бюджетное профессиональное образовательное учреждение Вологодской области

«Череповецкий лесомеханический техникум им. В.П. Чкалова»

Специальность **09.02.07** «Информационные системы и программирование»

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

_____

Выполнил студент _ курса группы ИС-____

_____

подпись _____

место практики

_____

наименование юридического лица, ФИО ИП

Период прохождения:

с «___» _____ 202_ г.

по «___» _____ 202_ г.

Руководитель практики от

предприятия

должность_____

_____

подпись_____

МП

Руководитель практики от

техникума: _____

_____

Оценка:_____

«___» _____202_ год

Введение

1. О компании

Малленом Системс – ведущая российская компания в области разработки и внедрения систем компьютерного зрения, промышленной видеоаналитики и интеллектуальной обработки данных. В основе создаваемых компанией

решений – технологии машинного зрения и искусственного интеллекта (машинное обучение, нейронные сети глубокого обучения).

Организационная структура предприятия:

- Центр по развитию интеллектуальных систем, отдел разработки ПО. Проектирование, разработка, оптимизация ПО для клиентов компании.
- Производственно-технический отдел. Отдел с инженерами, которые проводят пусконаладочные работы на предприятиях, проектируют местонахождение оборудования на предприятии и устанавливают его.
- АУП (Административно-управленческий персонал). Руководство компании, которое формирует стратегии развития, управляет отделами, планирует деятельность предприятия, обеспечивает внешние коммуникации компании на выставках, в СМИ.
- Группа Маркетинга. Формирование маркетинговой стратегии компании, внутренний и внешний PR-компании, продвижение бренда и продуктов на рынке.
- Коммерческий отдел. Продажа продуктов компании заказчикам, поиск новых клиентов, участие в PR-продвижении компании.
- Отдел технической поддержки и контроля качества. Техническая поддержка пользователей и тестировка ПО на выявление ошибок и проблем.
- Отдел акселерационных и образовательных программ. Разработка и проведение обучающих курсов по машинному зрению и языку программирования, PR компании на рынке образовательных учреждений

- Отдел кадров. Управление персоналом компании, поиск, подбор, адаптация сотрудников, ведение кадрового документооборота, разработка стратегия развития персоналом предприятия
- Юридический отдел. Обработка всех документов в компании в соответствии с законодательством, взаимодействие с заказчиками и менеджерами по договорным обязательствам.
- Бухгалтерия. Ведение экономической деятельности предприятия, бухгалтерского учета, формирование бюджетов компании.
- ОХР (общественно-хозяйственные рабочие). Поддержание чистоты, порядка на рабочих местах, ремонт, уборка служебных помещений.

В компании «Малленом Системс» действует график работы 5/2, с 09:00 до 18:00. В штате компании есть отдельный специалист по охране труда, который проводит вводные инструктажи при приеме на работу и практике, а также занимается выдачей пропусков для пусконаладочных работ инженеров. В компании 20.09.2018 г. Была проведена специальная оценка условий труда, согласно которой рабочие места, на территории которых установлены вредные производственные факторы, отсутствуют.

Так же производим обучение сотрудников по промышленной безопасности, по электробезопасности, охране труда и обучению в области применения средств защиты и оказания первой медицинской помощи в специализированном учебном центре (Негосударственное частное образовательное учреждение дополнительного профессионального образования «Учебный центр «Экоконсалт») с последующей аттестацией в Северо-Западном управлении Ростехнадзора в Вологодской области (только для промышленной безопасности и электробезопасности).

Должностные инструкции ИТ-специалистов

1.1. Настоящая должностная инструкция определяет должностные обязанности, права и ответственность Техника Общества с ограниченной ответственностью «Малленом Системс» (далее – Техник, Общество).

1.2. Техник относится к категории специалистов.

1.3. Техник принимается на работу и увольняется приказом генерального директора или уполномоченным им лицом.

1.4. На должность Техника назначается лицо, без предъявления требований к образованию и опыту работы.

1.5. Техник подчиняется непосредственно руководителю структурного подразделения, ведущему программисту и/или руководителю направления/проекта, в котором работает в настоящее время.

1.6. Техник должен знать:

– методы автоматической и автоматизированной проверки работоспособности программного обеспечения;

– основные виды диагностических данных и способы их представления;

– языки, утилиты и среды программирования, и средства пакетного выполнения процедур;

– типовые метрики программного обеспечения;

– основные методы измерения и оценки характеристик программного обеспечения;

– методы создания и документирования контрольных примеров и тестовых наборов данных;

– правила, алгоритмы и технологии создания тестовых наборов данных;

– требования к структуре и форматам хранения тестовых наборов данных;

– методы и средства проверки работоспособности программного обеспечения;
– среду проверки работоспособности и отладки программного обеспечения;
– внутренние нормативные документы, регламентирующие порядок документирования результатов проверки работоспособности программного обеспечения;
– методы и средства рефакторинга и оптимизации программного кода;
– языки программирования и среды разработки;
– внутренние нормативные документы, регламентирующие требования к программному коду, порядок отражения изменений в системе контроля версий;
– внутренние нормативные документы, регламентирующие порядок отражения результатов рефакторинга и оптимизации в коллективной базе знаний;
– методы и приемы отладки программного кода;
– типовые ошибки, возникающие при разработке программного обеспечения, и методы их диагностики и исправления;
1.7. Техник должен знать и уметь:
– писать программный код процедур проверки работоспособности программного обеспечения на выбранном языке программирования под руководством наставника;
– использовать выбранную среду программирования для разработки процедур проверки работоспособности программного обеспечения на выбранном языке программирования;
– применять методы и средства проверки работоспособности программного обеспечения;
– анализировать значения полученных характеристик программного

обеспечения;
– документировать результаты проверки работоспособности программного обеспечения;
– применять методы, средства для рефакторинга и оптимизации;
– публиковать результаты рефакторинга и оптимизации в коллективной базе знаний в виде лучших практик;
– использовать систему контроля версий для регистрации произведенных изменений;
– применять методы и приемы отладки дефектного программного кода;

2. Осуществление интеграции программных модулей

2.1 Разработка требований к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент

На этом этапе инженеры (архитекторы, ведущие разработчики) не создают сами модули, а определяют "правила игры" для них. Цель — убедиться, что все модули будут иметь совместимые интерфейсы и корректно взаимодействовать.

Конкретные действия:

Анализ документации: изучаются техническое задание (ТЗ), архитектурные диаграммы, спецификации API, схемы баз данных.

Определение контрактов интерфейсов:

- Для функций/методов: Какие параметры они принимают (тип, имя), что возвращают, какие исключения выбрасывают.
- Для API (REST, GraphQL и т.д.): Какие endpoints доступны, какие

HTTP-методы используются, форматы запросов и ответов (JSON, XML).

- Для данных: Форматы сообщений, структуры объектов, схемы таблиц в БД.

Документирование требований: Создание формальных документов, таких как:

- Спецификация API (например, в OpenAPI/Swagger).
- Протоколы взаимодействия (например, для микросервисов).
- Диаграммы последовательности, показывающие, как модули обмениваются данными.

2.2 Выполнение интеграции модулей в программное обеспечение

Это практический процесс объединения отдельных модулей в единую работающую систему или ее крупную подсистему.

Подходы к интеграции:

- Нисходящая интеграция: сначала собираются и тестируются модули верхнего уровня (менеджеры, контроллеры), а затем к ним постепенно подключаются модули нижнего уровня (сервисы, репозитории). Для отсутствующих модулей используются заглушки (stubs).
- Восходящая интеграция: сначала интегрируются и тестируются модули нижнего уровня, которые затем объединяются в более крупные подсистемы. Для модулей верхнего уровня создаются драйверы (drivers).

- Многоуровневая (сэндвич) интеграция: Комбинация нисходящего и восходящего подходов.
- Построение (Big Bang): Все модули интегрируются одновременно. Самый рискованный и сложный для отладки подход.

Конкретные действия:

- Настройка систем сборки (например, Maven, Gradle, Webpack).
- Написание сборочных скриптов (например, CI/CD пайплайны в Jenkins, GitLab CI).

2.3 Выполнение отладки программного модуля с использованием специализированных программных средств

После интеграции система почти наверняка будет работать с ошибками. Отладка — это процесс поиска, анализа и устранения причин этих ошибок (дефектов, багов).

Специализированные программные средства:

- Отладчики (Debuggers): Инструменты, позволяющие пошагово выполнять код, анализировать call stack, инспектировать и изменять значения переменных в режиме реального времени.
- Профилировщики (Profilers): Инструменты для поиска "узких мест" производительности — например, функций, потребляющих много процессорного времени или памяти.
- Средства мониторинга логов (Logging & Monitoring): Централизованный сбор, анализ и визуализация логов и метрик приложения.

2.4 Осуществление разработки тестовых наборов и тестовых сценариев для программного обеспечения

Это деятельность по созданию формализованных процедур для проверки корректности работы как интегрированной системы в целом, так и ее отдельных частей.

Типы разрабатываемых тестов:

- Интеграционные тесты: проверяют взаимодействие между несколькими модулями. Например, проверяют, что сервис корректно запрашивает данные из репозитория и обрабатывает их.
- Системные тесты: проверяют работу всей системы в среде, максимально приближенной к реальной, на соответствие исходным требованиям.
- Приемочные тесты (UAT): проверяют, удовлетворяет ли система потребностям пользователя и соответствует ли критериям приемки.

Конкретные действия:

- Тестовые наборы (Test Suites): Группы тестовых случаев, объединенных по какому-либо признаку (например, "тесты модуля аутентификации").
- Тестовые сценарии (Test Cases): Детальные пошаговые инструкции: "При заданных входных данных X и Y система должна вернуть результат Z".
- Использование фреймворков: Написание кода тестов с использованием таких инструментов, как JUnit (Java), pytest (Python), Selenium (веб-UI),

Postman (API).

2.5 Инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования

Это процесс статического анализа исходного кода (без его запуска) с целью проверки его качества, читаемости, поддерживаемости и соответствия установленным в команде правилам.

Методы и инструменты:Ревью кода (Code Review): Коллегиальная проверка кода другими разработчиками (через Pull/Merge Request в Git).
Основная часть

Модуль взаимодействия с пользователем предназначен для организации командного интерфейса системы обработки изображений. Он обеспечивает прием команд, выполнение операций и сохранение текущего состояния системы.

Основные функции:

Обработка команд, вводимых через консоль

Управление настройками системы

Ведение истории выполненных команд

Вывод информационных сообщений и сообщений об ошибках

1. Описание классов

UserSettings — класс для хранения пользовательских настроек:

Поля:

Language— язык интерфейса (по умолчанию: "ru")

Theme— тема оформления (по умолчанию: "dark")

DefaultWidth— ширина изображения по умолчанию (по умолчанию: 800)

DefaultHeight— высота изображения по умолчанию (по умолчанию: 600)

SettingsFilePath— путь к файлу настроек

CommandHistory— управление историей команд:

Commands— список выполненных команд

LastModified— дата последнего изменения

HistoryFilePath— путь к файлу истории

MaxHistorySize— максимальный размер истории (по умолчанию: 100)

SettingsManager— управление сохранением и загрузкой данных

Методы:

LoadSettings()— загрузка настроек из файла

SaveSettings()— сохранение настроек в файл

LoadCommandHistory()— загрузка истории команд

SaveCommandHistory()— сохранение истории команд

UserInterfaceModule — основной класс для управления интерфейсом:

Методы:

Run()— запуск основного цикла программы

ProcessCommand()— обработка введенной команды

ShowHelp()— вывод справочной информации

Обработчики команд, такие какHandleLoadCommand

HandleInfoCommandи др.

2. Функциональные возможности

Настройки системы:

Параметры:

language— язык интерфейса

theme— тема оформления

3. Реализация

Технологии и подходы:

Хранение данных:Использование текстовых файлов для простоты и надежности.

Формат хранения настроек — ключ=значение.

История команд сохраняется построчно.

Обработка ошибок:

Использование блоков

try-catch

для перехвата исключений.

Вывод информативных сообщений об ошибках.

Обеспечение устойчивости к некорректному вводу пользователя.

Особенности реализации:

Основной цикл программы:

Вывод приглашения для ввода команды.

Чтение команды от пользователя.

Добавление команды в историю.

Обработка команды.

Сохранение текущего состояния системы.

4. Преимущества и ограничения

Преимущества:Простота использования и интуитивно понятный интерфейс.

Автоматическое сохранение настроек и истории команд.

Расширяемая архитектура для добавления новых функций.

Кроссплатформенность — работает на любой ОС с .NET.

Минимальные зависимости.

Ограничения:Базовые возможности обработки изображений — реализованы в виде имитации.

Ограниченный набор команд.

5. Возможности расширения

Планируемые улучшения:Внедрение графического интерфейса.

Реальная обработка изображений.

Поддержка дополнительных форматов файлов.

Внедрение системы плагинов для новых команд.

Улучшенная система настроек с валидацией.

Интеграционные возможности:

REST API для удаленного управления.

Поддержка скриптов и пакетной обработки.

Интеграция с облачными хранилищами.

Заключение

В ходе выполнения работы был успешно разработан и реализован модуль взаимодействия с пользователем для системы управления изображениями. Модуль представляет собой законченное решение, которое полностью соответствует поставленным техническим требованиям и обеспечивает удобный консольный интерфейс для работы с системой.

Достигнутые результаты

Разработанный модуль демонстрирует высокую эффективность в решении ключевых задач:Организован интуитивно понятный консольный интерфейс с поддержкой основных команд управленияРеализована надежная система

сохранения и восстановления настроек пользователяОбеспечено ведение полной истории выполненных команд с автоматическим сохранениемСоздана расширяемая архитектура, позволяющая легко добавлять новые функциональные возможности