Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Магнитогорский государственный технический университет

им. Г.И. Носова»

Многопрофильный колледж

**Отчет**

**по практике по профилю специальности**

**по специальности 09.02.07 Информационные системы и программирование (базовой подготовки).**

**Квалификация: программист**

**ПМ. 02 Осуществление интеграции программных модулей**

|  |  |
| --- | --- |
|  | Студента гр. ИСпПК-22-7\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Дениса Евгеньевича Гусева\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(И.О. Фамилия)* |
| М.П. | Организация: МП трест «Водоканал» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(наименование места прохождения практики)*  Руководитель практики от организации  Роман Игоревич Задорожный\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(И.О. Фамилия)*  Руководитель практики от МпК  Елена Ревгатевна Рылова\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(И.О. Фамилия)* |

Магнитогорск, 2024

**ВНУТРЕННЯЯ ОПИСЬ**

**документов, находящихся в отчете**

|  |
| --- |
| Обучающегося гр. ИСпПК-22-7  Дениса Евгеньевича Гусева  *(И.О. Фамилия)* |

|  |  |  |
| --- | --- | --- |
| **№п/п** | **Наименование документа** | **Стр.** |
|  | Задание на практику | 3 |
|  | Аттестационный лист | 8 |
|  | Отчет о выполнении заданий по практике | 12 |
|  | Дневник по практике | 30 |

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Магнитогорский государственный технический университет**

**им. Г.И. Носова»**

(ФГБОУ ВО «МГТУ им. Г.И. Носова»)

Многопрофильный колледж

**ЗАДАНИЕ**

**на производственную (по профилю специальности) практику**

Обучающегося гр. ИСпПК-22-7 Дениса Евгеньевича Гусева

*(И.О. Фамилия)*

**09.02.07 Информационные системы и программирование (базовой подготовки) \_\_\_\_\_\_\_\_\_\_**

*(шифр и наименование специальности)*

**ПМ. 02 Осуществление интеграции программных модулей\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(индекс и наименование профессионального модуля)*

**Цели практики:**

Приобретение и углубление практического опыта в рамках формируемых компетенций

|  |  |  |
| --- | --- | --- |
| **Код ПК/ОК** | **Наименование** | **Практический опыт** |
| ПК 2.3 | Выполнять отладку программного модуля с использованием специализированных программных средств. | ПО2. Отлаживать программные модули.  ПО6. Инспектировать разработанные программные модули на предмет соответствия стандартам кодирования.  У03.3 определять и выстраивать траектории профессионального развития и самообразования;  У03.4 применять исследовательские приемы и навыки, чтобы быть в курсе последних отраслевых решений;  У06.1 отстаивать активную гражданско-патриотическую позицию;  У06.3 проявлять базовые общечеловеческие, культурные и национальные ценности российского государства в современном сообществе;  У07.1 соблюдать нормы экологической безопасности;  У07.3 использовать энергосберегающие и ресурсосберегающие технологии в профессиональной деятельности по специальности;  У07.4 оценивать чрезвычайную ситуацию;  У07.5 составлять алгоритм действий при чрезвычайной ситуации и определять необходимые ресурсы для её устранения; |
| ОК 03 | Планировать и реализовывать собственное профессиональное и личностное развитие. |
| ОК 06 | Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей. |
| ОК 07 | Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях. |
| ПК 2.4 | Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения. | ПО4. Разрабатывать тестовые наборы (пакеты) для программного модуля.  ПО5. Разрабатывать тестовые сценарии программного средства.  ПО6. Инспектировать разработанные программные модули на предмет соответствия стандартам кодирования.  У01.1 распознавать задачу и/или проблему в профессиональном и/или социальном контексте;  У01.2 анализировать задачу и/или проблему и выделять её составные части;  У03.3 определять и выстраивать траектории профессионального развития и самообразования;  У03.4 применять исследовательские приемы и навыки, чтобы быть в курсе последних отраслевых решений;  У04.2 взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности;  У04.3 понимать требования и оправдывать ожидания клиентов/работодателя;  У04.4 реагировать на запросы клиентов/руководства лично и опосредованно;  У09.1 применять средства информационных технологий для решения профессиональных задач;  У09.2 использовать современное программное обеспечение;  У09.3 проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий; |
| ОК 01 | Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам. |
| ОК 03 | Планировать и реализовывать собственное профессиональное и личностное развитие. |
| ОК 04 | Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами. |
| ОК 09 | Использовать информационные технологии в профессиональной деятельности. |
| ПК 2.5 | Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования. | ПО6. Инспектировать разработанные программные модули на предмет соответствия стандартам кодирования.  У03.3 определять и выстраивать траектории профессионального развития и самообразования;  У03.4 применять исследовательские приемы и навыки, чтобы быть в курсе последних отраслевых решений;  У08.1 использовать физкультурно-оздоровительную деятельность для укрепления здоровья, достижения жизненных и профессиональных целей;  У08.3 пользоваться средствами профилактики перенапряжения характерными для данной специальности;  У09.1 применять средства информационных технологий для решения профессиональных задач;  У09.2 использовать современное программное обеспечение;  У09.3 проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий;  У11.1 применять знания по финансовой грамотности для профессиональной деятельности и в повседневной жизни;  У11.2 выявлять достоинства и недостатки коммерческой идеи;  У11.3 презентовать идеи открытия собственного дела в профессиональной деятельности; |
| ОК 03 | Планировать и реализовывать собственное профессиональное и личностное развитие. |
| ОК 08 | Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности. |
| ОК 09 | Использовать информационные технологии в профессиональной деятельности. |
| ОК 11 | Планировать предпринимательскую деятельность в профессиональной сфере. |

**Задание на практику**

|  |  |
| --- | --- |
| Практический опыт, умения | Виды работ, выполняемых в период практики в рамках формируемых компетенций |
| ПО 2, ПО 6;  У 03.3, У 03.4, У 06.1, У 06.3, У 07.1, У 07.3, У 07.4, У 07.5. | 1.Применение основных методов для получения кода с заданной функциональностью и степенью качества.  2. Выполнение отладки программных модулей, используя методы и инструменты условной компиляции. |
| ПО 4, ПО 5, ПО 6;  У 01.1, У 01.2, У 03.3, У 03.4, У 04.2, У 04.3, У 09.1, У 09.2, У 09.3 | 1.Выполнение тестирования интеграции с помощью **HTTP-сервисов.**  2.Применение основных методов и способов идентификации сбоев и ошибок при интеграции приложений.  3.Применение встроенных и основных специализированных инструментов анализа качества программных продуктов. |
| ПО 6;  У 03.3, У 03.4, У 08.1, У 08.3, У 09.1, У 09.2, У 09.3, У 11.1, У 11.2, У11.3 | 1. Использование основных методов для получения кода с заданной функциональностью и степенью качества.  2. Анализ проектной и технической документации на  предмет соответствия стандартам |

Место практики МП трест «Водоканал»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Задание на практику**

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Содержание работ на практике** | **Примерные сроки выполнения** |
|  | Ознакомиться с организацией работы, структурой и деятельностью структурных подразделений организации по месту прохождения практики. | первая неделя |
|  | Прохождение техники безопасности на производственном участке. | в течение практики |
|  | Применение основных методов для получения кода с заданной функциональностью и степенью качества | в течение практики |
|  | Выполнение отладки программных модулей, используя методы и инструменты условной компиляции. | в течение практики |
|  | Выполнение тестирования интеграции с помощью **HTTP-сервисов.** | в течение практики |
|  | Применение основных методов и способов идентификации сбоев и ошибок при интеграции приложений | в течение практики |
|  | Применение встроенных и основных специализированных инструментов анализа качества программных продуктов. | в течение практики |
|  | Использование основных методов для получения кода с заданной функциональностью и степенью качества | в течение практики |
|  | Анализ проектной и технической документации на  предмет соответствия стандартам | в течение практики |
|  | Оформить документы для отчета по практике | последняя неделя |
|  | Подготовить и сдать отчет по практике | последняя неделя |

Примерный перечень документов, прилагаемых в качестве приложения к отчету по практике:

1. Нормативно-техническая документация

Руководитель практики от МпК Е. Р. Рылова *\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(подпись)*

«02» сентября 2024г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Магнитогорский государственный технический университет**

**им. Г.И. Носова»**

(ФГБОУ ВО «МГТУ им. Г.И. Носова»)

Многопрофильный колледж

**АТТЕСТАЦИОННЫЙ ЛИСТ ПО ПРАКТИКЕ**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­­\_\_\_\_\_** Дениса Евгеньевича Гусева**\_\_\_­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(И.О.Фамилия)*

обучающийся на 4 курсе специальности 09.02.07 Информационные системы и\_программирование (базовой подготовки) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(шифр и наименование специальности)*

успешно прошел по профилю специальности практику \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

по профессиональному модулю: **ПМ.02 Осуществление интеграции программных модулей\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(индекс и наименование профессионального модуля)*

в объеме 72 часа с «02» сентября 2024 г. по «14» сентября 2024г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

в организации \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(наименование организации, юридический адрес)*

**Цели практики:**

Приобретение и углубление практического опыта в рамках формируемых компетенций

|  |  |  |
| --- | --- | --- |
| **Код ПК/ОК** | **Наименование** | **Практический опыт** |
| ПК 2.3 | Выполнять отладку программного модуля с использованием специализированных программных средств. | ПО2. Отлаживать программные модули.  ПО6. Инспектировать разработанные программные модули на предмет соответствия стандартам кодирования.  У03.3 определять и выстраивать траектории профессионального развития и самообразования;  У03.4 применять исследовательские приемы и навыки, чтобы быть в курсе последних отраслевых решений;  У06.1 отстаивать активную гражданско-патриотическую позицию;  У06.3 проявлять базовые общечеловеческие, культурные и национальные ценности российского государства в современном сообществе;  У07.1 соблюдать нормы экологической безопасности;  У07.3 использовать энергосберегающие и ресурсосберегающие технологии в профессиональной деятельности по специальности;  У07.4 оценивать чрезвычайную ситуацию;  У07.5 составлять алгоритм действий при чрезвычайной ситуации и определять необходимые ресурсы для её устранения; |
| ОК 03 | Планировать и реализовывать собственное профессиональное и личностное развитие. |
| ОК 06 | Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей. |
| ОК 07 | Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях. |
| ПК 2.4 | Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения. | ПО4. Разрабатывать тестовые наборы (пакеты) для программного модуля.  ПО5. Разрабатывать тестовые сценарии программного средства.  ПО6. Инспектировать разработанные программные модули на предмет соответствия стандартам кодирования.  У01.1 распознавать задачу и/или проблему в профессиональном и/или социальном контексте;  У01.2 анализировать задачу и/или проблему и выделять её составные части;  У03.3 определять и выстраивать траектории профессионального развития и самообразования;  У03.4 применять исследовательские приемы и навыки, чтобы быть в курсе последних отраслевых решений;  У04.2 взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности;  У04.3 понимать требования и оправдывать ожидания клиентов/работодателя;  У04.4 реагировать на запросы клиентов/руководства лично и опосредованно;  У09.1 применять средства информационных технологий для решения профессиональных задач;  У09.2 использовать современное программное обеспечение;  У09.3 проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий; |
| ОК 01 | Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам. |
| ОК 03 | Планировать и реализовывать собственное профессиональное и личностное развитие. |
| ОК 04 | Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами. |
| ОК 09 | Использовать информационные технологии в профессиональной деятельности. |
| ПК 2.5 | Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования. | ПО6. Инспектировать разработанные программные модули на предмет соответствия стандартам кодирования.  У03.3 определять и выстраивать траектории профессионального развития и самообразования;  У03.4 применять исследовательские приемы и навыки, чтобы быть в курсе последних отраслевых решений;  У08.1 использовать физкультурно-оздоровительную деятельность для укрепления здоровья, достижения жизненных и профессиональных целей;  У08.3 пользоваться средствами профилактики перенапряжения характерными для данной специальности;  У09.1 применять средства информационных технологий для решения профессиональных задач;  У09.2 использовать современное программное обеспечение;  У09.3 проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий;  У11.1 применять знания по финансовой грамотности для профессиональной деятельности и в повседневной жизни;  У11.2 выявлять достоинства и недостатки коммерческой идеи;  У11.3 презентовать идеи открытия собственного дела в профессиональной деятельности; |
| ОК 03 | Планировать и реализовывать собственное профессиональное и личностное развитие. |
| ОК 08 | Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности. |
| ОК 09 | Использовать информационные технологии в профессиональной деятельности. |
| ОК 11 | Планировать предпринимательскую деятельность в профессиональной сфере. |

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| Практический опыт, умения | Виды и объем работ, выполненных обучающимися во время практики в рамках формируемых компетенций | Зачтено/  не зачтено |
| ПО 2, ПО 6;  У 03.3, У 03.4, У 06.1, У 06.3, У 07.1, У 07.3, У 07.4, У 07.5. | 1. Ознакомиться с организацией работы, структурой и деятельностью структурных подразделений организации по месту прохождения практики. |  |
| 2. Прохождение техники безопасности на производственном участке. |  |
| 3.Применение основных методов для получения кода с заданной функциональностью и степенью качества. |  |
| 4. Выполнение отладки программных модулей, используя методы и инструменты условной компиляции. |  |
| ПО 4, ПО 5, ПО 6;  У 01.1, У 01.2, У 03.3, У 03.4, У 04.2, У 04.3, У 09.1, У 09.2, У 09.3 | 1.Выполнение тестирования интеграции с помощью **HTTP-сервисов.**  2.Применение основных методов и способов идентификации сбоев и ошибок при интеграции |  |
| приложений.  3.Применение встроенных и основных специализированных инструментов анализа качества программных продуктов. |  |
| ПО 6;  У 03.3, У 03.4, У 08.1, У 08.3, У 09.1, У 09.2, У 09.3, У 11.1, У 11.2, У11.3 | 1. Использование основных методов для получения кода с заданной функциональностью и степенью качества. |  |
| 2. Анализ проектной и технической документации на предмет соответствия стандартам |  |

Руководитель практики от МпК Елена Ревгатевна Рылова\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(И.О. Фамилия)*

«14» сентября 2024г.

Руководитель практики от организации Роман Игоревич Задорожный\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(И.О. Фамилия)*

«14» сентября 2024г.

М.П.

**1. Ознакомиться с организацией работы, структурой и деятельностью структурных подразделений организации по месту прохождения практики.**

Компания МП трест «Водоканал», основанная в марте 1929 года, занимается водоснабжением и водоотведением в городе Магнитогорске, обеспечивая жителей и предприятия города питьевой, технической водой и системой канализации. Компания состоит из нескольких подразделений, каждое из которых выполняет определенные функции:

1. Техническое отделение:

* Проведение технического обслуживания и ремонта оборудования и сетей водоснабжения и водоотведения.
* Планирование и координация работ по модернизации и развитию водопроводных и канализационных систем.
* Участие в разработке проектов строительства и реконструкции водоочистных сооружений.
* Организация контроля за качеством воды и соблюдением санитарно-гигиенических норм.
* Разработка и внедрение мероприятий по снижению уровня потерь воды и повышению эффективности работы сетей водоснабжения и водоотведения.
* Сотрудничество с внешними организациями и контроль за выполнением условий договоров и соглашений.
* Участие в разработке программ по совершенствованию работы водоканала и повышению качества предоставляемых услуг.

1. Отделение по обслуживанию и ремонту водопроводов

* Обслуживание и проверка состояния водопроводных сетей.
* Выявление и устранение утечек воды.
* Ремонт и замена дефектных участков водопроводов.
* Профилактическое обслуживание и чистка водопроводных сетей.
* Проведение диагностики и испытаний водопроводных систем.
* Установка и замена сантехнического оборудования.
* Консультирование и информирование жителей о правилах эксплуатации водопровода.

1. Отделение по обслуживанию и ремонту канализации

* Очистка и прочистка канализационных колодцев и сточных труб.
* Ремонт и замена дефектных участков канализационных сетей.
* Установка и обслуживание насосных станций и фекальных насосов.
* Проведение диагностики состояния канализационных систем с помощью специализированного оборудования.
* Предоставление консультаций по вопросам эксплуатации и обслуживания канализации.
* Проведение профилактических работ для предотвращения аварий и аварийных ситуаций.

1. Эксплуатационно-диспетчерская служба

* Контроль за работой водопроводных и канализационных сетей.
* Организация и координация процесса водоснабжения и водоотведения.
* Мониторинг состояния водопроводных и канализационных сетей.
* Организация аварийно-восстановительных работ в случае поломок или аварий.
* Планирование и проведение профилактических работ на водопроводных и канализационных сетях.
* Взаимодействие с другими службами и организациями для обеспечения надлежащего функционирования коммунальной инфраструктуры.
* Контроль качества воды и обеспечение соблюдения санитарных норм.
* Учет и анализ потребления воды и канализационных услуг.
* Проведение мониторинга и анализа показателей работы сетей для оптимизации их функционирования.
* Работа с документацией и отчетностью по деятельности службы.

МП трест "Водоканал" с каждым годом переходит на более усовершенствованные ИТ-технологии и языки программирования для автоматизации своих процессов управления и обслуживания сетей водоснабжения и водоотведения. В компании используются разнообразные способы программирования, которые включают в себя:

* Системы управления базами данных (например, SQL)
* Геоинформационные системы (ГИС) для визуализации и анализа сетей водоснабжения
* Языки программирования, такие как Python, Java, C ++ для разработки прикладных программ и скриптов
* Системы управления контентом для хранения и обмена документами и информацией
* Средства автоматизации и мониторинга процессов (например, SCADA системы)
* Интеграция с облачными сервисами для повышения доступности и безопасности данных

**2. Прохождение техники безопасности на производственном участке.**

В рамках прохождения данного модуля, была предоставлена обучающая программа по технике безопасности на производственном участке компании МП трест «Водоканал». Программа включала в себя:

* Знакомство с правилами и нормами техники безопасности на производственном участке.
* Обучение по обращению с техническими устройствами и оборудованием.
* Инструктаж по правильному поведению на рабочем месте и в экстренных ситуациях.
* Обучение по методам предотвращения травм.

Данная программа предназначена для обеспечения безопасности работников и снижения риска возникновения аварий и несчастных случаев на производственном участке. Важно, чтобы каждый работник, в том числе программист, был владельцем навыков и знаний по технике безопасности, чтобы обеспечить безопасные условия труда.

Первая часть программы была посвящена общим принципам техники безопасности на производственном участке. В рамках этой части были рассмотрены основные правила пожарной безопасности, профилактики травм на рабочем месте, использования средств индивидуальной защиты и другие важные аспекты.

Далее были рассмотрена правила при нахождении вблизи технических устройств. Рабочее место программиста должно быть оборудовано следующим образом:

* Расположено на устойчивой и прочной поверхности, чтобы избежать его падения или сдвига во время работы.
* Компьютер и другая техническая оснастка должны быть установлены на надежном столе или стойке, чтобы избежать их повреждения и падения.
* Кабели и провода должны быть уложены таким образом, чтобы не создавали препятствий и не создавали риск споткнуться.
* Рабочее место должно быть освещено достаточно ярко, чтобы избежать напряжения глаз и усталости.
* Кресло программиста должно быть удобным, с возможностью регулировки высоты и наклона спинки для поддержки правильной осанки.
* Программист должен регулярно делать перерывы для разминки и отдыха, чтобы избежать перенапряжения мышц и глаз.
* Рабочее место должно быть чистым и организованным, чтобы избежать случайных травм и уменьшить риск возгорания от перегрева электроники.
* Программист должен иметь доступ к необходимым средствам защиты, таким как антивирусное программное обеспечение и парольные устройства, чтобы защитить конфиденциальные данные и информацию.
* В случае работы с электрическим оборудованием, программист должен быть ознакомлен с правилами пожарной безопасности и использовать защитные средства, такие как перчатки и очки.
* Кроме того, необходимо обеспечить хорошую вентиляцию на рабочем месте, чтобы избежать перегрева компьютера и соблюсти правила пожарной безопасности.

Действия при пожаре:

* Вызовите пожарную службу по номеру экстренных служб (101 или другой соответствующий номер в вашем регионе).
* Уведомите всех находящихся в здании о пожаре и призовите их к безопасной эвакуации.
* Используйте пожарный оповещатель, если он установлен в здании, чтобы предупредить о пожаре.
* Используйте огнетушитель, если пожар только начался и вы чувствуете себя достаточно уверенно в его применении.
* Если пламя слишком сильное, не пытайтесь тушить его самостоятельно, эвакуируйтесь из здания.
* Используйте аварийные выходы для эвакуации и никогда не используйте лифты в здании при пожаре.
* При покидании здания накройте лицо плотной тканью и каской, чтобы защититься от дыма и огня.
* Если пути эвакуации заблокированы, уберите препятствия или попробуйте найти другой путь для покидания здания.
* Соберитесь в безопасном месте за пределами здания и дождитесь приезда пожарной службы.
* После эвакуации ни в коем случае не возвращайтесь в здание, пока не будет дано разрешение пожарными спасателями.

Соблюдение всех этих требований поможет обеспечить безопасность и комфорт на рабочем месте программиста.

По окончании обучающей программы проводился тест, который позволил проверить уровень усвоения материала участниками.

В целом, обучающая программа по технике безопасности на производственном участке в компании МП трест «Водоканал» оптимально структурирована и охватывает все важные аспекты безопасности на рабочем месте. Участники получили необходимые знания и навыки для обеспечения безопасной работы на производстве.

Таким образом, прохождение модуля по соблюдению техники безопасности в компании МП трест «Водоканал» было полезным и позволило участникам приобрести необходимые навыки и знания для безопасной работы на производстве.

**3.** **Применение основных методов для получения кода с заданной функциональностью и степенью качества.**

1. Методологии разработки программного обеспечения

Одним из первых шагов в создании качественного кода является выбор подходящей методологии разработки. Выбор подхода зависит от характера проекта, команды и ожиданий клиента. Каждая методология имеет свои плюсы и минусы, и правильный выбор может значительно повлиять на качество итогового кода. Наиболее популярные среди них включают:

* Agile: гибкие методологии, которые фокусируются на итеративном и инкрементальном развитии, позволяют быстрее адаптироваться к изменяющимся требованиям и обеспечивают более высокое качество продукта за счет постоянного тестирования и обратной связи.
* Waterfall (водопад): традиционная последовательная модель, при которой каждая стадия разработки должна быть завершена перед переходом к следующей. Этот метод может быть эффективен для проектов с четко определенными требованиями, но менее гибок к изменениям.
* Scrum: структура Agile, которая организует работу в спринты, позволяя командам фокусироваться на определённых задачах и регулярно оценивать прогресс.

1. Проектирование архитектуры приложения

Архитектурное проектирование играет ключевую роль в создании функционального и качественного кода. Применение архитектурных паттернов, таких как MVC (Model-View-Controller), MVP (Model-View-Presenter) или MVVM (Model-View-ViewModel), помогает организовать код таким образом, чтобы разные компоненты системы могли взаимодействовать независимо друг от друга.

Эффективная архитектура улучшает масштабируемость и поддерживаемость кода, позволяя командам легче вносить изменения и добавлять новый функционал. Сложные системы часто требуют разработки микросервисов, что требует дополнительных усилий, но также позволяет повысить уровень разделения ответственности и уменьшить время загрузки компонентов.

1. Использование языков программирования и фреймворков

Выбор языка программирования и соответствующих фреймворков также критически важен. Каждый язык имеет свои особенности, преимущества и недостатки. Например, Python известен своей простотой и удобством для разработки, тогда как Java может предоставить высокую производительность и надежность в многопоточных приложениях.

Фреймворки, такие как Django для Python или Spring для Java, предлагают разработчикам заранее определенные структуры и инструменты, что ускоряет процесс разработки и улучшает качество кода. Использование библиотек и модулей также позволяет сократить время, необходимое для написания кода, так как разработчики могут использовать проверенные и протестированные решения.

1. Коды и стандарты

Следование кодовым стандартам и лучшим практикам разработки, таким как PEP8 для Python или Java Coding Conventions, играет важную роль в поддержании качества кода. Стандартизированные подходы к именованию переменных, структурированию классов и написанию документации помогают сделать код более читабельным и упрощают его поддержку.

Кроме того, применение принципов SOLID позволяет создать хорошо организованный и легко модифицируемый код, что в свою очередь улучшает функциональность приложения и облегчает процесс тестирования.

1. Тестирование и обеспечение качества

Тестирование является важным этапом разработки, позволяющим обнаружить и исправить дефекты на ранних стадиях. Включение юнит-тестирования, интеграционного тестирования и системного тестирования в разработку помогает предотвратить множественные проблемы и повысить уровень доверия к продукту.

Методы автоматизированного тестирования, такие как TDD (Test-Driven Development) и BDD (Behavior-Driven Development), позволяют разработать код, соответствующий требованиям перед его реализацией. Эти подходы создают более высокое качество кода, так как ориентированы на функциональные требования и избегают недоразумений на стадии реализации.

1. Рефакторинг и код-ревью

Регулярный рефакторинг позволяет улучшать структурное качество и производительность кода, не меняя при этом его функциональность. Это помогает убрать избыточный код, уменьшить сложность функций и повысить общую стабильность приложения.

Кроме того, практика код-ревью, когда другие разработчики проверяют написанный код, способствует выявлению возможных ошибок и улучшению качества кода. Обмен знаниями в команде также способствует внедрению лучших практик и улучшению навыков программистов, что в итоге позитивно сказывается на итоговом качестве продукта.

Таки образом применение вышеперечисленных методов и подходов позволяет создать код с заданной функциональностью и высоким качеством. Выбор подходящей методологии, продуманное архитектурное проектирование, соблюдение кодовых стандартов, а также активное тестирование и код-ревью являются основными компонентами успешной разработки программного обеспечения.

Эти практики помогают не только создавать качественный код, но и значительно упрощают его поддержку и масштабирование в будущем. В итоге, итоговым результатом является не только функциональный, но и устойчивый, читаемый и легко поддерживаемый продукт, соответствующий ожиданиям пользователей и требованиям бизнеса.

**4.** **Выполнение отладки программных модулей, используя методы и инструменты условной компиляции.**

Отладка программного обеспечения — это одна из ключевых стадий разработки, направленная на выявление и устранение ошибок, а также на улучшение качества кода. В современных условиях разработки программных приложений использование методов условной компиляции становится обоснованным подходом, позволяющим упростить процесс отладки, сократить время на тестирование и повысить качество кода.

Условная компиляция — это механизм, который позволяет включать или исключать определенные фрагменты кода во время компиляции в зависимости от заданных условий. Этот метод весьма полезен в процессе отладки, так как позволяет разработчикам гибко управлять содержимым программных модулей, не изменяя их логическую структуру.

1. Основные принципы условной компиляции

Условная компиляция осуществляется с использованием директив препроцессора, таких как #ifdef, #ifndef, #else, #elif и #endif. Эти директивы позволяют создавать различные конфигурации программного кода для различных сред выполнения, улучшая читаемость и управляемость исходного кода. Основные этапы работы с условной компиляцией включают:

* Определение условий компиляции — разработчики могут задать условия, при выполнении которых определенные участки кода будут включаться или исключаться из сборки. Например, это может быть полезно для включения отладочной информации или дополнительных логов во время разработки.
* Создание конфигураций — различные конфигурации могут иметь разные уровни детализации и функциональной полноты. Это позволяет добиться максимальной гибкости, а также адаптировать программу под различные платформы и условия эксплуатации.
* Управление зависимостями — с помощью условной компиляции можно контролировать, какие модули и функции будут доступны в разных сборках. Это позволяет уменьшить количество ошибок, связанных с отсутствием необходимых зависимостей в ходе выполнения программы.

1. Преимущества условной компиляции в отладке

* Сокращение объема кода: Когда разработчик добавляет отладочные сообщения или временные решения для исправления ошибок, это может привести к «забиванию» основного кода. Условная компиляция позволяет избежать этого, исключая неиспользуемые части кода из компиляции.
* Гибкость: Условная компиляция предоставляет возможность легко переключаться между различными режимами работы программы. Благодаря этому разработчик может протестировать несколько вариантов реализации одной и той же функции, не изменяя основной код.
* Упрощение анализа ошибок: Используя условную компиляцию, можно создавать специальные сборки программ, в которых будут активированы только те участки кода, которые необходимы для диагностирования конкретной проблемы. Это позволяет изолировать ошибки и проводить более эффективную отладку.
* Повышение читаемости кода: Код, включающий условные компиляции, может быть значительно более структурированным и читаемым. Разработчик может выделить код, который используется исключительно для отладки, что позволяет другим членам команды легче ориентироваться в основном функционале.

1. Инструменты для условной компиляции

Множество современных языков программирования предоставляет встроенные инструменты и библиотеки для условной компиляции. Например:

* C/C++: Использует препроцессоры для управления условной компиляцией.
* C#: Предоставляет директивы #define и #if для создания условных сборок.
* Java: Подход относительно менее распространен, но возможен с использованием систем сборки, таких как Maven или Gradle, которые позволят управлять профилями сборки.
* Python: Хотя в нем нет прямой условной компиляции, реализация аналогичного подхода возможна с помощью использования модулей и условий выполнения кода.

Условная компиляция — это инструмент, который значительно упрощает процессы отладки и разработки программного обеспечения. С её помощью разработчики могут эффективно управлять видимостью и доступностью различных частей кода, что позволяет сосредоточиться на основных задачах, минимизируя возможные ошибки и упрощая процесс тестирования.

Компании, активно использующие условную компиляцию, получают значительные преимущества в скорости разработки, надежности кода и его читабельности. Внедрение условной компиляции в процесс разработки, если оно правильно организовано, может стать одним из ключевых факторов успеха в современных условиях высоких требований к качеству программного обеспечения.

1. **Выполнение тестирования интеграции с помощью HTTP-сервисов.**

В рамках проекта по разработке программного обеспечения тестирование интеграции служит ключевым этапом, обеспечивающим взаимодействие различных компонентов системы. Одним из самых распространенных способов реализации тестирования интеграции является использование HTTP-сервисов, которые позволяют проверять корректность и стабильность работы системы в целом.

1. Определение и цели тестирования интеграции

Интеграционное тестирование нацелено на проверку взаимодействия между модулями и компонентами системы, которые могут быть разработаны разными командами или даже сторонними разработчиками. Основными целями тестирования интеграции являются:

* Проверка совместимости: Убедиться, что все системы и модули могут корректно взаимодействовать друг с другом.
* Выявление ошибок: Обнаружение потенциальных проблем, которые могут возникнуть при взаимодействии различных компонентов, таких как неправильные форматы данных, ошибки протоколов и любые другие проблемы.
* Оценка производительности: Проверить, как система справляется с различными нагрузками при интеграции нескольких сервисов.

1. Подготовка к тестированию

Перед началом тестирования интеграции необходимо выполнить несколько подготовительных шагов:

* Определение интеграционных точек: Необходимо определить, какие именно сервисы и модули будут подвергаться тестированию. Для HTTP-сервисов это, как правило, API, которые обеспечивают взаимодействие между фронтендом и бэкендом, а также взаимодействие с внешними системами.
* Документация API: Необходимо иметь актуальную документацию API, в которой описаны все доступные эндпоинты, методы запросов (GET, POST, PUT, DELETE и др.), форматы ожидаемых данных и возможные коды ответов.
* Создание тестовых сценариев: На основе документации формируются тестовые сценарии, которые будут использоваться в ходе тестирования. Важно учитывать как положительные, так и отрицательные тестовые случаи.

1. Инструменты и технологии

Для проведения тестирования интеграции с помощью HTTP-сервисов могут быть использованы различные инструменты. К числу наиболее популярных относятся:

* Postman: Удобный инструмент для тестирования API, который позволяет отправлять запросы и анализировать ответы. Postman обладает визуальным интерфейсом, что облегчает работу с ним.
* SOAP UI: Этот инструмент предназначен для тестирования как REST, так и SOAP-сервисов и позволяет создавать сложные тесты и автоматизировать их выполнение.
* JMeter: Подходит для нагрузочного тестирования, а также может использоваться для функционального тестирования HTTP-сервисов.
* Selenium: Хотя это инструмент в первую очередь предназначен для тестирования пользовательского интерфейса, его можно интегрировать с API-тестами для комплексного тестирования.

1. Этапы выполнения тестирования

Тестирование интеграции проходит несколько этапов:

* Подготовка окружения: Наличие тестового окружения, которое имитирует рабочую среду системы. Оно должно включать все необходимые сервисы и базу данных, которые необходимы для выполнения тестов.
* Выполнение тестов: На этом этапе проводятся тесты, согласно заранее подготовленным сценариям. Разные типы запросов отправляются к HTTP-сервисам, и фиксируются полученные ответы.
* Анализ результатов: После выполнения тестов следует проанализировать полученные результаты. Необходимо сопоставить фактические результаты с ожидаемыми, чтобы выявить возможные расхождения.
* Отладка: В случае обнаружения ошибок необходимо задать дополнительные вопросы разработчикам, проанализировать логи и найти причину возникновения проблемы. Это может потребовать прямого взаимодействия с командой разработчиков. Примеры тестирования

Рассмотрим несколько примеров тестов, которые могут быть проведены при помощи HTTP-сервисов:

* Тестирование успешного запроса: Проверка корректности работы конечной точки, отправляя допустимые данные и ожидая ответ с кодом 200 (OK) и правильным телом ответа.
* Тестирование ошибок валидации: Отправка запроса с недопустимыми данными (например, пустые поля или неправильный формат) и проверка, что система возвращает соответствующий код ошибки (например, 400 Bad Request).
* Тестирование производительности: Выполнение нагрузки на API, чтобы оценить, как система справляется с высокой частотой запросов. Это можно сделать, например, с помощью JMeter или другого инструмента нагрузочного тестирования.

1. Отчетность и документация

После завершения тестирования очень важно зафиксировать результаты. Протоколирование и создание отчетов о проведенных тестах позволит команде:

* Анализировать проблемы: Иметь возможность возвращаться к тестовым сценариям для дальнейшего анализа.
* Улучшать качество: На основе полученных данных команде разработчиков будет легче выявить слабые места системы и работать над их устранением.
* Обеспечивать отчетность: Для заинтересованных сторон важна прозрачность в процессе тестирования и результаты, которые были получены в ходе проверки интеграции.

Тестирование интеграции с помощью HTTP-сервисов является важной частью жизнецикла разработки программного обеспечения. Это помогает убедиться, что все компоненты системы работают корректно и взаимодействуют друг с другом так, как задумано. Системное и тщательное проведение интеграционного тестирования позволяет минимизировать риски, связанные с внедрением новых функций и улучшений, обеспечивая более высокое качество финального продукта.

**6. Применение основных методов и способов идентификации сбоев и ошибок при интеграции.**

Интеграция различных систем и приложений является ключевым аспектом в современной цифровой трансформации бизнеса. Однако в процессе интеграции неизбежно возникают сбои и ошибки, которые могут негативно сказаться на функционировании всей системы. Поэтому важнейшей задачей становится идентификация этих проблем на раннем этапе, что позволяет минимизировать их влияние. В данном разделе отчета рассматриваются основные методы и способы идентификации сбоев и ошибок при интеграции.

1. Мониторинг и логирование

Одним из основных способов идентификации сбоев является использование систем мониторинга и логирования. Эти системы позволяют отслеживать движение данных между интегрируемыми системами, включая запросы и ответы, которые могут быть полезны для диагностики проблем.

* Мониторинг в реальном времени: Современные решения для мониторинга позволяют отслеживать производительность интеграционных процессов в реальном времени. Они могут выявлять аномалии, такие как задержки в передаче данных или превышение предельных значений времени отклика, что служит сигналом о возможных сбоях.
* Логирование: Запись логов событий, возникающих в процессе интеграции, предоставляет важную информацию для последующего анализа. Логи могут содержать информацию об ошибках, возникающих в ходе выполнения запросов, что позволяет разработчикам определять, где произошел сбой.

1. Автоматизированное тестирование

Автоматизированное тестирование является важным этапом процесса интеграции, позволяющим быстро идентифицировать ошибки и сбои.

* Модульное тестирование: Позволяет тестировать отдельные компоненты системы изолированно, проверяя их корректность и выявляя потенциальные проблемы на ранних стадиях.
* Интеграционное тестирование: Сфокусировано на взаимодействии между компонентами. Оно помогает выявить проблемы, возникающие на стыке различных систем, включая несоответствие форматов данных и протоколов.
* Тестирование производительности: Позволяет оценить, как интегрированная система справляется с нагрузкой. Сбой может произойти в условиях высокой нагрузки, и тестирование производительности помогает выявить такие узкие места еще до фактической эксплуатации системы.

1. Анализ причин и следствий (Root Cause Analysis, RCA)

После выявления сбоя следующим шагом становится анализ его причин. Метод RCA помогает не только обнаружить проблему, но и понять ее первопричины.

* Диаграммы Ишикавы: Позволяют визуализировать возможные причины возникновения проблемы. Это мощный инструмент для групповой работы, который помогает команде найти корень проблемы и предложить пути ее устранения.

1. Статистический анализ

Статистические методы также могут быть полезными при идентификации сбоев и ошибок. Они позволяют выявлять закономерности и аномалии в данных, которые могут свидетельствовать о проблемах.

* Построение графиков и диаграмм: Визуальный анализ данных может помочь выявить аномальные пики или снижения производительности, которые могут являться следствием сбоя.
* Методы машинного обучения: Использование алгоритмов для анализа больших объемов данных позволяет выявить скрытые зависимости и аномалии, которые могут указать на наличие ошибок в интеграции.

1. Ручное тестирование и ревью

Несмотря на значительные преимущества автоматизированных методов, ручное тестирование и ревью кода все еще играют важную роль в процессе идентификации ошибок.

* Код-ревью: Простое просмотр кода другими разработчиками может помочь быстро выявить явные ошибки и недочеты. Это особенно актуально в сложных интеграциях, где не все нюансы могут быть уловлены автоматизированными тестами.
* Пользовательское тестирование: Вовлечение конечных пользователей в тестирование помогает выявить проблемы, которые могут не быть заметными для разработчиков. Пользователи могут указать на несоответствия в функционале, которые могут стать критическими в процессе работы системы.

1. Ретроспективы и анализ завершенных проектов

Проведение ретроспектив и анализ завершенных интеграционных проектов являются важными источниками знаний для повышения качества работы в будущем.

* Уроки, извлеченные из ошибок: Анализ прошедших сбоев и проблем, помогает команде понять, что именно привело к ошибкам, и как можно было бы их избежать.
* Документация: Ведение документации процессов интеграции позволит командам в будущем лучше понять какие аспекты работы были успешными, а какие вызывали трудности. Данные о сбоях и способах их решения могут стать основой для создания лучших практик.

**7. Применение встроенных и основных специализированных инструментов анализа качества программных продуктов.**

В последующие десятилетия программные продукты стали неотъемлемой частью современного общества, проникая во все сферы от бизнеса и медицины до образования и развлечений.

С увеличением сложности и уровня требований к программному обеспечению возрастает важность обеспечения их качества. Для этой цели разработаны различные инструменты анализа качества, которые можно разделить на встроенные инструменты и основные специализированные решения.

Начиная с простых методов тестирования и заканчивая многокомпонентными платформами для глубокого анализа, эти инструменты играют решающую роль в обеспечении стабильно высокого качества программных продуктов.

* Встроенные инструменты анализа качества программного обеспечения (ПО) часто интегрированы непосредственно в среды разработки и позволяют разработчикам оперативно выявлять и устранять проблемы в процессе написания кода. К таким инструментам относятся:
* Статические анализаторы кода: Эти инструменты (например, SonarQube, ESLint, и Checkstyle) анализируют исходный код без его выполнения, выявляя потенциальные ошибки, уязвимости и несоответствия стандартам кодирования. Они помогают разработчикам следовать установленным практикам и улучшать читаемость кода.
* Инструменты для проверки покрытия кода тестами: Такие инструменты, как JaCoCo и Istanbul, измеряют, какая доля кода была протестирована в ходе выполнения тестов. Это позволяет разработчикам понять, какие участки кода требуют более тщательного тестирования, а также помогает улучшить общую надежность продукта.
* Интеграционные и юнит-тесты: Системы непрерывной интеграции (CI) автоматически запускают тесты после каждого изменения кода, что позволяет быстро выявлять ошибки. Инструменты такие как Jenkins, CircleCI и Travis CI помогают интегрировать тестирование в процесс разработки.
* Инструменты для анализа производительности: Некоторые IDE имеют встроенные функции для профилирования кода, позволяющие анализировать время выполнения различных компонентов приложения и выявлять узкие места.

Эти встроенные инструменты облегчают ежедневные задачи разработчиков, повышая общую продуктивность команды и качество продукта на ранних стадиях разработки.

Специализированные инструменты анализа нацелены на более глубокое и всестороннее изучение качества программного продукта, а их применение может охватывать весь жизненный цикл программного обеспечения. К ним можно отнести:

* Инструменты для автоматизированного функционального тестирования: Такие инструменты, как Selenium, TestComplete и QTP, автоматизируют тестирование пользовательского интерфейса, позволяя создавать сценарии, которые имитируют действия пользователя. Это обеспечивает более глубокое тестирование функциональности системы и способствует выявлению недочетов, которые могут быть упущены при ручном тестировании.
* Инструменты для тестирования безопасности: В условиях растущей угрозы кибератак важность тестирования безопасности программных продуктов невозможно переоценить. Инструменты, такие как OWASP ZAP и Burp Suite, помогают выявлять уязвимости в приложениях, что является особенно важным для сервисов, работающих с конфиденциальной информацией.
* Инструменты для анализа производительности: Такие платформы, как Apache JMeter и LoadRunner, помогают проводить нагрузочные тесты и анализировать поведение ПО под воздействием высоких нагрузок. Это важно для понимания того, как приложение будет вести себя в реальных условиях, когда количество пользователей или объем запросов значительно увеличивается.
* Инструменты для мониторинга и логирования: Application Performance Management (APM) инструменты, такие как New Relic и Datadog, обеспечивают постоянный мониторинг работоспособности продуктов в реальном времени, выявляя проблемы до того, как они повлияют на конечных пользователей.
* Платформы для управления качеством: Такие решения как Quality Center или Jira позволяют управлять процессом тестирования, отслеживать ошибки и инциденты, а также контролировать выполнение тестов. Это помогает командам разработчиков и тестировщиков эффективно работать над улучшением качества продукта.

Эффективное применение вышеперечисленных инструментов требует грамотной интеграции их в процесс разработки. Применение Agile и DevOps методологий стало основой для разработки практик, где инструменты автоматизации тестирования и анализа встроены в каждую стадию жизненного цикла ПО. Это позволяет непосредственно взаимодействовать с инструментами анализа в процессе разработки, что значительно сокращает время на выявление и устранение ошибок.

Кроме того, интеграция различных инструментов позволяет создать единую экосистему для управления качеством. Например, результаты статического анализа кода могут быть автоматически преобразованы в задачи для разработчиков в системах управления проектами, таких как Jira.

Это обеспечивает прозрачность процесса и позволяет командам сосредоточиться на наиболее критических аспектах качества.

**8. Использование основных методов для получения кода с заданной функциональностью и степенью качества.**

Создание программного обеспечения требует интеграции множества элементов, включая дизайн, архитектуру, кодирование и тестирование. Одним из ключевых аспектов успешной разработки является использование методов, которые позволяют добиться заданной функциональности при высоком качестве кода. В этом пункте отчёта мы рассмотрим основные методы, используемые для достижения этих целей, включая гибкие методологии разработки, практики программирования и инструменты обеспечения качества.

1. Использование паттернов проектирования позволяет решать распространённые проблемы разработки, обеспечивая при этом поддержку принципов объектно-ориентированного программирования (ООП). К таким паттернам относятся:

* Singleton: гарантирует, что класс имеет только один экземпляр, и предоставляет глобальную точку доступа к нему.
* Observer: определяет зависимость «один ко многим» между объектами, позволяя автоматически уведомлять подчинённые объекты об изменениях состояния.
* Factory Method: предоставляет интерфейс для создания объектов, позволяя подклассам изменять тип создаваемых объектов.

Использование паттернов проектирования способствует улучшению читаемости и поддержки кода, а также помогает избежать дублирования.

1. Методологии тестирования, такие как TDD (Test-Driven Development), помогают разработчикам писать код, который соответствует заданной функциональности. Основные этапы TDD включают:

* Написание тестов, фиксирующих желаемую функциональность до написания самого кода.
* Разработка минимального необходимого кода, который пройдёт тесты.
* Рефакторинг кода, сохраняя его работоспособность в рамках оставшихся тестов.

Подход TDD помогает снизить количество ошибок, увеличивает уверенность в функции кода и способствует созданию более прочного архитектурного решения.

1. Инструменты статического анализа, такие как SonarQube и ESLint, помогают разработчикам идентифицировать потенциальные проблемы в коде до его выполнения. Эти инструменты анализируют код на наличие нарушений стандартов, ошибок, уязвимостей и неэффективных конструкций. Регулярное использование статического анализа позволяет значительно повысить качество кода.
2. Преимущества внедрения практик непрерывной интеграции (CI) и непрерывной доставки (CD) включают:

* Автоматизация тестирования и сборки, что позволяет выявлять ошибки на ранних стадиях цикла разработки.
* Обеспечение более быстрой обратной связи для разработчиков, что позволяет сокращать время между этапами разработки и развертывания.
* Поддержание высокой степени качества за счёт дробления изменений на небольшие, управляемые фрагменты.

1. Документирование кода и архитектуры не менее важно, чем написание самого кода. Чёткая и понятная документация помогает:

* Упростить процесс быстрого понимания функциональности и структуры кода для новых участников команды.
* Обеспечить единообразие в подходах к разработке и документации, что способствует лучшему взаимодействию внутри команды.
* Ускорить процесс поддержки и обновления программного обеспечения в будущем.

**9.** **Анализ проектной и технической документации на предмет соответствия стандартам.**

Анализ проектной и технической документации является неотъемлемой частью процесса разработки и реализации проектов. Он предоставляет ценную информацию о соответствии проектных решений установленным стандартам и нормативам, что критически важно для обеспечения качества, безопасности и эффективности создаваемых объектов. В данном разделе мы рассмотрим ключевые аспекты проведения такого анализа, его методологию, основные критерии оценки и выводы, которые можно сделать на основе результатов.

Основной целью анализа проектной и технической документации является определение уровня соответствия разработанных материалов действующим стандартам и требованиям. В рамках данной задачи можно выделить несколько подзадач:

* Оценка полноты и актуальности документации.
* Проверка на соответствие проектных решений законодательным и нормативным актам.
* Выявление несоответствий и ошибок в расчетах и проектировании.
* Анализ возможных рисков, связанных с отклонениями от стандартов.
* Рекомендации по коррекции недостатков.

Анализ документации проводится с использованием следующей методологии:

* Сбор и подготовка документации: На данном этапе осуществляем сбор всех необходимых документов, включая проектные чертежи, технические задания, спецификации и отчеты о проведенных испытаниях. Важно убедиться, что все материалы актуальны и охватывают все ключевые аспекты проекта.
* Определение стандартов и нормативов: На основе характеристик проекта определяются применимые стандарты и нормативные документы, как национального, так и международного уровня. Это могут быть строительные нормы, стандарты безопасности, экологические требования и др.
* Сравнительный анализ: Происходит сопоставление проектной и технической документации с установленными стандартами. На этом этапе важно зафиксировать все соответствия и несоответствия, а также оценить их степень влияния на реализацию проекта.
* Выявление недостатков: Анализирует факторы, которые могут привести к отклонениям от стандартов. Это включает в себя ошибки проектирования, недостатки в расчетах, а также отсутствие необходимых одобрений и сертификатов.
* Формулирование выводов и рекомендаций: На основе проведенного анализа подготавливается отчет, в который включены выводы о соответствии документации стандартам, а также конкретные рекомендации по устранению выявленных недостатков.

При проведении анализа проектной и технической документации следует использовать следующие ключевые критерии оценки:

* Полнота: Оценка того, насколько исчерпывающей является представляемая документация. Все ключевые аспекты должны быть отражены, чтобы избежать неопределенности в реализации проекта.
* Актуальность: Документация должна быть основана на самых современных версиях стандартов, а также учитывать новейшие технологии и методы проектирования.
* Точность расчетов: Все математические расчеты и технические спецификации должны быть выполнены точно. Ошибки в расчетах могут привести к серьезным последствиям на этапе реализации строительного или производственного проекта.
* Соответствие нормам безопасности: Особое внимание следует уделить стандартам в области безопасности, чтобы избежать возможных угроз как для работников, так и для окружающей среды.
* Экологические требования: Проект должен учитывать все эколого-правовые аспекты, соответствовать законам о защите окружающей среды и стандартам устойчивого развития.

На основе проведенного анализа можно выделить несколько типовых выводов:

* Выявленные соответствия и несоответствия: В процессе анализа могут быть найдены как положительные аспекты, так и недостатки. Например, проект может полностью соответствовать строительным нормам, но иметь недостатки в расчетах электрических систем.
* Риски: Несоответствия могут представлять собой значительные риски, которые могут повлечь за собой финансовые потери или задержки в проекте. Например, игнорирование стандартов безопасности может привести к авариям.
* Рекомендации: В случае выявления несоответствий необходимо предложить четкие шаги для их устранения. Это может включать переработку определенных разделов документации, повторные расчеты, а также дополнительные проверки и испытания.

**Дневник по производственной практике**

|  |  |  |
| --- | --- | --- |
| **Дата** | **Виды выполняемых работ** | **Оценка качества выполняемых работ руководителем практики от предприятия** |
| 03.06.24г. | Ознакомление с организацией работы, структурой и деятельностью структурных подразделений организации по месту прохождения практики.  Прохождение техники безопасности на производственном участке. |  |
| 04.06.24г. | При прохождении производственной практики в дистанционном формате было получено техническое задание: " Проанализировать существующие программные модули и разработать план их интеграции в единую систему ". |  |
| 07.06.24г. | Сбор информации о текущих программных модулях. Проведение интервью с разработчиками и пользователями для выявления функциональных особенностей и проблем. |  |
| 08.06.24г.-10.06.24г. | Разработка технического задания для объединения модулей. Определение требований к интеграции, выбор технологии и средств, которые будут использоваться. |  |
| 11.06.24г.-12.06.24г. | Разработка прототипа интеграции. Создание тестового окружения для проверки совместимости программных модулей. |  |
| 13.06.24г.- | Тестирование интеграционной модели. Исправление выявленных ошибок и настройка параметров для оптимальной работы системы. |  |
| 14.06.24г. | Оформление документов для отчета по практике. |  |
| 15.06.24г. | Подготовка и сдача отчета по практике. |  |

Подпись руководителя практики от предприятия\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись руководителя практики от колледжа\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Профессиональный модуль ПМ.02**

**Основные источники:**

1. Гагарина, Л Г. Технология разработки программного обеспечения [Электронный ресурс] : учебное пособие / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Сидорова-Виснадул ; под ред. Л. Г. Гагариной. — Москва : ИД «ФОРУМ» : ИНФРА-М, 2019. — 400 с. — Режим доступа: <https://new.znanium.com/read?id=336552>
2. Гагарина, Л .Г. Введение в архитектуру программного обеспечения [Электронный ресурс] : учебное пособие / Л. Г. Гагарина, А. Р. Федоров, П. А. Федоров. — Москва : ФОРУМ : ИНФРА-М, 2018. — 320 с. — Режим доступа: <https://new.znanium.com/read?id=315623>
3. Черткова, Е. А. Программная инженерия. Визуальное моделирование программных систем [Электронный ресурс] : учебник для среднего профессионального образования / Е. А. Черткова. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2019. — 147 с. — (Профессиональное образование). — ISBN 978-5-534-09823-5. — Режим доступа: <https://www.biblio-online.ru/bcode/441255>
4. Кундышева, Е. С. Математические методы и модели в экономике [Электронный ресурс] : учебник для бакалавров / Кундышева Е. С.; Под ред. Суслаков Б. А. - Москва :Дашков и К, 2017. - 286 с. ISBN 978-5-394-02488-7 - Режим доступа: <https://new.znanium.com/read?id=300594>
5. Новиков, А. И. Экономико-математические методы и модели [Электронный ресурс] : учебник для бакалавров / А. И. Новиков. - Москва :Дашков и К, 2017. - 532 с. ISBN 978-5-394-02615-7 - Режим доступа: <https://new.znanium.com/read?id=300712>

**Дополнительные источники:**

1.Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности [Электронный ресурс]: учебное пособие / Г. Н. Федорова. — М. :КУРС : ИНФРА-М, 2019. — 336 с. (Среднее Профессиональное Образование). - Режим доступа: <https://new.znanium.com/read?id=330691>

2.Хорев, П. Б. Объектно-ориентированное программирование с примерами на С# [Электронный ресурс] : учебное пособие / П. Б. Хорев. — Москва : ФОРУМ : ИНФРА-М, 2019. — 200 с. — Режим доступа: <https://new.znanium.com/read?id=339308>

3.Гетманчук, А. В. Экономико-математические методы и модели [Электронный ресурс] : учебное пособие / Гетманчук А. В., Ермилов М. М. - Москва :Дашков и К, 2017. - 186 с.: ISBN 978-5-394-01575-5 - Режим доступа: <https://new.znanium.com/read?id=108077>

**Интернет-ресурсы:**

1.Интуит Национальный открытый университет курс Язык UML 2 в анализе и проектировании программных систем и бизнес-процессов [Электронный ресурс]. – Режим доступа: <http://www.intuit.ru/studies/courses/480/336/info>, свободный.– Загл. с экрана. Яз. рус.3.3 Учебно-методическое обеспечение самостоятельной работы обучающихся

2.Интуит Национальный открытый университет курс Проектирование информационных систем в Microsoft SQL Server 2008 и Visual Studio 2008 [Электронный ресурс]. – Режим доступа: <http://www.intuit.ru/studies/courses/502/358/info>, свободный.– Загл. с экрана. Яз. рус.