибки;
- с трудом общается в устной и письменной форме на русском и иностранном языках, допускает серьёзные ошибки;
- допускает серьёзные ошибки при анализе рынка программно-технических средств, информационных продуктов и услуг, применяемых для решения прикладных задач;
- с грубыми ошибками формулирует требования к информационным системам на основе анализа предметной области, испытывает серьёзные затруднения при идентификации, классификации и описании проблемы, определении методов и подходов к её решению;
- не способен проектировать и разрабатывать программные продукты даже на базовом уровне (не может написать работающий код, спроектировать БД или выполнить тестирование);
- не знает основные алгоритмы и структуры данных, не может объяснить разницу между стеком и очередью, между DFS и BFS;
- не владеет инструментарием разработчика (Git, IDE, системы сборки).

Сводная таблица критериев оценивания

Критерий	Отлично (5)	Хорошо (4)	Удовлетворительно (3)	Неудовлетворительно (2)
Знание теоретического материала	Глубокое, прочное, системное	Твёрдое, с незначительными пробелами	Поверхностное, только основные понятия	Отсутствует или фрагментарное
Практические навыки (БД)	Проектирование схем, оптимизация запросов, миграции	Базовое проектирование, простые запросы	Только SELECT-запросы	Не может написать запрос
Практические навыки (Алгоритмы)	Анализ сложности, выбор оптимальной структуры	Базовый анализ сложности	Знает алгоритмы, но не анализирует сложность	Не знает алгоритмов
Практические	Полный ЖЦ,	Знает ЖЦ и	Знает	Не знает моделей

Критерий	Отлично (5)	Хорошо (4)	Удовлетворительно (3)	Неудовлетворительно (2)
навыки (Программная инженерия)	Agile, Git Flow, документирование	Agile, но с пробелами	отдельные модели ЖЦ	ЖЦ
Практические навыки (Конструирование ПО)	TDD, CI/CD, нагрузочное тестирование, рефакторинг	Базовое тестирование, CI/CD с пробелами	Знает о тестировании, но не применяет	Не владеет инструментами тестирования
Иностранный язык	Свободно использует профессиональную терминологию	Использует базовую терминологию	Затрудняется в профессиональной коммуникации	Не использует иностранный язык
Документация	Полный комплект по ГОСТ	Основные документы		

Критерии оценивания компетенций на государственном экзамене

Оценка «отлично» (5 баллов)

Выставляется студенту, глубоко и прочно усвоившему программный материал по дисциплинам «Базы данных», «Программная инженерия», «Конструирование программного обеспечения» и «Алгоритмы и структуры данных», исчерпывающе, последовательно, грамотно и логично его излагающему, в ответе которого тесно связывается теория с практикой разработки программного обеспечения.

При этом студент:

- не затрудняется с ответом при видоизменении задания, свободно справляется с задачами по проектированию БД (нормализация, SQL-запросы, индексы), анализу сложности алгоритмов (O-нотация), выбору архитектурных паттернов (GoF, Clean Architecture, CQRS, Saga) и настройке CI/CD;
- демонстрирует знание и понимание полного жизненного цикла разработки ПО (от анализа требований до сопровождения), владеет методологиями Agile (Scrum, Kanban) и инструментами управления проектами (Jira, Trello, Git Flow);
- находит, обрабатывает и анализирует информацию из официальной документации фреймворков (Spring, Hibernate, Django), научно-технических статей (IEEE, ACM) и профессиональных сообществ (Habr, GitHub);
- использует спецификации, стандарты, правила и рекомендации в профессиональной области: ГОСТ 34, ISO/IEC 12207, OpenAPI/Swagger, рекомендации по безопасности OWASP Top 10;
- способен готовить обзоры технической литературы и электронных образовательных ресурсов для профессиональной деятельности;
- способен общаться в устной и письменной форме на русском языке и иностранном (английском) языке, используя профессиональную терминологию (SQL, ORM, JWT, CI/CD, latency, throughput);
- анализирует рынок программно-технических средств, информационных продуктов и услуг для решения прикладных задач, обосновывает выбор стека технологий (языки, фреймворки, СУБД, облачные платформы);
- способен формировать требования к информационным системам на основе анализа предметной области, идентифицировать, классифицировать и описывать проблемы, находить методы и подходы к их решению (в том числе с использованием алгоритмов машинного обучения и анализа данных);
- способен проектировать и разрабатывать программные продукты с применением современных архитектурных стилей (Clean Architecture, микросервисы, событийно-ориентированная архитектура), паттернов проектирования GoF и принципов SOLID;
- владеет навыками тестирования (unit, интеграционное, нагрузочное), профилирования и отладки кода, настройки CI/CD пайплайнов (GitHub Actions, GitLab CI);
- демонстрирует умение работать с базами данных на уровне проектирования схем (нормализация до 3НФ), оптимизации запросов (индексы, оконные функции) и миграций (Liquibase, Flyway);
- свободно применяет алгоритмы и структуры данных для решения практических задач (сортировки, поиск, обход графов, динамическое программирование), анализирует временную и ёмкостную сложность.

Оценка «хорошо» (4 балла)

Выставляется студенту, твёрдо знающему программный материал, грамотно и по существу излагающему его, не допускающему существенных неточностей в ответе на вопрос, правильно применяющему теоретические положения при решении практических вопросов и задач, владеющему необходимыми знаниями и приёмами их выполнения.

При этом студент:

- демонстрирует хорошие знания учебной литературы и нормативных актов, обладает навыками анализа источников, знает основные проблемы дисциплин, умеет устанавливать основные причинно-следственные связи;
- знает и понимает предметную область и профессию, но испытывает затруднения в некоторых трудовых функциях (например, в выборе оптимального архитектурного стиля или в настройке

сложных CI/CD пайплайнов);
- способен находить, обрабатывать и анализировать информацию из ограниченного круга источников (основная учебная литература, документация);
- способен использовать спецификации, стандарты, правила и рекомендации в профессиональной области не в полном объёме (например, знает OpenAPI, но не использует аннотации для документирования);
- способен готовить обзоры научной литературы и электронных ресурсов для профессиональной деятельности для решения узкого круга задач;
- способен общаться в устной и письменной форме на русском языке и иностранном языке, но имеет недостаточный уровень профессиональной терминологии;
- способен анализировать рынок программно-технических средств, информационных продуктов и услуг для решения прикладных задач не в полном объёме;
- формирует требования к информационным системам на основе анализа предметной области, идентифицирует, классифицирует и описывает проблемы, находит методы и подходы к решению узкого круга задач;
- способен проектировать и разрабатывать программные продукты, но испытывает проблемы с проектной документацией (оформление ТЗ, архитектурных ADR, API-спецификаций);
- владеет навыками тестирования и работы с БД, но допускает незначительные ошибки при оптимизации запросов или при настройке нагрузочного тестирования;
- применяет алгоритмы и структуры данных, но может испытывать затруднения при анализе сложности нестандартных алгоритмов.

Оценка «удовлетворительно» (3 балла)

Выставляется студенту, который имеет знания только основного материала, но не усвоил его деталей, допускает неточности, недостаточно правильные формулировки, испытывает затруднения в применении нормативных актов и стандартов.

При этом студент:

- знает и понимает предметную область, но не в полном объёме профессию (путает этапы жизненного цикла ПО, методологии Agile и Waterfall);
- способен находить, обрабатывать информацию из разных источников, но навыки анализа не развиты (не может выделить ключевые требования из технического задания);
- способен использовать некоторые спецификации, стандарты, правила и рекомендации в профессиональной области (например, знает о существовании ГОСТ, но не применяет их на практике);
- способен готовить обзоры научной литературы и электронных ресурсов, но испытывает затруднения в её использовании для профессиональной деятельности;
- способен общаться в устной и письменной форме на русском языке, но испытывает затруднения при общении на иностранном языке;
- способен анализировать рынок программно-технических средств, информационных продуктов и услуг, но испытывает затруднения при использовании результатов анализа для решения прикладных задач;
- способен формировать требования к информационным системам на основе анализа предметной области, но испытывает проблемы с идентификацией и описанием методов и подходов к их решению;
- способен проектировать и разрабатывать программные продукты не на всех этапах жизненного цикла (например, может написать код, но не умеет его тестировать или документировать);
- имеет базовые навыки работы с БД (простые SELECT-запросы), но не владеет нормализацией и оптимизацией;
- знает основные алгоритмы и структуры данных, но не умеет оценивать их сложность и выбирать подходящие для конкретной задачи;
- имеет представление о тестировании, но не может написать unit-тесты или настроить CI/CD.

Оценка «неудовлетворительно» (2 балла)

Выставляется студенту, который демонстрирует слабое знание содержания дисциплин, плохо ориентируется в основных понятиях, знает значительную часть программного материала на очень низком уровне, допускает существенные ошибки, неуверенно, с большим затруднением показывает практические знания.

При этом студент:

- слабо владеет законодательным материалом и стандартами, при определении причинно-следственных связей испытывает серьёзные затруднения;
- слабо ориентируется в предметной области и с трудом понимает профессию (не может объяснить разницу между программированием и программной инженерией);
- испытывает серьёзные затруднения при сборе, обработке и анализе информации из разных источников, а также при использовании спецификаций, стандартов, правил и рекомендаций в профессиональной области;
- при подготовке обзоров научной литературы и электронных ресурсов для профессиональной деятельности допускает серьёзные ошибки;
- с трудом общается в устной и письменной форме на русском и иностранном языках, допускает серьёзные ошибки;
- допускает серьёзные ошибки при анализе рынка программно-технических средств, информационных продуктов и услуг, применяемых для решения прикладных задач;
- с грубыми ошибками формулирует требования к информационным системам на основе анализа предметной области, испытывает серьёзные затруднения при идентификации, классификации и описании проблемы, определении методов и подходов к её решению;
- не способен проектировать и разрабатывать программные продукты даже на базовом уровне (не может написать работающий код, спроектировать БД или выполнить тестирование);
- не знает основные алгоритмы и структуры данных, не может объяснить разницу между стеком и очередью, между DFS и BFS;
- не владеет инструментарием разработчика (Git, IDE, системы сборки).

Сводная таблица критериев оценивания

Критерий	Отлично (5)	Хорошо (4)	Удовлетворительно (3)	Неудовлетворительно (2)
Знание теоретического материала	Глубокое, прочное, системное	Твёрдое, с незначительными пробелами	Поверхностное, только основные понятия	Отсутствует или фрагментарное
Практические навыки (БД)	Проектирование схем, оптимизация запросов, миграции	Базовое проектирование, простые запросы	Только SELECT-запросы	Не может написать запрос
Практические навыки (Алгоритмы)	Анализ сложности, выбор оптимальной структуры	Базовый анализ сложности	Знает алгоритмы, но не анализирует сложность	Не знает алгоритмов
Практические	Полный ЖЦ,	Знает ЖЦ и	Знает отдельные	Не знает моделей

Критерий	Отлично (5)	Хорошо (4)	Удовлетворительно (3)	Неудовлетворительно (2)
навыки (Программная инженерия)	Agile, Git Flow, документирование	Agile, но с пробелами	модели ЖЦ	ЖЦ
Практические навыки (Конструирование ПО)	TDD, CI/CD, нагрузочное тестирование, рефакторинг	Базовое тестирование, CI/CD с пробелами	Знает о тестировании, но не применяет	Не владеет инструментами тестирования
Иностранный язык	Свободно использует профессиональную терминологию	Использует базовую терминологию	Затрудняется в профессиональной коммуникации	Не использует иностранный язык
Документация	Полный комплект по ГОСТ	Основные документы		

Критерии оценивания компетенций на защите выпускной квалификационной работы

Оценка «отлично» (5 баллов)

Выставляется обучающемуся, если он **на высоком уровне** применяет методы проектирования, разработки и сопровождения программного обеспечения, а также демонстрирует:

По разделу 1 (Анализ и требования):

- Глубокий анализ предметной области с выявлением всех значимых сущностей и связей.
- Качественное исследование рынка и аналогов (не менее 3) с обоснованием конкурентных преимуществ.
- Полную и непротиворечивую спецификацию функциональных и нефункциональных требований (FURPS+).
- Корректно построенные диаграммы вариантов использования (UML Use Case).

По разделу 2 (Архитектура):

- Обоснованный выбор архитектурного стиля (Clean Architecture / PCMEF / микросервисы) на основе сценариев качества.
- Профессионально выполненное документирование архитектуры в нотациях C4 model (уровни C1, C2, C3) и UML.

- Корректно спроектированную базу данных (ER-диаграмма, нормализация до 3НФ, выбор СУБД).
- Обоснованный выбор стека технологий (языки, фреймворки, инструменты) с учётом требований проекта.

По разделу 3 (Реализация):

- Качественно написанный программный код с соблюдением принципов SOLID, DRY, KISS.
- Применение современных паттернов проектирования GoF (не менее 3).
- Профессиональное использование Git со стратегией ветвления (Git Flow / GitHub Flow), осмысленные коммиты, документация в README.md.
- Реализацию аутентификации (JWT/OAuth2), авторизации (RBAC), защиты от OWASP Top 10, логирования.

По разделу 4 (Тестирование и качество):

- Разработанные unit-тесты с покрытием ключевых компонентов не менее 70%.
- Написанные интеграционные и E2E-тесты (при наличии UI).
- Проведённое нагрузочное тестирование (JMeter/k6) с анализом результатов (RPS, latency p95/p99).
- Настроенный CI/CD пайплайн (GitHub Actions / GitLab CI) с автоматической сборкой, тестированием и деплоем.

По разделу 5 (Технико-экономическое обоснование):

- Корректный расчёт трудоёмкости (PERT / COCOMO II / функциональные точки).
- Полную себестоимость разработки с обоснованием всех статей затрат.
- Расчёт показателей экономической эффективности (NPV, PI, IRR, срок окупаемости) с интерпретацией результатов.
- Обоснованный вывод о целесообразности разработки (make-or-buy анализ).

По разделу 6 (Безопасность и охрана труда):

- Полный анализ опасных и вредных факторов на рабочем месте оператора ПК.
- Качественно выполненный расчёт одного из параметров (освещение / вентиляция / шум) в соответствии с нормами СанПиН.
- Аргументированные мероприятия по обеспечению электробезопасности и пожарной безопасности.
- **По защите ВКР:**
- Чёткая и логичная презентация (не менее 5 слайдов), отражающая все ключевые этапы работы.
- Уверенная демонстрация работающего программного продукта.
- Аргументированные и полные ответы на вопросы членов ГЭК на русском и (при необходимости) иностранном языках.

Оценка «хорошо» (4 балла)

Выставляется обучающемуся, если пояснительная записка к ВКР содержит **незначительные ошибки, не влияющие на общий результат**, а

обучающийся **на хорошем уровне** применяет методы проектирования, разработки и сопровождения ПО, демонстрируя знание и умение по всем перечисленным выше пунктам, но с **несущественными неточностями**:

- Анализ предметной области выполнен, но не полностью выявлены все взаимосвязи.
- Исследование аналогов проведено, но сравнение выполнено не по всем критериям.
- Архитектура спроектирована в целом верно, но нотации C4/UML оформлены с незначительными ошибками.
- Код написан с соблюдением SOLID, но есть отдельные нарушения принципов (например, не полностью реализован Dependency Inversion).
- Unit-тесты написаны, но покрытие кода составляет 50–70%.
- Нагрузочное тестирование проведено, но анализ результатов неполный.
- CI/CD пайплайн настроен, но автоматизирован не полностью.
- Экономический расчёт содержит незначительные арифметические ошибки, не влияющие на итоговые выводы.
- Расчёт по безопасности выполнен, но с небольшими отклонениями от нормативов.
- Защита: презентация оформлена хорошо, демонстрация ПО есть, но ответы на отдельные вопросы неполные или требующие уточнений.

Оценка «удовлетворительно» (3 балла)

Выставляется обучающемуся, если пояснительная записка к ВКР содержит **ошибки, но в целом работа признаётся выполненной**, а обучающийся **на недостаточно хорошем уровне** применяет методы проектирования, разработки и сопровождения ПО, демонстрируя:

- Поверхностный анализ предметной области (не выявлены ключевые сущности и связи).
- Исследование аналогов неполное (менее 3 аналогов) или без сравнительного анализа.
- Архитектура спроектирована с ошибками, нотации C4/UML выполнены частично, ADR отсутствуют.
- Код написан, но принципы SOLID соблюдены не полностью, есть «запахи кода».
- Git используется только на уровне commit/push, стратегия ветвления отсутствует.
- Unit-тесты написаны, но покрытие кода менее 50%, интеграционные тесты отсутствуют.
- Нагрузочное тестирование не проводилось или проведено неверно.
- CI/CD пайплайн не настроен.
- Экономический расчёт содержит существенные ошибки, но основные показатели рассчитаны.
- Расчёт по безопасности выполнен с ошибками, не соответствует нормативам.
- Документация оформлена с нарушениями структуры и требований ГОСТ.
- Защита: презентация оформлена небрежно, демонстрация ПО нестабильна, ответы на вопросы неуверенные.

Оценка «неудовлетворительно» (2 балла)

Выставляется обучающемуся, если пояснительная записка к ВКР содержит **грубые ошибки**, а обучающийся **на очень низком уровне** (или не применяет вовсе) методы проектирования, разработки и сопровождения ПО, а также демонстрирует:

- Отсутствие анализа предметной области или выполнение его формально (без выявления сущностей и связей).
- Исследование аналогов не проведено.
- Архитектура не спроектирована или не соответствует предъявленным требованиям.
- Программный код не работает или не соответствует техническому заданию.
- Git не использовался или репозиторий пуст.
- Тестирование не проводилось или тесты отсутствуют.
- Экономический расчёт отсутствует или содержит грубые ошибки, делающие выводы неверными.
- Раздел по безопасности отсутствует или выполнен формально (без расчётов и ссылок на нормативы).
- Документация отсутствует или не соответствует требованиям.
- Защита: презентация отсутствует, демонстрация ПО не произведена, ответы на вопросы не даны или неверны.

Сводная таблица критериев оценивания

Критерий	**Отлично (5)**	**Хорошо (4)**	**Удовлетворительно (3)**	**Неудовлетворительно (2)**
Анализ предметной области	Полный, глубокий	В целом полный, с незначительными недочётами	Поверхностный, не выявлены ключевые связи	Отсутствует или формальный
Исследование аналогов	≥3 аналогов, полное сравнение	≥3 аналогов, сравнение не по всем критериям	Менее 3 аналогов, сравнение неполное	Не проведено
Архитектура (C4/UML)	Полная, профессиональная	В целом верная, с незначите	С ошибками, нотации	Не спроектирована

Критерий	Отлично (5)	Хорошо (4)	Удовлетворительно (3)	Неудовлетворительно (2)
		льными ошибками	выполнены частично	
Качество кода (SOLID, паттерны)	Высокое, SOLID соблюдён, ≥3 паттернов	Хорошее, SOLID в основном соблюдён	Удовлетворительное, SOLID соблюдён не полностью	Низкое, код не работает
Git (ветвление, коммиты)	Профессиональный Git Flow	GitHub Flow с осмысленными коммитами	Только commit/push, без стратегии	Не использовался
Unit-тесты (покрытие)	≥70%	50–70%	<50%	Отсутствуют
Нагрузочное тестирование	Проведено, анализ полный	Проведено, анализ неполный	Не проведено или неверно	Не проведено
CI/CD	Полностью настроен	Настроен частично	Не настроен	Не настроен
Экономический расчёт	Корректный, с интерпретацией	С незначительными ошибками	С существенными ошибками	Отсутствует или грубо ошибочный
Безопасность и охрана труда	Полный анализ + расчёт	Анализ + расчёт с неточност	Анализ без расчёта	Отсутствует

Критерий	Отлично (5)	Хорошо (4)	Удовлетворительно (3)	Неудовлетворительно (2)
		ями	или расчёт с ошибками	
Документация (ПЗ)	Полная, по ГОСТ	В целом по ГОСТ, с неточностями	С нарушениями структуры и ГОСТ	Отсутствует или не соответствует
Презентация и защита	5+ слайдов, уверенно, демонстрация	Хорошо, но отдельные ответы неполные	Небрежно, демонстрация нестабильна	Отсутствует

2. Типовые контрольные задания или иные материалы, необходимые для оценки результатов освоения образовательной программы

2.1. Вопросы к экзамену

Дисциплина «Базы данных»

1. Основные понятия БД: база данных, ИС, вычислительная система, банк данных, СУБД, словарь данных, администратор БД.
2. Перечислите и охарактеризуйте функции СУБД.
3. Перечислите и охарактеризуйте классификации СУБД.
4. Назовите и охарактеризуйте уровни архитектуры СУБД.
5. Дайте определения понятий: клиент, сервер, архитектура «файл- сервер», архитектура «клиент-сервер».
6. Опишите процесс функционирования информационной системы с файл-сервером.
7. Опишите процесс функционирования информационной системы с сервером баз данных.
8. Дайте определение понятия «транзакция». Приведите пример транзакции. Перечислите свойства транзакции и опишите процессы журнализации и отката транзакций.
9. Опишите реляционную модель данных.

10. Опишите понятия инкапсуляция, наследование и полиморфизм с точки зрения теории БД.
11. Опишите элементы реляционной модели БД: отношение, кортеж, атрибут, домен, значение атрибута, схема отношения, первичный ключ. Перечислите свойства отношений.
12. Перечислите и охарактеризуйте виды связей между отношениями. Приведите примеры.
13. Сравните понятия потенциальный, первичный и внешний ключ.
14. Опишите процессы ограничения и каскадирования операции.
15. Опишите операции реляционной алгебры: объединение, пересечение, разность и декартово произведение отношений. Приведите примеры.
16. Опишите операции реляционной алгебры: выборка, проекция, соединение и деление отношений. Приведите примеры.
17. Опишите понятие функциональной зависимости и процесс выделения первичного ключа из потенциального ключа.
18. Перечислите характеристики «эффективной» БД. Опишите процесс приведения БД к 1НФ.
19. Опишите процесс приведения БД к 2НФ.
20. Опишите процесс приведения БД к 3НФ.
21. Опишите понятия: сущность, атрибут, связь. Охарактеризуйте процесс преобразования ER-модели в реляционную БД.
22. Опишите процесс восстановления целостности БД.
23. Перечислите проблемы, возникающие в результате параллелизма транзакций, и назовите методы их разрешения.
24. Охарактеризуйте подходы к обеспечению безопасности БД и методы управления доступом к БД.
25. Дайте определение понятия целостности БД и перечислите существующие уровни изолированности транзакций.
26. Охарактеризуйте свойства полей таблицы: значение по умолчанию, условие на значение, маска ввода, формат полей. Приведите примеры использования каждого из данных свойств.
27. Опишите возможности использования построителя выражений при создании различных объектов БД.
28. Язык SQL. Инструкции SQL
29. Язык SQL. Типы данных в SQL
30. Язык SQL. Функции языка SQL
31. Язык SQL. Создание баз данных. Язык DDL
32. Язык SQL. Манипуляции данными. Язык DML
33. Язык SQL. Запросы на выборку данных.
34. Язык SQL. Объединения в многотабличных запросах на выборку в SQL
35. Язык SQL. Правила выполнения запросов в SQL
36. Язык SQL. Подчиненный запрос в SQL. Примеры
37. Язык SQL. Восстановление базы данных. Транзакции
38. Язык SQL. Конструкция SELECT
39. Язык SQL. Операторы манипулирования данными (DML).
40. Язык SQL. Операторы определения объектов БД (DDL).
41. Язык SQL. Оператор SELECT.
42. Язык SQL. Создание таблиц с уникальными и внешними ключами.

43. Язык SQL. Модификация таблиц.
44. Язык SQL. Задание условий отбора в предложении WHERE.
45. Язык SQL. Предложения WHERE и HAVING.
46. Язык SQL. Триггеры
47. Язык SQL. Работа с строковыми типами данных.
48. Язык SQL. Работа с числовыми типами данных.
49. Язык SQL. Вложенные запросы
50. Язык SQL. Операция объединения.
51. Язык SQL. Операция пересечения.
52. Язык SQL. Операция выборки
53. Язык SQL. Представления
54. Язык SQL. Функции работы с датами
55. Язык SQL. Индексы. Назначение. Применение
56. Язык SQL. Создание уникальных индексов
57. Язык SQL. Правила выполнения запросов в SQL
58. Язык SQL. Подчиненный запрос в SQL. Примеры
59. Язык SQL. Восстановление базы данных. Транзакции
60. Язык SQL. Конструкция SELECT

Дисциплина «Программная инженерия»

1. Понятие программной инженерии. Отличие программной инженерии от программирования. Шесть тезисов программной инженерии. Стандарты ISO/IEC 12207 и SWEBOK.
2. Модели жизненного цикла ПО. Каскадная (Waterfall), V-модель, итеративные, спиральная модель. Достоинства, недостатки, области применения.
3. Гибкие методологии разработки (Agile). Манифест и принципы Agile. Scrum: роли, артефакты, события. Kanban: доска, ограничение WIP.
4. Инструментарий программной инженерии. Классификация инструментов: CASE-средства, IDE, VCS, системы управления проектами. MS Project, Git, PlantUML, Enterprise Architect.
5. Управление требованиями к ПО. Классификация требований (функциональные, нефункциональные, ограничения). Процесс управления требованиями. Трассировка требований (матрица трассировки).
6. Планирование проекта и оценка трудоёмкости. WBS (иерархическая структура работ). Диаграмма Ганта. Методы оценки трудоёмкости: экспертные оценки (Planning Poker), COCOMO I и COCOMO II.
7. Отслеживание выполнения и управление рисками. EVM (освоенный объём): PV, EV, AC, SPI, CPI. Идентификация рисков. Матрица вероятности и влияния. Планирование реагирования на риски.
8. Язык UML. Диаграммы вариантов использования и классов. Use Case Diagram: акторы, прецеденты, отношения include/extend. Class Diagram: классы, атрибуты, методы, отношения (ассоциация, агрегация, композиция, наследование).
9. Диаграммы последовательности, деятельности и состояний UML. Sequence Diagram: объекты, линии жизни, сообщения. Activity Diagram: действия, ветвления, параллельные потоки. State Machine Diagram: состояния, переходы, события.
10. Диаграммы компонентов и развёртывания UML. Component Diagram: компоненты, интерфейсы, зависимости. Deployment Diagram: узлы, артефакты, соединения.
11. Введение в архитектуру ПО. Паттерн PCMEF. Принципы архитектуры. Слои PCMEF: Presentation, Control, Mediator, Entity, Foundation. Правило зависимостей.
12. Адаптация архитектурного паттерна PCMEF для различных типов приложений. Desktop, Web, Mobile, Enterprise. Сравнение подходов.
13. Паттерны проектирования GoF. Классификация. Порождающие паттерны: Singleton, Factory Method, Abstract Factory. Примеры применения.
14. Структурные паттерны GoF. Adapter, Facade, Decorator, Proxy. Назначение, примеры использования, сравнение.
15. Поведенческие паттерны GoF. Observer, Strategy, Command. Назначение, примеры использования.

16. Продвинутые паттерны GoF. Composite, Chain of Responsibility, Visitor. Назначение, примеры использования.
17. Проектирование баз данных. Реляционная модель. Первичные и внешние ключи. Нормализация: 1НФ, 2НФ, 3НФ. Денормализация: цели и риски.
18. ORM (Object-Relational Mapping). JPA и Hibernate. Преимущества и недостатки ORM по сравнению с прямым SQL. Проблема N+1 запроса.
19. Детальное проектирование и спецификация. Диаграммы последовательности. JavaDoc. OpenAPI / Swagger для спецификации REST API.
20. Реализация ядра системы в архитектуре PCMEF. Слои Entity, Foundation, Mediator, Control. Внедрение зависимостей (DI) через конструктор.
21. Качество кода. Запахи кода (Code Smells). Примеры: длинный метод, большой класс, дублирование кода. Статический анализ (SonarQube, Checkstyle).
22. Рефакторинг. Техники рефакторинга. Каталог рефакторингов Фаулера (извлечение метода, замена условного оператора полиморфизмом). Технический долг: стратегии управления.
23. Модульное тестирование и TDD. JUnit. Mockito для создания заглушек. Покрытие кода (Code Coverage). Цикл TDD: Red → Green → Refactor.
24. Тест-дизайн. Классы эквивалентности. Граничные значения. Попарное тестирование (pairwise). Примеры применения.
25. Интеграционное и системное тестирование. Стратегии интеграции. TestContainers. Selenium для тестирования веб-интерфейсов. Нагрузочное тестирование (JMeter, k6): метрики RPS, latency p95/p99.
26. Микросервисная архитектура. Принципы, преимущества, недостатки. API Gateway, Service Discovery. Сравнение с монолитной архитектурой.
27. Событийно-ориентированная архитектура (EDA). События, обработчики, брокеры сообщений. Когда применять EDA.
28. CQRS (Command Query Responsibility Segregation). Разделение команд и запросов. Преимущества и недостатки. Event Sourcing: хранение событий вместо текущего состояния.
29. Serverless архитектура. Функции как сервис (FaaS). Примеры: AWS Lambda, Yandex Cloud Functions. Преимущества, ограничения, сценарии применения.
30. Чистая архитектура (Clean Architecture). Слои Clean Architecture (Entities, Use Cases, Interface Adapters, Frameworks). Сравнение с PCMEF. Независимость от фреймворков через Dependency Inversion.
31. Domain-Driven Design (DDD). Стратегическое проектирование: ограниченные контексты, контекстная карта. Тактические паттерны: Entity, Value Object, Aggregate, Repository.
32. Безопасность на уровне базы данных и транзакций. ACID. Уровни изоляции транзакций (Read Uncommitted, Read Committed, Repeatable Read, Serializable). Проблемы (грязное чтение, фантомы).
33. Аутентификация и авторизация в ПО. JWT (JSON Web Token): структура, подпись, принцип работы. BCrypt для хеширования паролей.
34. Продвинутая безопасность. OAuth2 и OpenID Connect. SSO (Single Sign-On). API-шлюзы. Безопасность микросервисов (mTLS, JWT).
35. Распределённые транзакции. Двухфазная фиксация (2PC): этапы, проблемы. Паттерн Saga: хореография vs оркестрация. Компенсирующие транзакции. CAP-теорема.
36. Проектирование высоконагруженных систем. Горизонтальное и вертикальное масштабирование. Кэширование (Redis, Memcached, CDN). Шардирование. Репликация (master-slave, master-master).
37. Message Brokers и асинхронное взаимодействие. RabbitMQ: очереди, обменники. Apache Kafka: топики, партиции, consumer groups. Гарантии доставки (at-most-once, at-least-once, exactly-once).
38. Контейнеризация с Docker. Dockerfile: основные инструкции (FROM, RUN, COPY, CMD). Слои. Docker Compose: многоконтейнерные приложения.
39. Оркестрация контейнеров с Kubernetes. Pods, Deployments, Services, ConfigMaps, Secrets, Ingress. Сравнение с Docker Compose.
40. CI/CD и DevOps практики. GitHub Actions / GitLab CI / Jenkins. Infrastructure as Code (Terraform, Ansible). Пайплайны: сборка, тестирование, деплой.

Алгоритмы и структуры данных

1. Размещение массивов в оперативной памяти
2. АВЛ-деревья, организация и балансировка

3. Статические и динамические массивы. Операции удаления и вставки
4. Красно-чёрные деревья, организация и балансировка
5. Поиск в упорядоченном и неупорядоченном массиве. Оценка сложности
6. Представление графов матрицей смежности и матрицей инцидентности
7. Способы выборки K наибольших/наименьших элементов массива
8. Представление графов коллекцией списков смежностей
9. Блочное умножение матриц
10. Представление графов списком рёбер
11. Алгоритм Штрассена умножения матриц
12. Представление графов с подвижными вершинами на координатной сетке
13. Умножение матриц по Винограду
14. Методы обхода графов
15. Мозаичный способ умножения матриц
16. Алгоритм поиска компонент связности графов
17. Алгоритм быстрой сортировки
18. Алгоритм (Прима) Ярника поиска минимальных остовных деревьев
19. Алгоритм пирамидальной сортировки
20. Алгоритм Краскала поиска минимальных остовных деревьев
21. Алгоритм сортировки слиянием
22. Алгоритм Дейкстры поиска минимальных путей на графе
23. Алгоритм сортировки подсчётом
24. Алгоритм Беллмана-Форда поиска минимальных путей на графе
25. Алгоритм поразрядной сортировки
26. Алгоритм блочной сортировки
27. Жадный алгоритм поиска минимального пути на графе
28. Принцип организации односвязных списков. Операции добавления, вставки и удаления элементов
29. Алгоритм Ахо-Корасик поиска подстрок в строке
30. Бинарная куча. Способ реализации. Операции
31. Очередь с приоритетом. Способ реализации. Операции
32. Словари и отображения. Операции над словарями
33. Алгоритм Бойера-Мура поиска подстроки в строке. Таблицы стоп-символов и хороших суффиксов
34. Способы обхода вершин дерева
35. Бор для хранения и поиска подстрок в строке
36. Упорядоченные двоичные деревья. Поиск, добавление и удаление вершин
37. Виды деревьев. Свойства бинарных деревьев
38. Алгоритм Рабина-Карпа поиска подстроки в строке
39. Алгоритм Кнута-Морриса-Пратта поиска подстроки в строке. Бордер. Префикс функция
40. Наивный алгоритм поиска подстроки в строке. Оценка сложности
41. Принцип организации списков на динамических массивах. Представление в памяти и операции
42. Достоинства и недостатки списков на динамических массивах
43. Кольцевые хэш-функции
44. Круговые списки и барабаны
45. Назначение и принцип организации хеш-таблиц и хеш-деревьев. Способы разрешения коллизий
46. Принцип организации двусвязных списков. Операции добавления, вставки и удаления элементов
47. Хеширование, основанное на операциях сдвига и сложения по модулю 2
48. Дек на двусвязных списках. Операции над деком

49. Хеширование, основанное на умножении и делении

50. Окрестности фон Неймана и Мура, эвристические правила определения расстояния на размеченной плоской карте

51. Стек на односвязных списках. Стековые операции

52. Очередь на односвязных списках. Операции над очередью

53. Назначение хеширования, способы реализации, понятие коллизии

Конструирование программного обеспечения

1. Классы эквивалентности и граничные значения в тест-дизайне: определение и применение.
2. TDD (Test-Driven Development): цикл Red-Green-Refactor и пример разработки сервиса заказов.
3. Mockito для мокирования зависимостей: зачем нужны моки, пример мокирования внешнего API.
4. TestContainers для интеграционного тестирования: преимущества и пример с PostgreSQL.
5. Нагрузочное тестирование в JMeter: цели, метрики (RPS, latency, throughput).
6. Профилирование производительности: инструменты (JProfiler, async profiler) и поиск узких мест.
7. Запахи кода (Code Smells) по каталогу Фаулера: длинный метод, большой класс, дублирование кода.
8. Техники рефакторинга: извлечение метода, замена условного оператора полиморфизмом.
9. Технический долг (Technical Debt): виды, стратегии управления, приоритизация.
10. Проблема N+1 запроса в JPA/Hibernate: причины и решения (@EntityGraph, join fetch).
11. Уровни изоляции транзакций: Read Uncommitted, Read Committed, Repeatable Read, Serializable.
12. Миграции баз данных: Liquibase и Flyway, интеграция в CI/CD.
13. JWT (JSON Web Token): структура, access и refresh токены, реализация в Spring Security.
14. OAuth2: Authorization Code Flow, пример входа через GitHub/Google.
15. Ролевая модель доступа (RBAC): реализация ролей USER, ADMIN в Spring Security.
16. Защита от SQL-инъекций и XSS: параметризованные запросы, санитизация вывода.
17. Хеширование паролей: BCrypt, соль (salt), работающий фактор (work factor).
18. CompletableFuture в Java: создание, композиция (thenApply, thenCompose), обработка ошибок.
19. Project Reactor: Mono и Flux, основы реактивного программирования.
20. Backpressure в реактивных потоках: механизм request(n), реализация в Project Reactor.
21. Spring WebFlux: реактивный REST API, сравнение производительности с блокирующим I/O.
22. GraphQL: схема (types, queries, mutations), резолверы, преимущества перед REST.
23. DataLoader в GraphQL: решение проблемы N+1 запросов через пакетную загрузку.
24. gRPC и Protocol Buffers: бинарный протокол, типы вызовов (Unary, Server Streaming, Client Streaming, Bidirectional Streaming).
25. Сравнение gRPC и REST/JSON: производительность, latency, размер сообщений.
26. MongoDB: документно-ориентированная БД, моделирование данных, репозиторий в Spring Data.
27. Neo4j: графовая БД, моделирование графовых структур (узлы, рёбра, свойства).
28. Полиглотное хранение (Polyglot Persistence): гибридное хранение SQL + NoSQL, когда что выбирать.
29. CI/CD пайплайны: GitHub Actions, GitLab CI, Jenkins, структура stages.
30. Статический анализ кода: SonarQube, Checkstyle, метрики качества.
31. Мониторинг приложений с Prometheus и Grafana: сбор метрик (RPS, latency, ошибки).
32. Наблюдаемость (Observability): метрики, логи, трейсинг, Correlation ID.
33. Попарное тестирование (pairwise): суть метода и минимизация количества тестов.
34. Сравнение типов тестовых двойников: Stub, Mock, Fake, Spy, Dummy.
35. Chaos-тестирование и отказоустойчивость: принципы Chaos Engineering, инструменты (Chaos Mesh).
36. Автоматический рефакторинг в IDE: возможности и примеры (IntelliJ IDEA, Eclipse).
37. Стратегии загрузки в JPA: EAGER vs LAZY, LazyInitializationException, Open Session in View.
38. Circuit Breaker: состояния Closed, Open, Half-Open, реализация в Resilience4j.
39. Retry с политиками: exponential backoff, jitter, бюджетирование повторов.
40. Распределённая трассировка: Jaeger, Zipkin, связывание логов и трейсов через Correlation ID.

Задания для проверки уровня обученности Базы данных

1. Составить на основе этой модели реляционную модель данных.

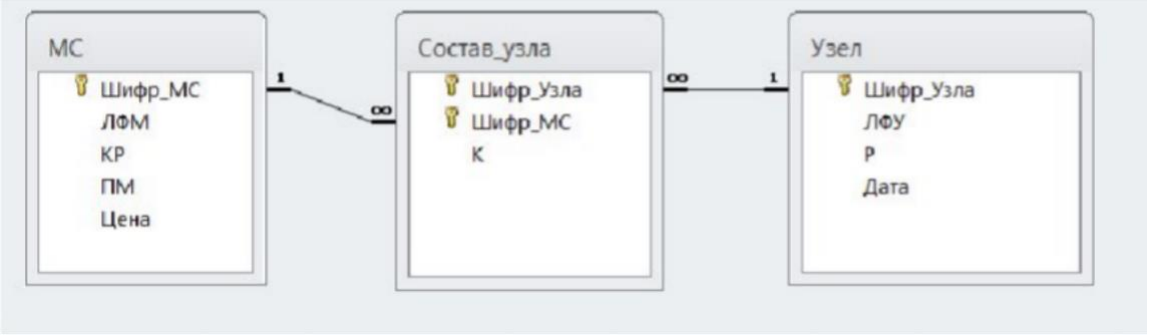

Таблица «МС»

- Шифр_МС;
- ЛФМ – логическая функция микросхемы;
- ПМ – потребляемая мощность;
- Цена – цена одной микросхемы.

Таблица «Состав_узла»

- К – количество микросхем в узле.

Таблица «Узел»

- Шифр_узла;
- ЛФУ – логическая функция узла;
- Р – разрядность;
- Дата – дата разработки узла.

Задача 2: Согласно представленной схемы данных, необходимо определить число используемых типов микросхем и оценить стоимость микросхем.

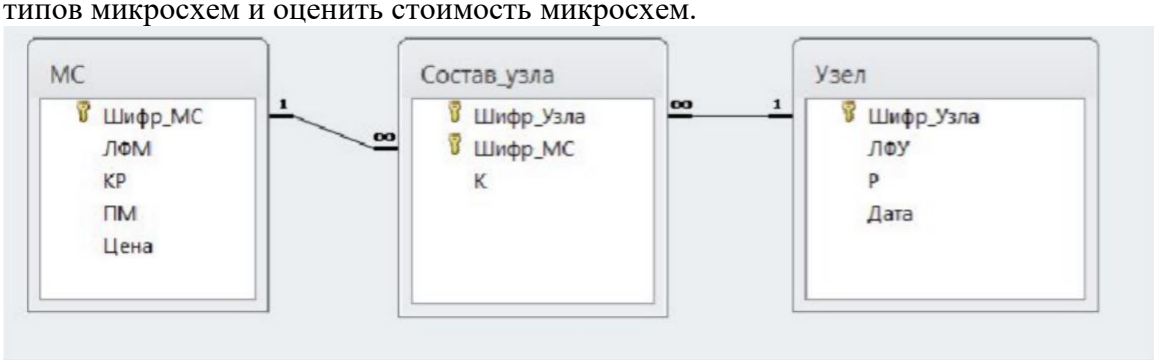

Таблица «МС»

- Шифр_МС;
- ЛФМ – логическая функция микросхемы;
- ПМ – потребляемая мощность;
- Цена – цена одной микросхемы.Таблица «Состав_узла»
- К – количество микросхем в узле.Таблица «Узел»
- Шифр_узла;
- ЛФУ – логическая функция узла;
- Р – разрядность;
- Дата – дата разработки узла.

3. Создан файл сценария с именем test2.sql и содержит следующие команды.Пояснить каждую из команд и определить результат запуска сценария.

1) SET VERIFY OFF
SELECT product_name, product_price FROM product_nWHERE
product_price >= &price;

SET VERIFY ON

2) SET VERIFY OFF
SELECT product_name, product_price FROM product_nWHERE
product_name >= ‘&name’;

SET VERIFY ON

4. Определить верный программный код сценария SQL, осуществляющий приглашение на ввод значения name и помещающий введенную строку в конструкцию WHERE. Пояснить ответ.

1) SET VERIFY
OFFSET ECHO OFF
ACCEPT name PROMPT ‘Enter name? (text)’SELECT
product_name, product_price

FROM product_n WHERE product_name = ‘&name’;SET
VERIFY ON

SET ECHO ON

2) SET VERIFY OFF
SELECT product_name, product_price FROM product_nWHERE
product_name >= ‘name’;

SET VERIFY ON 3)SET
VERIFY OFF SET ECHO OFF

SELECT product_name, product_price FROM product_nWHERE
product_name >= &name;

SET VERIFY OFFSET
ECHO OFF

5. Необходимо определить номер строки с ошибкой в следующем SQL- коде

1 ALTER TABLE hase (
2 product_name VARCHAR2(25),
3 product_price NUMBER(4,2),
4 sales NUMBER(4,2)
5);

6 INSERT INTO hase (‘ProductName 1’, 1, .08);
7 INSERT INTO hase VALUES (‘ProductName 2’, 2.5, .21);
8 INSERT INTO hase VALUES (‘ProductName 3’, 50.75, 4.19);
9 INSERT INTO hase VALUES (‘ProductName 4’, 99.99, 8.25);
10 SELECT product_name, product_price + sales hase;
11 SELECT product_name, 100 - product_price FROM
hase;12 SELECT product_name, sales / product_price from
hase;

6. Согласно представленной схемы данных, выдать список микросхем, входящих более чем в один узел и количество микросхем каждого типа, входящих более чем в одинузел.

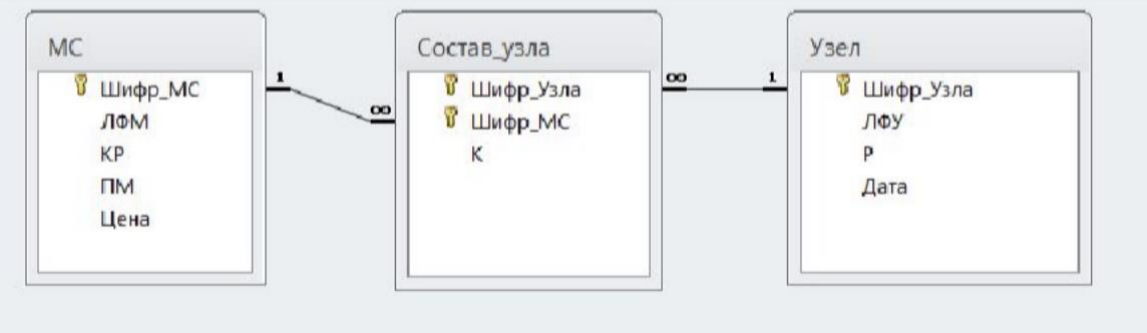

Таблица «МС»

- Шифр_МС;
- ЛФМ – логическая функция микросхемы;
- ПМ – потребляемая мощность;
- Цена – цена одной микросхемы.Таблица «Состав_узла»
- К – количество микросхем в узле.Таблица «Узел»
- Шифр_узла;
- ЛФУ – логическая функция узла;
- Р – разрядность;
- Дата – дата разработки узла.

7. Согласно представленной схемы данных, для каждого типа микросхемы выдатьколичество узлов, в которые она входит. Если микросхема не входит ни в один узел, она должна присутствовать в отчете с количеством узлов, равным нулю.

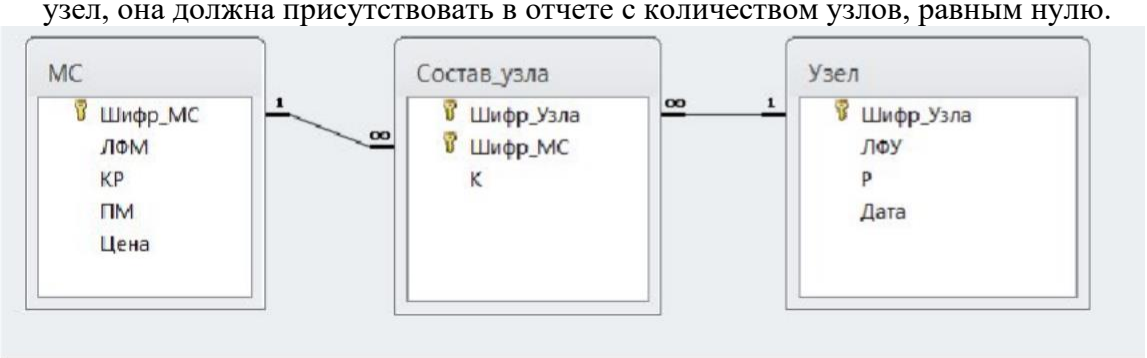

Таблица «МС»

- Шифр_МС;
- ЛФМ – логическая функция микросхемы;
- ПМ – потребляемая мощность;
- Цена – цена одной микросхемы.Таблица «Состав_узла»
- К – количество микросхем в узле.Таблица «Узел»
- Шифр_узла;
- ЛФУ – логическая функция узла;
- Р – разрядность;
- Дата – дата разработки узла.

8. Согласно представленной схемы данных, выдать мощность тех микросхем,количество которых в узлах =16. Использовать

1. Внутреннее соединение.

2. Вложенных подзапрос.
3. Коррелированный подзапрос.

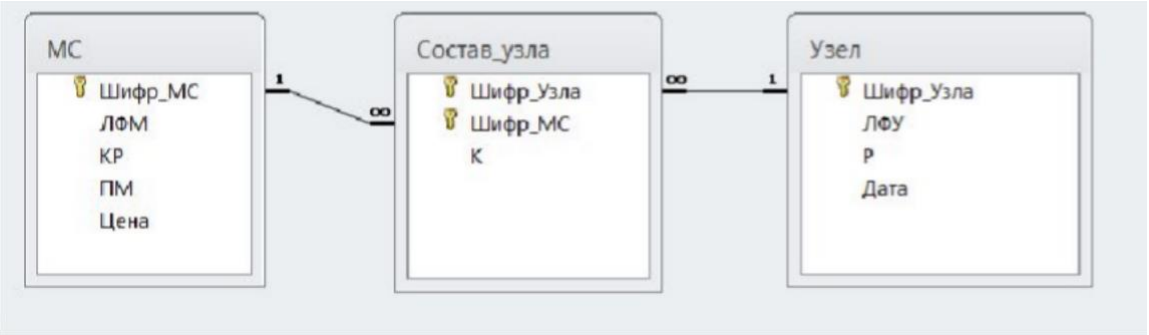

Таблица «МС»

– Шифр_МС;
– ЛФМ – логическая функция микросхемы;
– ПМ – потребляемая мощность;
– Цена – цена одной микросхемы.Таблица «Состав_узла»
– К – количество микросхем в узле.Таблица «Узел»
– Шифр_узла;
– ЛФУ – логическая функция узла;
– Р – разрядность;
– Дата – дата разработки узла.

9. Согласно представленной схемы данных, найти количество микросхем в узле снаибольшей разрядностью. Если таких узлов окажется два или больше, необходимо вывести тот узел, который был спроектирован позже других.

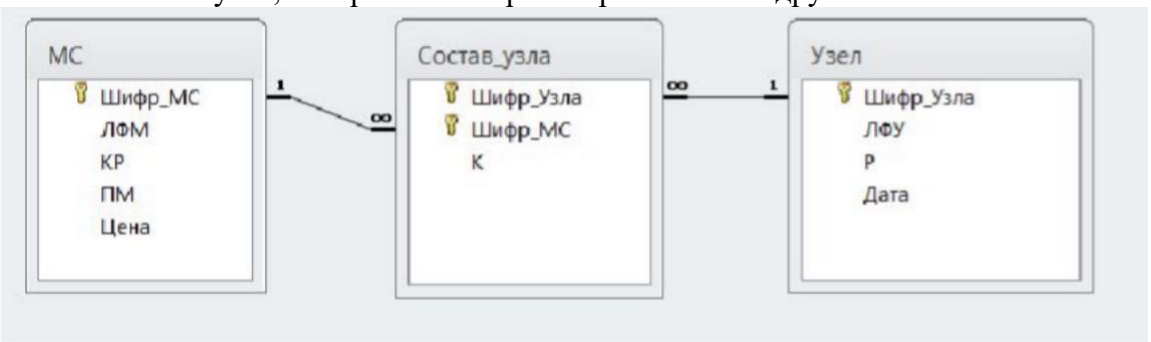

Таблица «МС»

– Шифр_МС;
– ЛФМ – логическая функция микросхемы;
– ПМ – потребляемая мощность;
– Цена – цена одной микросхемы.Таблица «Состав_узла»
– К – количество микросхем в узле.Таблица «Узел»
– Шифр_узла;
– ЛФУ – логическая функция узла;
– Р – разрядность;
– Дата – дата разработки узла.

10. Согласно представленной схемы данных, выдать ЛФМ той микросхемы, котораявстречается в узлах чаще других. Если таких МС окажется два или больше, необходимо вывести ту МС, у которой больше цена.

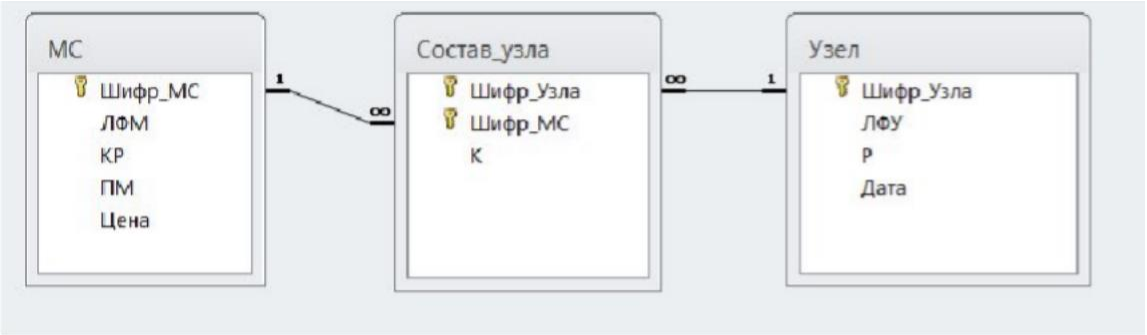

Таблица «МС»

- Шифр_МС;
- ЛФМ – логическая функция микросхемы;
- ПМ – потребляемая мощность;
- Цена – цена одной микросхемы.Таблица «Состав_узла»
- К – количество микросхем в узле.Таблица «Узел»
- Шифр_узла;
- ЛФУ – логическая функция узла;
- Р – разрядность;
- Дата – дата разработки узла.

Программная инженерия

Задание 1. Разработка WBS и диаграммы Ганта

Условие:

Вы Project Manager в IT-компании. Вам необходимо спланировать разработку веб-приложения «Система онлайн-бронирования переговорных комнат» (функции: просмотр свободных комнат, бронирование на выбранное время, отмена бронирования, уведомления на email).

Требуется:

Построить иерархическую структуру работ (WBS) не менее чем из 15 задач (минимум 3 уровня).
Определить последовательность и зависимости задач.
Построить диаграмму Ганта (руками на листе или в любом инструменте — в ответе описать словами или приложить схему).
Выделить критический путь и указать его длительность в днях.

Ожидаемый результат:

WBS в виде списка или диаграммы, диаграмма Ганта с указанным критическим путём.

Задание 2. Оценка трудоёмкости методом PERT

Условие:

Для задачи «Разработка модуля аутентификации (JWT + Refresh token)» получены три оценки от экспертов:
оптимистическая (O) = 5 дней
наиболее вероятная (M) = 8 дней
пессимистическая (P) = 14 дней

Требуется:

Рассчитать ожидаемую трудоёмкость (E) по формуле PERT.
Рассчитать стандартное отклонение (σ).
Определить диапазон, в котором трудоёмкость будет находиться с вероятностью 95% (доверительный интервал).

Формулы:
E = (O + 4M + P) / 6
σ = (P — O) / 6
95% доверительный интервал: E ± 2σ
Ожидаемый результат:
Численные значения E, σ, и интервал (например, «8,5 ± 1,5 дня»).

Задание 3. Оценка архитектуры методом ATAM (ролевая игра)

Условие:

Вам представлена архитектура микросервисной системы интернет-магазина (API Gateway, сервис заказов, сервис оплаты, сервис доставки, очередь сообщений Kafka). Заказчик требует высокой доступности (99,9%) и обработки не менее 1000 заказов в секунду.

Требуется (в роли архитектора):

Сформулировать 3 сценария атрибутов качества (например, «при увеличении нагрузки…»).
Для каждого сценария указать возможные архитектурные риски.
Предложить способы снижения этих рисков (например, резервирование, кэширование, rate limiting).

Ожидаемый результат:

Три сценария с анализом рисков и мерами по их снижению.

Задание 4. Проектирование архитектуры в нотации C4 (уровень Container)

Условие:

Необходимо спроектировать архитектуру веб-приложения «Система управления задачами (Task Manager)» с функциями:
аутентификация пользователей (JWT);
создание/редактирование/удаление задач;
комментарии к задачам;
уведомления по email.

Требуется:

Нарисовать диаграмму контейнеров (уровень C2) в нотации C4.
Указать на диаграмме: тип каждого контейнера (веб-приложение, БД, кэш, очередь, email-сервис), протоколы взаимодействия (REST/HTTP, JDBC, SMTP).

Ожидаемый результат:

Схема (описанная словами или нарисованная в тексте), включающая не менее 4 контейнеров и связи между ними.

Задание 5. Применение паттерна Saga (хореография) для распределённой транзакции

Условие:

В микросервисной системе «Заказ → Оплата → Доставка» необходимо реализовать распределённую транзакцию. Если оплата не прошла, заказ должен быть отменён. При отказе оплаты нужно отправить компенсирующее событие.

Требуется:

Описать события, которые публикует каждый сервис.
Описать подписчиков (consumer'ов) на эти события.
Объяснить, как происходит откат (компенсация) при сбое.
Указать, какие гарантии доставки сообщений нужны (at-least-once, exactly-once) и почему.

Ожидаемый результат:

Схема событий и подписчиков (в виде списка или диаграммы), пояснение механизма компенсации.

Задание 6. Проектирование диаграммы классов и последовательности

Условие:

Разрабатывается класс «Корзина покупок» (ShoppingCart) для интернет-магазина. Класс должен позволять:
добавлять товар (addItem);
удалять товар (removeItem);
изменять количество товара (updateQuantity);
рассчитывать общую стоимость (calculateTotal).

Требуется:

Нарисовать диаграмму классов UML (классы: ShoppingCart, CartItem, Product).

Нарисовать диаграмму последовательности (sequence diagram) для сценария «Пользователь добавляет товар в корзину».

Ожидаемый результат:

Диаграмма классов (3 класса с атрибутами и методами) и диаграмма последовательности (не менее 4 объектов, сообщения между ними).

Задание 7. Разработка стратегии миграции с монолита на микросервисы

Условие:

У вас есть монолитное приложение интернет-магазина (модули: каталог, корзина, заказы, оплата, доставка, пользователи). Необходимо постепенно мигрировать на микросервисную архитектуру без остановки работы системы.

Требуется:

Предложить архитектурную стратегию (назвать паттерн, например Strangler Fig).
Описать пошаговый план миграции (минимум 3 шага).
Указать, как обеспечить роутинг трафика между старым монолитом и новым микросервисом на каждом шаге.

Ожидаемый результат:

Название стратегии, пошаговый план (3–5 шагов), описание роутинга.

Задание 8. Оценка и снижение технического долга

Условие:

В коде проекта обнаружены следующие проблемы:
Метод processOrder() длиной 350 строк.
Класс UserService содержит 15 методов, которые не связаны друг с другом (нарушение SRP).
Дублирование кода: три одинаковых валидатора email в разных модулях.
Отсутствуют unit-тесты для класса PaymentProcessor.
Прямые вызовы SQL-запросов в коде контроллера (нарушение слоёной архитектуры).
Требуется:
Оценить стоимость исправления и стоимость владения (в абстрактных единицах) для каждой проблемы (по шкале «низкая/средняя/высокая»).
Приоритизировать задачи по снижению техдолга (сначала самые важные).
Для первой по приоритету проблемы предложить план рефакторинга (конкретные действия).

Ожидаемый результат:

Таблица приоритетов (проблема → приоритет → план исправления).

Задание 9. Настройка CI/CD пайплайна (описание)

Условие:

Для проекта на Java (Spring Boot) с GitLab необходимо настроить CI/CD пайплайн, который будет:
запускать модульные тесты при каждом push;
при успешных тестах собирать Docker-образ;
деплоить на staging-окружение.

Требуется:

Написать (в виде псевдокода или описать структуру) файл .gitlab-ci.yml для этого пайплайна.
Указать stages, jobs, артефакты (если нужны).
Объяснить, как настроить условный деплой только для ветки main.
Ожидаемый результат:
Описание файла CI/CD (3–5 jobs) с пояснениями.

Задание 10. Рефакторинг: замена условного оператора полиморфизмом

Условие:

Дан «плохой» код:

```java
public double calculateDiscount(String customerType, double amount) {
    if (customerType.equals("REGULAR")) {
        return amount * 0.05;
    } else if (customerType.equals("GOLD")) {
```

```
        return amount * 0.10;
    } else if (customerType.equals("PLATINUM")) {
        return amount * 0.15;
    } else {
        return 0;
    }
}
```

Требуется:
Применить рефакторинг «Replace Conditional with Polymorphism».
Спроектировать иерархию классов (например, абстрактный класс CustomerDiscount, наследники RegularDiscount, GoldDiscount, PlatinumDiscount).
Написать Java-код для реализованного решения (фрагмент, демонстрирующий полиморфное поведение).
Ожидаемый результат:
Код иерархии классов и фрагмент, использующий полиморфизм вместо if-else.

Алгоритмы и структуры данных

Задание 1. Анализ сложности алгоритма

Условие:
Дан фрагмент кода на псевдокоде:
text

```
function process(arr):
    n = len(arr)
    for i = 0 to n-1:
        for j = 0 to i-1:
            if arr[j] > arr[j+1]:
                swap(arr[j], arr[j+1])
        print(arr[i])
    for i = 0 to n-1:
        arr[i] = arr[i] * 2
    return arr
```

Требуется:
Определить асимптотическую сложность алгоритма (O-нотация) и обосновать ответ.
Указать, какие из вложенных циклов являются независимыми, а какие — зависимыми.
Оценить количество выполняемых операций в лучшем и худшем случае.
Ожидаемый результат:
Формула сложности, пояснение, оценка best/worst case.

Задание 2. Реализация быстрой сортировки (QuickSort) и анализ производительности

Условие:
Дан массив целых чисел: [5, 2, 9, 1, 5, 6, 3, 8, 7, 4].
Требуется:
Выполнить один шаг быстрой сортировки (выбрать опорный элемент, разделить массив на две части, вернуть индексы границ).
Написать (на псевдокоде или на любом языке программирования) рекурсивную функцию быстрой сортировки.
Объяснить, как выбор опорного элемента влияет на сложность в худшем случае.
Предложить способ выбора опорного элемента, гарантирующий сложность O(n log n) на любых входных данных.
Ожидаемый результат:
Шаг разделения, код функции (псевдокод), пояснение про медиану из трёх.

Задание 3. Реализация алгоритма Дейкстры для поиска кратчайшего пути

Условие:
Задан граф в виде списка смежности:
text

A: (B, 4), (C, 2)
B: (C, 1), (D, 5)
C: (D, 8), (E, 10)
D: (E, 2), (F, 6)
E: (F, 3)
F: (пусто)
(Число — вес ребра.)
Требуется:
Нарисовать граф по описанию.
Найти кратчайшие расстояния от вершины A до всех остальных вершин, выполнив алгоритм Дейкстры вручную (по шагам).
Указать, почему алгоритм Дейкстры не работает с отрицательными весами, и назвать алгоритм, который решает эту проблему.
Ожидаемый результат:
Рисунок графа, таблица расстояний после каждого шага, итоговые расстояния.

Задание 4. Обход графа: поиск в глубину (DFS) и поиск в ширину (BFS)
Условие:
Задан ориентированный граф:
text
1 → 2, 4
2 → 3, 5
3 → 1
4 → 5
5 → (пусто)
Требуется:
Выполнить обход графа в глубину (DFS), начиная с вершины 1. Записать порядок посещения вершин (используя стек или рекурсию). Указать, какие вершины были посещены повторно (если есть цикл).
Выполнить обход графа в ширину (BFS), начиная с вершины 1. Записать порядок посещения вершин (используя очередь).
Объяснить, как можно обнаружить цикл в ориентированном графе с помощью DFS.
Ожидаемый результат:
Два порядка обхода (DFS и BFS), пояснение про три цвета (белый/серый/чёрный) для обнаружения цикла.

Задание 5. Реализация хеш-таблицы с разрешением коллизий методом цепочек
Условие:
Размер хеш-таблицы = 7. Хеш-**функция: h(key) = key % 7.**
Последовательно вставляются ключи: [15, 22, 8, 29, 36, 1, 43].
Требуется:
Выполнить вставку всех ключей в хеш-таблицу методом цепочек (каждый элемент — список). Показать конечное состояние таблицы.
Определить коэффициент заполнения (load factor) после вставки всех элементов.
Предложить другую хеш-функцию, которая уменьшит количество коллизий для данного набора ключей.
Оценить среднее время поиска элемента в лучшем и худшем случае для полученной таблицы.
Ожидаемый результат:
Таблица (7 ячеек) с цепочками, load factor, пример другой хеш-функции, оценка сложности.

Задание 6. Реализация бинарного дерева поиска (BST): вставка и удаление
Условие:
Исходное пустое бинарное дерево поиска.
Требуется:
Последовательно вставить ключи: [50, 30, 70, 20, 40, 60, 80, 25, 35, 55]. Показать дерево после каждой вставки (схематично).
Удалить ключ 50 (корень). Показать два возможных способа замены (самый правый в левом поддереве или самый левый в правом поддереве). Привести итоговое дерево после удаления.
Объяснить, какой обход дерева даст отсортированный порядок ключей.
Ожидаемый результат:

Схемы дерева после вставок, два варианта после удаления, название обхода (in-order).

Задание 7. Алгоритм сжатия: кодирование длин серий (RLE)
Условие:
Дана строка: "AAABBBCCCCDDDEEEEEEFFFFFFFFF".
Требуется:
Выполнить сжатие строки методом RLE (Run-Length Encoding) — представить её в виде «символ + количество повторений».
Вычислить степень сжатия (исходный размер / сжатый размер).
Для строки "ABCDEFGHIJKLMNOPQRST" объяснить, почему RLE неэффективен, и предложить альтернативный метод сжатия.
Ожидаемый результат:
Сжатая строка (например, A3B3C4D3E5F8), степень сжатия, пояснение про альтернативы (Хаффман, LZ77).

Задание 8. Алгоритм поиска подстроки: Кнута-Морриса-Пратта (КМП)
Условие:
Текст: "ABABDABACDABABCABAB".
Образец (pattern): "ABABCABAB".
Требуется:
Построить префикс-функцию (π) для образца.
Выполнить поиск образца в тексте, используя алгоритм КМП.
Указать индексы (позиции) начала всех вхождений образца в текст (если есть).
Ожидаемый результат:
Таблица префикс-функции, позиции вхождений.

Задание 9. Динамическое программирование: задача о рюкзаке (0/1)
Условие:
Даны предметы:

Предмет	Вес	Ценность
A	2	3
B	3	4
C	4	5
D	5	8
E	6	9

Вместимость рюкзака = 8.
Требуется:
Заполнить таблицу динамического программирования (размер [n+1] × [capacity+1]) для задачи о рюкзаке 0/1.
Определить максимальную суммарную ценность, которую можно унести.
Вывести набор предметов, которые дают эту максимальную ценность.
Ожидаемый результат:
Таблица (можно описать словами или нарисовать), максимальная ценность, набор предметов.

Задание 10. Реализация приоритетной очереди (heap) и пирамидальная сортировка
Условие:
Дан массив: [4, 10, 3, 5, 1, 8, 7, 2, 9, 6].
Требуется:
Построить из этого массива max-кучу **(бинарную пирамиду) с помощью процедуры heapify (просеивания** вниз). Показать промежуточные шаги.
Выполнить пирамидальную сортировку (Heapsort): последовательно извлекать корень и восстанавливать

кучу. Записать отсортированный массив.
Оценить временную сложность пирамидальной сортировки в лучшем, худшем и среднем случаях.
Ожидаемый результат:
Промежуточные состояния кучи, итоговый отсортированный массив, оценка сложности (O(n log n) во всех случаях).

Конструирование программного обеспечения

Задание 1. TDD: разработка сервиса скидок

Условие:

Необходимо разработать метод calculateDiscount(customerType, amount), который возвращает сумму скидки.
Правила:
REGULAR → скидка 5%
GOLD → скидка 10%
PLATINUM → скидка 15%
NONE → скидка 0%
Скидка не может превышать 1000 рублей.
Требуется (применяя TDD):
Написать модульные-тесты (на JUnit или псевдокоде) для всех сценариев (включая граничные значения: amount = 0, amount = 20000).
Написать код метода, удовлетворяющий тестам.
Объяснить цикл Red-Green-Refactor и продемонстрировать его на примере хотя бы одного теста.
Ожидаемый результат:
Код тестов (3–5 штук), код метода, пояснение TDD-цикла.

Задание 2. Mockito: тестирование сервиса с внешней зависимостью

Условие:
Есть сервис OrderService, который зависит от PaymentGateway (внешний API оплаты). PaymentGateway.processPayment(order) может вернуть true (успех) или false (отказ). Нужно протестировать OrderService.placeOrder(order), который при успешной оплате сохраняет заказ в БД (через OrderRepository), а при отказе — не сохраняет.
Требуется:
Написать тест с использованием Mockito (или псевдокод), который мокирует PaymentGateway и OrderRepository.
Проверить, что при paymentGateway.processPayment() = true, метод orderRepository.save() вызывается ровно один раз.
Проверить, что при paymentGateway.processPayment() = false, метод orderRepository.save() не вызывается.
Ожидаемый результат:
Код теста (псевдокод или на Java с Mockito), пояснение, зачем нужны моки.

Задание 3. Интеграционное тестирование с TestContainers

Условие:
У вас есть Spring Boot приложение, использующее PostgreSQL. Репозиторий UserRepository имеет метод findByEmail(email). Нужно проверить, что он корректно возвращает пользователя из базы данных.
Требуется:
Написать (на Java или псевдокоде) интеграционный тест с использованием TestContainers, который:
поднимает контейнер с PostgreSQL;
применяет миграции (Liquibase/Flyway);
вставляет тестового пользователя;
выполняет findByEmail() и проверяет результат.
Объяснить, чем TestContainers отличается от эмуляции (H2 in-memory) и в каких случаях его использование обязательно.
Ожидаемый результат:
Код интеграционного теста (фрагмент), пояснение преимуществ TestContainers.

Задание 4. Рефакторинг: устранение длинного метода и дублирования
Условие:
Дан «плохой» код (псевдокод):
text
```
def processOrder(order):
    # валидация заказа
    if order.total <= 0:
        raise Exception("Invalid total")
    if order.items is None or len(order.items) == 0:
        raise Exception("No items")
    # расчёт доставки
    if order.total > 1000:
        deliveryCost = 0
    else:
        deliveryCost = 200
    # логирование
    print("Processing order: " + order.id)
    print("Total: " + order.total)
    print("Delivery: " + deliveryCost)
```
Требуется:
Выявить «запахи кода» (минимум 3).
Применить рефакторинг: извлечь методы (validateOrder, calculateDeliveryCost, logOrder).
Переписать код после рефакторинга (на любом языке или псевдокоде).
Ожидаемый результат:
Список запахов кода, итоговый код после рефакторинга.

Задание 5. Настройка CI/CD пайплайна (GitHub Actions)
Условие:
Для проекта на Python (FastAPI) необходимо настроить CI/CD пайплайн в GitHub Actions, который будет:
при каждом push в ветку main:
запускать модульные тесты (pytest);
проверять линтинг (flake8, black);
собирать Docker-образ;
пушить образ в GitHub Container Registry (ghcr.io);
деплоить на staging-сервер (по SSH, выполняя docker compose up -d).
Требуется:
Написать (в виде псевдокода или описать структуру) файл .github/workflows/ci-cd.yml.
Указать, какие secrets (секреты) потребуются.
Объяснить, как сделать так, чтобы деплой выполнялся только после успешного прохождения тестов и линтинга.
Ожидаемый результат:
Структура YAML-файла (2–4 job’a), список необходимых secrets, пояснение про needs.

Задание 6. JWT аутентификация: реализация и защита эндпоинта
Условие:
Необходимо реализовать JWT-аутентификацию для REST API (Spring Security).
Требования:
эндпоинт /login принимает логин/пароль, выдаёт JWT (access + refresh);
эндпоинт /api/orders доступен только пользователям с ролью USER;
эндпоинт /api/admin доступен только с ролью ADMIN;
JWT должен быть подписан (HS256 или RS256).
Требуется:
Написать (на псевдокоде или Java) фрагмент кода, реализующий:
генерацию JWT;
фильтр для проверки JWT.
Указать, где хранить refresh токен (http-only cookie или отдельный эндпоинт).
Объяснить, как можно отозвать JWT до истечения срока (стратегии).

Ожидаемый результат:
Код (фрагмент), пояснение про Refresh Token и отзыв JWT.

Задание 7. Реактивное программирование: WebFlux + backpressure
Условие:
Есть реактивный REST-эндпоинт, который возвращает поток (Flux) данных из базы данных (R2DBC). Клиент потребляет данные медленнее, чем сервер генерирует.
Требуется:
Написать (на псевдокоде или Java с WebFlux) код контроллера, возвращающего Flux<Item>.
Объяснить, как работает backpressure и как его реализовать через request(n).
Предложить способ ограничения размера буфера в памяти, чтобы избежать OutOfMemoryError.
Ожидаемый результат:
Код контроллера (фрагмент), пояснение про backpressure и onBackpressureBuffer.

Задание 8. GraphQL: решение проблемы N+1 запросов
Условие:
GraphQL-схема:
text
```
type User {
  id: ID!
  name: String!
  orders: [Order!]!
}

type Order {
  id: ID!
  total: Float!
}
```
Резолвер для User.orders делает отдельный SQL-запрос для каждого пользователя (проблема N+1). Нужно решить эту проблему с помощью DataLoader.
Требуется:
Объяснить, почему возникает проблема N+1 в GraphQL (в отличие от REST).
Написать (на Java/Kotlin с GraphQL Java или псевдокоде) реализацию DataLoader для пакетной загрузки заказов по списку userId.
Показать, как зарегистрировать DataLoader в контексте GraphQL.
Ожидаемый результат:
Пояснение проблемы, код DataLoader (фрагмент).

Задание 9. Нагрузочное тестирование с k6/JMeter
Условие:
Есть REST-эндпоинт GET /products (возвращает список товаров). Нужно провести нагрузочное тестирование, чтобы выяснить, сколько одновременных пользователей выдерживает сервер при условии, что p95 latency не превышает 500 мс.
Требуется:
Описать сценарий нагрузочного теста (например, количество виртуальных пользователей, длительность, этапы).
Написать (на псевдокоде) простой тест для k6 (или JMeter): 10 виртуальных пользователей в течение 1 минуты.
Объяснить, какие метрики нужно собирать (RPS, p95, p99, ошибки, загрузка CPU сервера).
Ожидаемый результат:
Сценарий, псевдокод теста, перечень метрик.

Задание 10. NoSQL: выбор модели данных для MongoDB
Условие:
Для блога нужно хранить: пользователей, статьи (посты), комментарии к статьям.
Основные запросы:
Получить статью по ID со всеми комментариями (отсортированными по дате).

Получить все статьи пользователя (только заголовки, без комментариев).
Добавить комментарий к статье.
Требуется:
Предложить схему данных для MongoDB (2 варианта: «вложенные документы» и «ссылки»).
Для каждого варианта оценить плюсы и минусы (количество запросов к БД, удобство обновления, ограничения по размеру документа).
Выбрать оптимальный вариант и обосновать выбор.
Ожидаемый результат:
Две схемы (JSON-like), анализ (таблица +/–), итоговый выбор с пояснением.

2.2. Оценочные средства для государственной итоговой аттестации (выпускной квалификационной работы)

2.2.1 Перечень тем выпускных квалификационных работ:

Примерная тематика выпускных квалификационных работ (для каждого вида деятельности, предусмотренного образовательной программой)

Направление деятельности	Примерная тематика
проектная деятельность:	- разработка стратегии проектирования, определение целей проектирования, критериев эффективности, ограничений применимости; - концептуальное проектирование информационных систем и технологий; - подготовка заданий на проектирование компонентов программного продукта и информационных систем и технологий на основе методологии программной инженерии; - выбор и внедрение в практику средств автоматизированного проектирования; - унификация и типизация проектных решений
производственно-технологическая деятельность:	авторское сопровождение процессов проектирования, разработки, внедрения и сопровождения программного обеспечения, информационных систем и технологий на производстве; авторское сопровождение процессов проектирования, разработки, внедрения и сопровождения программного продукта и технологий для производства;
организационно-управленческая деятельность:	организация взаимодействия коллективов разработчика и заказчика, принятие управленческих решений в условиях различных мнений; - нахождение компромисса между различными требованиями (стоимости, качества, сроков исполнения)как при долгосрочном, так и при краткосрочном планировании, нахождение

	оптимальных решений;
научно-исследовательская деятельность	сбор, анализ научно-технической информации, отечественного и зарубежного опыта по тематикеисследования; разработка и исследование теоретических и экспериментальных моделей объектов профессиональнойдеятельности в областях: машиностроение, приборостроение, наука, техника, образование, медицина,административное управление, юриспруденция, бизнес, предпринимательство, коммерция, менеджмент, банковские системы, безопасность информационныхсистем, управление технологическими процессами, механика, техническая физика, энергетика, ядерная энергетика, силовая электроника, металлургия, строительство, транспорт, железнодорожный транспорт, связь, телекоммуникации, управление инфокоммуникациями, почтовая связь, химическая промышленность, сельское хозяйство, текстильная илегкая промышленность, пищевая промышленность, медицинские и биотехнологии, горное дело, обеспечение безопасности подземных предприятий и производств, геология, нефтегазовая отрасль, геодезия и картография, геоинформационные системы, лесной комплекс, химико-лесной комплекс, экология, сфера сервиса, системы массовой информации, дизайн, медиаиндустрия, а такжена предприятиях различного профиля и всех видов деятельности в условиях экономики информационногообщества; разработка и исследование методик анализа, синтеза, оптимизации и прогнозирования качества процессов функционирования этих объектов; моделирование процессов и объектов на базе стандартных пакетов автоматизированного проектирования и исследований; постановка и проведение экспериментов по заданной методике и анализ результатов; анализ результатов проведения экспериментов, подготовка и составление обзоров, отчетов и научных публикаций;

	- прогнозирование развития информационных систем итехнологий;

4.2.2. Структура работы

Структура работы утверждена на заседании межинститутской базовой кафедры 05 февраля 2025 года, протокол №6.

Раздел 1. Введение

Формулировка задания	Контролируемые компетенции или их части		
	УК	ОПК	ПК
Задание 1. Обоснование актуальности, безопасности и экологичности темы ВКР	УК-1, УК-2, УК-4, УК-5 УК-6, УК-7, УК-8, УК-10		ПК-1
Задание 2. Формулировка цели и задач ВКР	УК-2, УК-8		ПК-1

Раздел 2. Предпроектное обследование

Формулировка задания	Контролируемые компетенции или их части		
	УК	ОПК	ПК
Задание 1. Изучить и описать общую характеристику предприятия (организации), систему или подсистему автоматизации.			ПК-1
Задание 2. Изучить и описать организационную структуру предприятия, группы функциональных задач и подзадач			ПК-1
Задание 3. Изучить и описать организационно-управленческую модель			ПК-1

Задание 4. Изучить и описать общую характеристику программируемой системы; задачи, решаемые с использованием программной системы (подсистемы), используемые технические средства, технические средства локальной сети (подсети), используемые системные программные средства (общего назначения и уникальные), организацию защиты информации			ОПК-3		ПК-2, ПК-4, ПК-6, ПК-7			
Задание 5. Изучить и описать пользователей программной системы (подсистемы), используемые базы данных, входные и выходные данные		УК-3						
Задание 6. Выявить и описать проблемные ситуации, возможные способы их решений. выбор проблемной ситуации для решения, и способа решения проблемной ситуации.	УК-1		ОПК-1					
Задание 7. Произвести выбор проблемной ситуации для решения, и способ решения проблемной ситуации.	УК-1		ОПК-1		ПК-1			
Задание 8. Сформулировать задачу проектирования в виде технического задания напроектирование программной системы (подсистемы).					ПК-1			

Листинги программ, блок-схемы алгоритмов и т. д.

Раздел 3. Реализация программного продукта и информационной системы (подсистемы)

Формулировка задания	Контролируемые компетенции или их части	
	ОПК	ПК

Задание 1. Обосновать выбор инструментальной среды разработки проектируемой программной системы (подсистемы).	ОПК-7	ПК-4
Задание 2. Обосновать выбор информационных технологий и программных средств, в том числе отечественного производства разработки программного продукта и информационной системы (подсистемы).	ОПК-2, ОПК-3	ПК-4, ПК-10
Задание 3. Описать реализацию программного продукта и проектируемой информационной системы (подсистемы) в выбранной инструментальной среде разработки	ОПК-5, ОПК-6, ОПК-7, ОПК-8	ПК-4, ПК-6, ПК-8, ПК-9, ПК-10
Задание 4. Описать и проанализировать характеристики разработанной программной системы (подсистемы),степень соответствия разработанной программной системы (подсистемы) требованиям технического задания на проект	ОПК-5, ОПК-6, ОПК-7, ОПК-8	ПК-6, ПК-7, ПК-8, ПК-9, ПК-10

Графический материал: Листинги программ, блок-схемы алгоритмов и т. д.

Раздел 4. Информационное и программное обеспечение

Формулировка задания	Контролируемые компетенции или их части	
	ОПК	ПК
Задание 1. Документирование разработанного программного продукта. Разработка инструкции по эксплуатации.	ОПК-4	ПК-3, ПК-4, ПК-5
Задание 2. Тестирование программной системы, – информационной системы (подсистемы)		ПК-8

Задание 3. Обоснование минимальных и достаточных требований к техническому обеспечению.	ОПК-7	ПК-7

Раздел 5. Технико-экономическое обоснование.

Формулировка задания	Контролируемые компетенции или их части	
	УК	ПК
Задание 1. Выполнить технико-экономическое обоснование проекта информационной системы	УК-9	ПК-12

Раздел 6. Безопасность и экологичность проекта.

Формулировка задания	Контролируемые компетенции или их части	
	УК	ПК
Задание 1. Выполнить анализ основных опасных и вредных факторов на рабочем месте оператора информационной системы (подсистемы)	УК-8	ПК-1
Задание 2. Выполнить анализ и описать общие мероприятия по обеспечению безопасности жизнедеятельности на рабочем месте оператора информационной системы	УК-8	ПК-1

Раздел 7. Заключение. Подготовка доклада и презентации для защиты ВКР

Формулировка задания	Контролируемые компетенции или их части	
	ОПК	ПК
Задание 1. Сформулировать выводы ВКР: соответствие заданию; проанализировать правильность выбранной модели и адекватность использованных методов обработки, анализа и синтеза результатов профессиональных исследований.	ОПК-2	ПК-1
Задание 2.Подготовить доклад и презентацию для защиты ВКР		ПК-5

5. Методические материалы, определяющие процедуры оценивания результатов освоения образовательной программы.

5.1 Методические материалы, определяющие процедуры оценивания результатов освоения образовательной программы на итоговом (государственном) экзамене

Процедура проведения государственного экзамена осуществляется в соответствии с Положением о порядке проведения государственной итоговой аттестации по образовательным программам высшего образования – программам бакалавриата, программам специалитета и программам магистратуры в СКФУ и Положением о порядке проведения государственной итоговой аттестации по образовательным программам высшего образования – программам аспирантуры, ординатуры в СКФУ.

В экзаменационный билет включаются 2 теоретических вопроса и задача.

Каждый обучающийся самостоятельно выбирает экзаменационный билет один раз посредством произвольного извлечения. Номер билета фиксируется секретарем ГЭК в соответствующем протоколе.

На подготовку к ответу на экзаменационный билет обучающемуся отводится: (как правило, 30 минут (для технических направлений подготовки (специальностей) – до 1 часа)).

При подготовке обучающийся имеет право пользоваться программой государственного экзамена, а также с разрешения ГЭК – справочной литературой

При проверке практического задания, оцениваются: умение понимать содержание вопроса; умение применять на практике полученные знания и навыки; полнота ответов; правильность ответов и решений.

5.2 Методические материалы, определяющие процедуры оценивания результатов освоения образовательной программы на защите выпускной квалификационной работы

На каждом этапе осуществляется текущий контроль за процессом формирования компетенций. Предлагаемые обучающемуся задания позволяют проверить усвоенные компетенции: УК-1, УК-2, УК-3, УК-4, УК-5, УК-6, УК-7, УК-8, УК-9, УК-10, ОПК-1, ОПК-2, ОПК-3, ОПК-4, ОПК-5, ОПК-6, ОПК-7, ОПК-8, ПК-1, ПК-2, ПК- 3, ПК-4, ПК-5, ПК-6, ПК-7, ПК-8, ПК-9, ПК-10. Описание показателей для этих компетенций представлено в таблице 3.1.

При защите выпускной квалификационной работы оцениваются:

- актуальность и новизна темы выпускной квалификационной работы;
- полнота выполнения задания;
- грамотность изложения;
- использование CASE-средств проектирования программного продукта и ИС;
- использование унифицированного языка моделирования (UML) при построении модели программного продукта и ИС;

- соблюдение требований гостов при оформлении текстуальной части пояснительной записки;
- соблюдение требований гостов при оформлении рисунков;
- соблюдение требований гостов при оформлении таблиц;
- соблюдение требований гостов при оформлении приложений;
- соблюдение требований гостов при оформлении чертежей;
- полнота раскрытия темы выпускной квалификационной работы;
- соответствие доклада содержанию пояснительной записки;
- культура речи и умение владеть собой;
- использование вспомогательных материалов в ходе доклада (текста доклада и др.);
- умение формулировать выводы и акцентировать внимание на узловых моментах выполненной работы.

Критерии оценки выпускных квалификационных работ:

Итоговая оценка защиты ВКР вычисляется по формуле:

$$Оценка = \frac{К_1 \cdot К_2 \cdot К_3 \cdot К_4 \cdot К_5}{5}, \quad (5.1)$$

где $К_1$ – критерий оценки содержание пояснительной записки;

$К_2$ – критерий оценки качества оформления пояснительной записки и плакатов;

$К_3$– критерий оценки качества и содержания доклада на защите выпускных квалификационных работ

$К_4$ – критерий оценки полноты и правильности ответов на вопросы членов ГАК;

$К_5$– критерий оценки качества разработки программного продукта и умение продемонстрировать его работоспособность.

Результаты вычислений по формуле (1) округляются до ближайшего целого.

Каждый из коэффициентов $К_2 .. К_5$, использованных в форму (5.1), может принимать значения в интервале от нуля до пяти.

Таким образом, максимальное значение, конторе может принять числительвыражения (1) равно 25.

Значение коэффициентов $K_1, K_2 \ldots K_5$ рассчитывается в соответствии с данными таблицы 5.2 по формуле:

$$K_i = \sum_{i=1}^{5} K_{ij}, \quad i=1,2,\ldots 5$$

Оценка «отлично» выставляется студенту, если вычисленное поформуле (5.1) значение равно или больше, чем 4,5.

Оценка «хорошо» выставляется студенту, если вычисленное поформуле (5.1) значение больше, чем 3,5 и меньше, чем 4,5.

Оценка «удовлетворительно» выставляется студенту, если вычисленное по формуле (5.1) значение больше, чем 2,5 и меньше, чем 3,5.

Оценка «неудовлетворительно» выставляется студенту, если вычисленное по формуле (5.1) меньше, чем 2,5.