**АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ (ФИЛИАЛ) СПбГУТ**

**(АКТ (ф) СПбГУТ)**

**Отчеты по лабораторным и практическим работам**

**по МДК.02.02**

Студент: Кузьмин Андрей

Преподаватель: Маломан Юлия Сергеевна

Архангельск 2024

**Лабораторная работа №1**

**Исследование способов рефакторинга программного кода**

1. **Цель работы**
   1. Изучить техники рефакторинга программного кода.
   2. Изучить процесс документирования программного кода.
2. **Контрольные вопросы**
   1. Рефакторинг — это процесс изменения структуры кода без изменения его внешнего поведения, направленный на улучшение читаемости, упрощение и оптимизацию кода.
   2. Существуют различные группы техник рефакторинга, включая улучшение имен переменных и методов, упрощение сложных выражений, удаление дублирующегося кода, разделение больших методов на более мелкие и улучшение структуры классов.
   3. XML-документация — это способ документирования кода с использованием XML-формата, который позволяет описывать классы, методы и их параметры, а также возвращаемые значения. Это помогает разработчикам лучше понимать и использовать код.
   4. В разделе summary пишется краткое описание метода или класса, которое объясняет его основное назначение и функциональность.
   5. В разделе param указывается описание параметров метода, включая их типы и назначение, что помогает понять, какие данные необходимо передать в метод.
   6. В разделе returns описывается значение, которое возвращает метод, включая его тип и смысл, что позволяет пользователям метода знать, что они могут ожидать в результате его выполнения.
3. **Вывод**

Изучены техники рефакторинга программного кода.

Изучен процесс документирования программного кода.

**Лабораторная работа №2**

**Исследование способов оптимизации программного кода**

1. **Цель работы**
   1. Изучить методы оптимизации программного кода.
2. **Контрольные вопросы**
   1. Оптимизация программного кода — это процесс улучшения его производительности, уменьшения потребления ресурсов и повышения эффективности выполнения. Это может включать в себя улучшение скорости выполнения, снижение использования памяти и других ресурсов.
   2. Цель оптимизации программного кода заключается в повышении его производительности и эффективности, что позволяет приложениям работать быстрее и более эффективно использовать системные ресурсы. Это может привести к улучшению пользовательского опыта и снижению затрат на инфраструктуру.
   3. Среди методов оптимизации программного кода можно выделить алгоритмическую оптимизацию, использование кэширования, минимизацию количества операций ввода-вывода, оптимизацию циклов и использование более эффективных структур данных. Также важным аспектом является профилирование кода для выявления узких мест.
   4. Асинхронный вызов относится к оптимизации, поскольку он позволяет выполнять операции параллельно, не блокируя основной поток выполнения программы. Это особенно полезно для операций, которые могут занять значительное время, таких как сетевые запросы или операции с диском, что позволяет улучшить отзывчивость приложения и более эффективно использовать ресурсы.
3. **Вывод**

Изучены методы оптимизации программного кода.

**Лабораторная работа №3**

**Отладка проекта**

1. **Цель работы**
   1. Изучить процесс отладки приложений,
   2. Изучить процесс применения отладочных классов.
2. **Контрольные вопросы**
   1. Отладка — это процесс выявления и устранения ошибок или дефектов в программном коде. Она включает в себя анализ поведения программы, чтобы понять, почему она не работает так, как ожидалось.
   2. Этапы отладки обычно включают в себя воспроизведение ошибки, анализ кода, использование отладочных инструментов для отслеживания выполнения программы, внесение исправлений и повторное тестирование, чтобы убедиться, что ошибка устранена.
   3. Точка останова — это специальная метка, устанавливаемая в коде, которая при выполнении программы останавливает её выполнение. Это позволяет разработчику исследовать текущее состояние программы, включая значения переменных и поток выполнения.
   4. В проекте могут использоваться различные отладочные классы, такие как классы для логирования, классы для обработки исключений и классы, предоставляющие интерфейсы для отладочных инструментов. Эти классы помогают в отслеживании и анализе ошибок.
   5. Классы трассировки могут потребоваться для записи информации о выполнении программы, что позволяет разработчикам отслеживать её поведение и выявлять проблемы. Трассировка может включать в себя запись событий, ошибок и других значимых действий, что помогает в анализе и отладке кода.
3. **Вывод**

Изучен процесс отладки приложений,

Изучен процесс применения отладочных классов.

**Лабораторная работа №4**

**Отладка веб-приложений**

1. **Цель работы**
   1. Изучить процесс отладки веб-приложений средствами разработчика в браузере.
   2. Изучить процесс средств тестирования API.
2. **Контрольные вопросы**
   1. Postman — это инструмент для тестирования API, который позволяет разработчикам отправлять запросы к серверу и получать ответы. Он предоставляет удобный интерфейс для работы с различными типами запросов (GET, POST, PUT, DELETE и т. д.), а также возможность сохранять и организовывать запросы, тестировать их и документировать API.
   2. Swagger — это набор инструментов и спецификаций для документирования и тестирования RESTful API. Он позволяет разработчикам описывать API в формате OpenAPI, что делает его более понятным и доступным для других разработчиков. Swagger также предоставляет интерфейс для интерактивного тестирования API.
   3. Чтобы получить доступ к элементу с помощью document.querySelector(), необходимо передать в качестве аргумента CSS-селектор, который соответствует искомому элементу. Например, document.querySelector('.class-name') вернет первый элемент с указанным классом.
   4. Для получения информации о производительности и памяти с помощью средств разработчика в браузере можно использовать вкладки "Performance" и "Memory". Вкладка "Performance" позволяет записывать и анализировать производительность приложения, а вкладка "Memory" предоставляет инструменты для анализа использования памяти, включая сборку мусора и утечки памяти.
   5. Логирование средствами JavaScript выполняется с помощью функции console.log(), которая выводит сообщения в консоль разработчика. Также доступны другие методы, такие как console.error(), console.warn(), и console.info(), которые позволяют выводить сообщения различного уровня важности.
   6. Для тестирования интерфейса веб-приложения с использованием средств разработчика в браузере можно использовать инструменты, такие как "Elements" для изменения HTML и CSS в реальном времени, "Console" для выполнения JavaScript-кода, а также "Network" для анализа сетевых запросов и ответов. Эти инструменты позволяют разработчикам проверять и отлаживать интерфейс, а также тестировать взаимодействие с пользователем.
3. **Вывод**

Изучен процесс отладки веб-приложений средствами разработчика в браузере.

Изучен процесс средств тестирования API.

**Лабораторная работа №5**

**Тестирование интеграции**

1. **Цель работы**
   1. Изучить процесс проведения интеграционного тестирования.
2. **Контрольные вопросы**
   1. Интеграционное тестирование — это этап тестирования программного обеспечения, на котором проверяется взаимодействие между различными модулями или компонентами системы. Цель интеграционного тестирования заключается в выявлении ошибок, возникающих при взаимодействии этих компонентов, и в обеспечении их корректной работы в совокупности.
   2. Существуют несколько способов интеграционного тестирования, включая тестирование по большому взрыву, когда все модули интегрируются одновременно, и инкрементное тестирование, которое может быть выполнено поэтапно. Инкрементное тестирование делится на два подхода: снизу вверх, когда тестируются низкоуровневые модули, и сверху вниз, когда тестируются высокоуровневые модули.
   3. Интеграционное тестирование позволяет проверить, как модули взаимодействуют друг с другом, выявить проблемы с интерфейсами, совместимостью данных и корректностью передачи информации между компонентами. Оно также помогает убедиться, что система в целом работает так, как задумано.
   4. Для автоматизации проведения интеграционного тестирования можно использовать различные инструменты и программное обеспечение, такие как JUnit для тестирования Java-приложений, TestNG для тестирования Java, Postman для тестирования API, Selenium для автоматизации тестирования веб-приложений, Cypress для тестирования современных веб-приложений и Jenkins для автоматизации процессов CI/CD. Эти инструменты помогают упростить и ускорить процесс тестирования, обеспечивая более высокое качество программного обеспечения.
3. **Вывод**

Изучен процесс проведения интеграционного тестирования.

**Лабораторная работа №6**

**Повышение удобства использования приложения**

1. **Цель работы**
   1. Изучить процесс проектирования и разработки интерфейса пользователя.
   2. Научиться оптимизировать приложение для удобства работы пользователя.
2. **Контрольные вопросы**
   1. Основные подходы к реализации адаптивного дизайна включают использование медиа-запросов для изменения стилей в зависимости от размера экрана, гибкие сетки и изображения, а также подход "mobile-first", при котором дизайн сначала создается для мобильных устройств, а затем адаптируется для более крупных экранов. Современные технологии, такие как CSS Grid и Flexbox, позволяют создавать гибкие и отзывчивые макеты, а фреймворки, такие как Bootstrap и Foundation, предоставляют готовые компоненты для быстрого создания адаптивных интерфейсов.
   2. При выборе цветовой схемы пользовательского интерфейса следует учитывать такие факторы, как целевая аудитория, контекст использования, психологическое восприятие цветов, доступность для людей с нарушениями зрения (например, дальтонизм), а также соответствие бренду и общему стилю приложения. Важно также обеспечить достаточный контраст между текстом и фоном для удобства чтения.
   3. Принципы и методы, используемые при проектировании расположения элементов управления, включают принцип близости, который предполагает размещение связанных элементов рядом друг с другом, принцип выравнивания для создания визуальной структуры, а также использование иерархии для выделения наиболее важных элементов. Также важно учитывать удобство доступа к элементам управления, их размер и расстояние между ними, чтобы избежать ошибок при взаимодействии.
   4. Алгоритм расстояния Левенштейна используется в разработке пользовательских интерфейсов для реализации функций, связанных с обработкой текста, таких как автозаполнение, исправление опечаток и поиск с учетом ошибок. Он позволяет оценить, насколько близки два текста, что помогает улучшить взаимодействие пользователя с интерфейсом, предлагая более точные результаты.
   5. Интеграция QR-кодов может улучшить пользовательский опыт в приложении, предоставляя быстрый доступ к информации или функциям, таким как ссылки на веб-страницы, скидки, контактные данные или инструкции. Пользователи могут сканировать QR-коды с помощью своих мобильных устройств, что упрощает взаимодействие и делает его более удобным, особенно в ситуациях, когда ввод текста может быть затруднительным.
3. **Вывод**

Изучен процесс проектирования и разработки интерфейса пользователя.

Научились оптимизировать приложение для удобства работы пользователя.

**Лабораторная работа №7**

**Защита данных**

1. **Цель работы**
   1. Изучить процесс шифрования данных пользователя.
   2. Научиться применять криптографические методы для защиты данных пользователя.
2. **Контрольные вопросы**
   1. Шифрование — это процесс преобразования данных в нечитабельный формат с целью защиты информации от несанкционированного доступа. Шифрование позволяет обеспечить конфиденциальность данных, так как только обладатель ключа может расшифровать и получить доступ к исходной информации.
   2. Хеширование — это процесс преобразования входных данных произвольной длины в фиксированное значение (хеш), которое представляет собой уникальный идентификатор для этих данных. Хеширование используется для проверки целостности данных и быстрого поиска.
   3. Хеширование применяется в различных случаях, таких как хранение паролей, проверка целостности файлов, создание уникальных идентификаторов для данных и в системах управления версиями.
   4. При хешировании добавляется соль — это случайная строка, которая добавляется к паролю перед хешированием. Соль помогает защитить от атак с использованием радужных таблиц и делает хеши уникальными даже для одинаковых паролей, что повышает безопасность хранения паролей.
   5. Рекомендуемыми алгоритмами хеширования являются SHA-256 и SHA-3, так как они обеспечивают высокий уровень безопасности и устойчивы к коллизиям. Также можно использовать bcrypt, scrypt и Argon2 для хеширования паролей, так как они включают механизмы для замедления процесса хеширования.
   6. Существуют различные алгоритмы симметричного шифрования, такие как AES (Advanced Encryption Standard), DES (Data Encryption Standard) и 3DES (Triple DES). Эти алгоритмы используют один и тот же ключ для шифрования и расшифровки данных.
   7. Среди алгоритмов асимметричного шифрования можно выделить RSA (Rivest-Shamir-Adleman), DSA (Digital Signature Algorithm) и ECC (Elliptic Curve Cryptography). Эти алгоритмы используют пару ключей: открытый ключ для шифрования и закрытый ключ для расшифровки.
   8. Цифровая подпись создается путем хеширования сообщения с использованием криптографического хеш-алгоритма, а затем шифрования полученного хеша с помощью закрытого ключа отправителя. Это позволяет получателю проверить подлинность сообщения, расшифровав подпись с помощью открытого ключа отправителя и сравнив полученный хеш с хешем оригинального сообщения.
3. **Вывод**

Изучен процесс шифрования данных пользователя.

Научились применять криптографические методы для защиты данных пользователя.

**Лабораторная работа №8**

**Проектирование и разработка интерфейса пользователя**

1. **Цель работы**
   1. Изучить процесс проектирования и разработки интерфейса пользователя.
2. **Контрольные вопросы**
   1. При проектировании интерфейсов пользователя (UI) для обеспечения их удобства и интуитивности следует учитывать такие принципы, как простота и ясность, последовательность в использовании элементов, доступность информации, визуальная иерархия, а также отзывчивость интерфейса. Важно также учитывать потребности и ожидания пользователей, проводить тестирование и собирать обратную связь для улучшения дизайна.
   2. Пользовательский интерфейс (UI) относится к визуальным элементам и элементам управления, с которыми взаимодействует пользователь, таким как кнопки, меню и формы. Пользовательский опыт (UX) охватывает более широкий спектр, включая общее восприятие и удовлетворение пользователя от взаимодействия с продуктом. UI является частью UX, и хороший интерфейс способствует положительному пользовательскому опыту.
   3. Для разработки пользовательских интерфейсов используются современные технологии и инструменты, такие как HTML, CSS и JavaScript для создания веб-страниц, а также фреймворки и библиотеки, такие как React, Angular и Vue.js. Для дизайна интерфейсов применяются инструменты, такие как Figma, Adobe XD и Sketch, которые позволяют создавать прототипы и макеты.
   4. Для улучшения доступности интерфейса для пользователей с ограниченными возможностями можно использовать такие технологии и решения, как альтернативный текст для изображений, поддержка экранных читалок, контрастные цветовые схемы, возможность навигации с клавиатуры и использование ARIA (Accessible Rich Internet Applications) для улучшения семантики интерфейса. Также важно проводить тестирование доступности с участием пользователей с ограниченными возможностями.
   5. Основные преимущества использования адаптивного дизайна (responsive design) в веб-интерфейсах включают улучшенное пользовательское восприятие на различных устройствах, снижение затрат на разработку и поддержку, а также улучшение SEO, так как один и тот же URL используется для всех устройств. Недостатками могут быть сложности в реализации сложных макетов, возможные проблемы с производительностью на мобильных устройствах и необходимость тестирования на множестве устройств и экранов для обеспечения корректного отображения.
3. **Вывод**

Изучен процесс проектирования и разработки интерфейса пользователя.

**Лабораторная работа №9**

**Разработка и интеграция модулей проекта (командная работа)**

1. **Цель работы**
   1. Получить навыки совместной разработки с использованием системы контроля версий
   2. Получить навыки интеграции компонентов ПО при командной разработке
2. **Вывод**

Получены навыки совместной разработки с использованием системы контроля версий

Получены навыки интеграции компонентов ПО при командной разработке

**Лабораторная работа №10**

**Развертывание и установка ПО**

1. **Цель работы**
   1. Изучить процесс развертывания и установки разработанного ПО.
2. **Контрольные вопросы**
   1. Portable-приложение — это программа, которая может быть запущена без необходимости установки на компьютер. Такие приложения обычно не требуют записи в реестр и могут быть запущены с внешних носителей, таких как USB-накопители. Это делает их удобными для использования на разных устройствах без изменения их конфигурации.
   2. Для хостинга своих приложений можно использовать различные веб-серверы, такие как Apache, Nginx и Microsoft IIS. Эти серверы поддерживают различные языки программирования, включая PHP, Python, Ruby, Node.js и Java. В качестве систем управления базами данных (СУБД) могут использоваться MySQL, PostgreSQL, MongoDB и SQLite.
   3. Docker используется для контейнеризации приложений, что позволяет упаковать приложение и все его зависимости в единый контейнер. Это обеспечивает консистентность среды выполнения, упрощает развертывание и масштабирование приложений, а также позволяет легко управлять версиями и зависимостями.
   4. В Dockerfile указываются инструкции для создания образа контейнера. Пример Dockerfile может выглядеть следующим образом:

|  |
| --- |
| # Указываем базовый образ  FROM node:14  # Устанавливаем рабочую директорию  WORKDIR /app  # Копируем package.json и устанавливаем зависимости  COPY package.json ./  RUN npm install  # Копируем остальные файлы приложения  COPY . .  # Указываем команду для запуска приложения  CMD ["npm", "start"] |

* 1. Существуют различные приложения для создания инсталляторов, такие как Inno Setup, NSIS (Nullsoft Scriptable Install System), WiX Toolset и Advanced Installer. Эти инструменты позволяют создавать установочные пакеты для Windows и других операционных систем, упрощая процесс установки программного обеспечения для пользователей.

1. **Вывод**

Изучен процесс развертывания и установки разработанного ПО.

**Лабораторная работа №11**

**Разработка справочной документации**

1. **Цель работы**
   1. Изучить процесс оформления эксплуатационной документации.
2. **Контрольные вопросы**
   1. Руководство пользователя — это документ, который предоставляет пользователям информацию о том, как использовать программное обеспечение или устройство. Оно содержит инструкции, советы и рекомендации, которые помогают пользователям понять функциональность продукта и эффективно его использовать.
   2. Способы добавления справки в приложение могут включать встроенные справочные системы, контекстные подсказки, интерактивные руководства, видеоинструкции, а также доступ к онлайн-справке или FAQ. Также можно использовать системы поддержки, такие как чаты или форумы, для получения помощи от сообщества или службы поддержки.
   3. Разделы, которые обычно входят в руководство пользователя, могут включать введение, описание системы и ее функциональности, инструкции по установке и настройке, руководство по использованию, разделы по устранению неполадок, а также информацию о поддержке и обновлениях.
   4. В России существуют ГОСТы и РД, которые описывают требования к разделам руководства пользователя и аналогичной эксплуатационной документации. Например, ГОСТ 19.201-78 "Единая система программной документации. Руководство пользователя" и ГОСТ 34.601-90 "Системы программной документации. Общие требования". Эти документы устанавливают стандарты для оформления и содержания документации, включая требования к структуре и содержанию руководств пользователя.
3. **Вывод**

Изучен процесс оформления эксплуатационной документации.

**Лабораторная работа №12**

**Презентация ПО**

1. **Цель работы**
   1. Изучить процесс оформления презентации.
   2. Научиться презентовать разработанное ПО
2. **Контрольные вопросы**
   1. Питч-презентация — это краткая и убедительная презентация, предназначенная для представления идеи, продукта или проекта потенциальным инвесторам, партнерам или клиентам. Она проводится с целью привлечения внимания и получения финансирования, поддержки или интереса к проекту.
   2. Существуют различные онлайн-средства, которые позволяют создать презентацию, такие как Google Slides, Microsoft PowerPoint Online, Prezi, Canva и SlideShare. Эти инструменты предлагают шаблоны и функции для создания визуально привлекательных слайдов.
   3. Требования к наполнению слайдов презентации включают ясность и лаконичность текста, использование визуальных элементов (графиков, изображений, диаграмм) для иллюстрации ключевых моментов, соблюдение единого стиля и цветовой схемы, а также ограничение количества информации на одном слайде, чтобы не перегружать аудиторию.
   4. В презентации ПО для технических специалистов указываются детали архитектуры системы, используемые технологии, интеграционные возможности, требования к системе, а также информация о безопасности и производительности. Это помогает техническим специалистам понять, как продукт будет работать в их среде.
   5. В презентации ПО для потенциальных пользователей акцент делается на преимуществах и функциональности продукта, его удобстве, примерах использования, а также на том, как он решает конкретные проблемы пользователей. Важно также включить информацию о поддержке и обучении.
   6. MVP (Minimum Viable Product) — это минимально жизнеспособный продукт, который содержит только основные функции, необходимые для удовлетворения потребностей ранних пользователей. MVP позволяет быстро протестировать идею на рынке, получить обратную связь и внести улучшения, прежде чем инвестировать в разработку более полной версии продукта.
3. **Вывод**

Изучен процесс оформления презентации.

Научились презентовать разработанное ПО