СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc182861399)

[1 ОБЩАЯ ХАРАКТЕРИСТИКА ПРЕДПРИЯТИЯ 4](#_Toc182861400)

[2 ОСУЩЕСТВЛЕНИЕ ИНТЕГРАЦИИ ПРОГРАММНЫХ МОДУЛЕЙ 7](#_Toc182861401)

[2.1 Разработка требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент 7](#_Toc182861402)

[2.2 Выполнение интеграции модулей в программное обеспечение 7](#_Toc182861403)

[2.3 Выполнение отладки программного модуля с использованием специализированных программных средств 8](#_Toc182861404)

[2.4 Осуществление разработки тестовых наборов и тестовых сценариев для программного обеспечения 9](#_Toc182861405)

[2.5 Инспектирование компонента программного обеспечения на предмет соответствия стандартам кодирования 10](#_Toc182861406)

[3 ВЫПОЛНЯЕМЫЕ ЗАДАНИЯ 12](#_Toc182861407)

[ЗАКЛЮЧЕНИЕ 13](#_Toc182861408)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 14](#_Toc182861409)

# ВВЕДЕНИЕ

Практика проходила в ООО «Малленом Системс»

Дата проведения практики: 27.09.2024г. – 09.10.2024г.

Цель производственной практики: закрепление и углубление теоретических знаний, приобретение навыков по общепрофессиональным дисциплинам и профессиональным модулям.

Задача: закрепить и совершенствовать общие и профессиональные компетенции.

Научиться:

1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации.

2. Выполнять интеграцию модулей в программном обеспечении, отладку программного модуля (используя специализированные программные средства).

3. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

4. Производить инспектирование компонента программного обеспечения на предмет соответствия стандартам кодирования.

1 ОБЩАЯ ХАРАКТЕРИСТИКА ПРЕДПРИЯТИЯ

Малленом Системс – ведущая российская компания в области разработки и внедрения систем компьютерного зрения и промышленной видеоаналитики на основе технологий машинного зрения и искусственного интеллекта (машинное обучение, нейронные сети глубокого обучения). Профилем компании является – реализация наукоемких IT-проектов в сфере транспорта, машиностроения, нефтегазовой, металлургической, алмазодобывающей, атомной промышленности и других отраслях.

1. Организационная структура предприятия

Живиця Анна Эдуардовна - генеральный директор, кандидат экономических наук.

Царев Владимир Александрович - директор по развитию, кандидат технических наук.

Веснин Евгений Николаевич - технический директор.

Орголайнен Анна Николаевна - коммерческий директор.

Михайлов Андрей Евгеньевич - руководитель отдела разработки и сопровождения ПО.

Воскресенский Евгений Михайлович - руководитель направления системной интеграции, кандидат технических наук.

1. Внутренний распорядок работы предприятия

Пн.-Пт: 09:00-13:00

14:00-18:00

Сб.-Вс.: выходной

1. Должностные инструкции ИТ-специалистов предприятия

Техник относится к категории специалистов.

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

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

Требования к знаниям техника: Языки, утилиты и среды программирования, и средства пакетного выполнения процедур;

Методы автоматической и автоматизированной проверки работоспособности программного обеспечения;

Методы создания и документирования контрольных примеров и тестовых наборов данных;

Основные методы измерения и оценки характеристик программного обеспечения;

Языки, утилиты и среды программирования, и средства пакетного выполнения процедур;

Типовые метрики программного обеспечения;

Языки программирования и среды разработки;

Внутренние нормативные документы, регламентирующие порядок документирования результатов проверки работоспособности программного обеспечения;

Методы и средства рефакторинга и оптимизации программного кода;

Методы и средства проверки работоспособности программного обеспечения;

Среду проверки работоспособности и отладки программного обеспечения;

Требования к структуре и форматам хранения тестовых наборов данных; 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;

Локально-нормативные акты Общества, касающиеся выполнения его должностных обязанностей; Требования охраны труда и правила пожарной безопасности.

Права техника: участвовать в обсуждении проектов решений, в совещаниях по их подготовке и выполнению.

Запрашивать у непосредственного руководителя разъяснения и уточнения по данным поручениям, выданным заданиям.

Запрашивать по поручению непосредственного руководителя и получать от других работников организации необходимую информацию, документы, необходимые для исполнения поручения.

Знакомиться с проектами решений руководства, касающихся выполняемой им функции, с документами, определяющими его права и обязанности по занимаемой должности, критерии оценки качества исполнения своих трудовых функций.

Вносить на рассмотрение своего непосредственного руководителя предложения по организации труда в рамках своих трудовых функций.

Участвовать в обсуждении вопросов, касающихся исполняемых должностных обязанностей.

2 ОСУЩЕСТВЛЕНИЕ ИНТЕГРАЦИИ ПРОГРАММНЫХ МОДУЛЕЙ

1. Разработка требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент

Разработка требования к программным модулям включает в себя следующие шаги:

1. Анализ проектной документации (изучение архитектуры системы, используемых технологий и компонентов).
2. Определение взаимодействия (анализ технологий обмена данными) и выявление зависимостей между компонентами.
3. Формулирование требований (функциональные, нефункциональные, требования к совместимости).
4. Документирование (создание спецификаций).
5. Верификация и валидация (проверка требований на полноту и соответствие бизнес-целям).
6. Выполнение интеграции модулей в программное обеспечение

Интеграция модулей в ПО может проходить по следующей схеме:

1. Планирование интеграции: определение архитектуры системы и стратегии интеграции, включая выбор подходов.
2. Подготовка окружения: настройка среды разработки и тестирования, необходимых ля интеграции модулей.
3. Разработка интерфейсов: создание и документирование API и интерфейсов для взаимодействия между модулями.
4. Кодирование: реализация интеграции согласно разработанным интерфейсам и требованиям.
5. Тестирование интеграции: функциональное тестирование (проверка что все модули работают и взаимодействуют между собой), нагрузочное тестирование (оценка производительности системы под нагрузкой).
6. Документирование: регистрация всех изменений и использования модулей, создание документации для будущего использования и поддержки.
7. Деплоймент: развертывание интегрированного программного обеспечения в рабочей среде.
8. Мониторинг и поддержка: наблюдение за работой интегрированной системы и выявление возможных проблем.
9. Выполнение отладки программного модуля с использованием специализированных программных средств

Отладка программного обеспечения – это процесс поиска и исправления ошибок или багов в программе.

Процесс отладки включает несколько этапов:

1. Обнаружение ошибки: определение симптомов, указывающих на наличие ошибки в программе.
2. Воспроизведение ошибки: создание условий, при которых ошибка проявляется снова, чтобы лучше понять ее природу.
3. Диагностика проблемы: использование различных инструментов и методов для анализа кода и поиска причины ошибки.
4. Исправление ошибки: внесение изменений в код для устранения ошибки.
5. Тестирование: проверка исправления и тестирование программы, чтобы убедиться, что ошибка устранена и не появились новые проблемы.
6. Осуществление разработки тестовых наборов и тестовых сценариев для программного обеспечения

Цели разработки и выполнения тестов:

1. Обеспечить очищение ПО от ошибок (основная задача – ликвидировать очевидные ошибки);
2. Убедить, что ПО отвечает оригинальным требованиям и спецификации;
3. Обеспечить уверенность в ПО (пользователям, заказчикам и т.д.).

Схема разработки тестовых наборов и сценариев может быть разной. Примерная схема разработки:

1. Определение целей тестирования (функции и цели, которые должны быть протестированы).
2. Анализ требований (изучить функциональные и нефункциональные требования к ПО).
3. Разработка тестовых наборов: классификация (определите типы тестов), создание тестовых наборов (объедините тестовые случаи в наборы на основе общей функциональности или характеристик)
4. Создание тестовых сценариев: тестовые условия (опишите условия, при которых будет выполняться тест), шаги тестирования (опишите последовательность шагов, для проверки функциональности), ожидаемые результаты (что должно произойти на каждом шаге или в результате теста).
5. Использование инструментария: используйте специализированные инструменты для управления тестами и отслеживания тестовых случаев.
6. Ревью и оптимизация: проведите ревью тестовых наборов и сценариев с командой (чтобы убедиться в их полноте и актуальности), оптимизируйте тестовые сценарии (устраняя дублирование и ненужные шаги).
7. Автоматизация тестирования: определите, какие тесты могут быть автоматизированы, и используйте инструменты автоматизации.
8. Ведение документации: поддерживайте актуальность документации по тестовым сценариям и наборам, фиксируя изменения и обновления
9. Выполнение тестов: запустите тестовые сценарии, зафиксируйте результаты и анализируйте их для выявления ошибок и проблем.
10. Анализ результатов тестирования: оцените успешность тестов, составьте отчеты и передайте их заинтересованным сторонам (с информацией о найденных дефектах и рекомендациях по исправлению).
11. Инспектирование компонента программного обеспечения на предмет соответствия стандартам кодирования

В первую очередь нужно заняться подготовкой к инспекции, она включает сбор документов, выбор участников, выбор инструмента.

Далее - проведение инспекции, её можно провести по следующему плану:

1. Обзор кода (инспектируют код компонента).
2. Использование инструментов статического анализа: примените инструменты статического анализа для выявления нарушений стандартов.

После инспекции следует задокументировать замечания, оценить критичность.

По завершению инспекции следует выполнить следующие рекомендации:

1. Обсуждение результатов: проведите встречу с участниками инспекции для обсуждения выявленных проблем.
2. Исправление недостатков: создайте план по исправлению выявленных несоответствий.
3. Заключительные действия: подтверждение исправлений, обновление документации.
4. Обучение и улучшение: организуйте обучение для команды на основе замечаний и выявленных проблем.

3 ВЫПОЛНЯЕМЫЕ ЗАДАНИЯ

В первые дни я ознакомилась с внутренним распорядком компании, инструкциями, требованиями к работе сотрудников и подписывала необходимые документы, о безопасности, конфиденциальности, обязанностях и неразглашении.

Со второго дня и до конца недели я выполняла задания от руководителя,в своём репозитории.

Во время третьей недели выполняла задания по созданию проекта на выбранную тему, так же параллельно писала отчёт.

4 GITHUB

GitHub — это мощная платформа для хранения кода и совместной разработки проектов с богатым набором инструментов для автоматизации, контроля версий и управления задачами. Благодаря своей популярности и активному сообществу он стал стандартом де-факто в индустрии программирования.

Если у вас есть конкретные вопросы или нужно подробнее о каком-то аспекте — спрашивайте!

Основные понятия и функции GitHub

1. Репозитории (Repositories)

Это хранилища проектов, содержащие файлы, историю изменений и метаданные.

Можно создавать публичные (открытые для всех) или приватные (доступ только определённым пользователям).

В репозиториях хранится весь код проекта, документация, скрипты и другие ресурсы.

2. Ветки (Branches)

Позволяют параллельно работать над разными версиями проекта.

Обычно создаются для разработки новых функций или исправлений ошибок без влияния на основную ветку (обычно main или master).

3. Pull Requests (Запросы на изменение)

Предлагают внести изменения из одной ветки в другую после проверки и обсуждения.

Обеспечивают контроль качества кода и совместную работу.

4. Issues (Задачи/Ошибки)

Используются для отслеживания ошибок, задач, предложений по улучшению проекта.

5. Actions (Автоматизация)

Позволяют автоматизировать процессы CI/CD: тестирование, сборка, деплой.

6. Wiki и документация

Встроенные инструменты для ведения документации проекта.

Возможности GitHub

Совместная работа: команда может одновременно работать над проектом.

Контроль версий: отслеживание изменений в коде за время.

Код-ревью: проверка изменений перед их внедрением.

Интеграции: подключение сторонних сервисов (например, CI/CD системы).

Обучение и сообщество: возможность участвовать в open-source проектах.

Облачное хранение: доступ к проектам из любой точки мира.

5 РЕПОЗИТОРИИ НА GITHUB

Репозиторий (или "репо") — это хранилище проекта, где хранится весь его код, история изменений, документация и другие связанные файлы. Он служит основой для организации работы над проектом и совместной разработки.

Основные компоненты репозитория

1. Файлы и папки

Хранят исходный код, документацию, скрипты, конфигурационные файлы и прочие ресурсы проекта.

Структура папок зависит от типа проекта и предпочтений команды.

2. История версий (коммиты)

Каждое изменение в файлах фиксируется с помощью коммита — снимка состояния проекта в определённый момент времени.

В коммите указывается сообщение, описывающее сделанные изменения.

3. Ветки (Branches)

Позволяют параллельно работать над разными версиями или функциями проекта.

Основная ветка обычно называется main или master.

Создаются дополнительные ветки для разработки новых функций или исправлений.

4. Теги (Tags)

Используются для обозначения релизов или важных точек в истории проекта (например, версии 1.0).

5. Pull Requests (Запросы на слияние)

Предлагают внести изменения из одной ветки в другую после проверки и обсуждения.

Обеспечивают контроль качества кода перед объединением.

Типы репозиториев

Публичные (Public): доступны всем пользователям GitHub; идеально подходят для open-source проектов.

Приватные (Private): доступны только определённым пользователям или команде; используются для закрытых проектов или корпоративных разработок.

Возможности работы с репозиториями

1. Клонирование (Clone)

Создание локальной копии репозитория на вашем компьютере командой git clone.

Позволяет работать с проектом офлайн.

2. Внесение изменений и коммиты

После редактирования файлов вы делаете коммиты (git commit), фиксируя изменения.

3. Отправка изменений (Push)

Загружаете локальные изменения в удалённый репозиторий командой git push

4. Получение обновлений (Pull)

Обновляете локальную копию последними изменениями из удалённого репозитория командой git pull.

5. Создание веток и слияние (Branching & Merging)

Создаёте новые ветки (git branch), работаете над ними, а затем объединяете (git merge) с основной веткой.

Управление доступом

На GitHub можно настроить уровни доступа к репозиторию:

Чтение: просмотр содержимого и истории изменений.

Запись: возможность вносить изменения и делать коммиты/пуши.

Администрирование: управление настройками репозитория, командами и правами.

Инструменты внутри репозитория

Issues: отслеживание ошибок, задач и предложений по улучшению проекта.

Projects: доски Kanban для организации задач и планирования работы.

Wiki: встроенная документация по проекту.

Actions: автоматизация процессов CI/CD прямо внутри репозитория.

5 Разработка API-приложения «Модуль обработки и работы с изображениями» с использованием FastAPI и SQLite

Разработка API-приложения «Модуль обработки и работы с изображениями» с использованием FastAPI и SQLite — это создание программного интерфейса (API), который позволяет пользователям или другим системам взаимодействовать с функционалом для обработки изображений, при этом данные хранятся в базе данных SQLite. Ниже я подробно объясню, что это означает:

Что такое API-приложение?

API (Application Programming Interface) — это набор правил и интерфейсов, который позволяет разным программам взаимодействовать друг с другом. В данном случае, API-приложение предоставляет возможность отправлять запросы (например, загружать изображение, получать обработанное изображение, удалять или обновлять информацию о изображениях) через HTTP.

Почему именно FastAPI?

FastAPI — современный фреймворк для разработки веб-приложений на Python, который особенно хорошо подходит для создания REST API по следующим причинам:

Высокая производительность

Простота и лаконичность кода

Поддержка асинхронных операций

Автоматическая генерация документации (Swagger UI)

Легкая интеграция с другими библиотеками

Почему SQLite?

SQLite — легкая встроенная база данных, которая не требует отдельного сервера. Она идеально подходит для небольших проектов или прототипов:

Простая установка и настройка

Хранит данные в одном файле

Хорошо подходит для хранения информации о изображениях, метаданных, пользователях и т.п.

Что включает такой проект?

Обработка изображений:

Загрузка изображений на сервер

Выполнение операций обработки (например, изменение размера, фильтры, преобразования)

Сохранение обработанных изображений или их метаданных

Работа с базой данных:

Хранение информации о загруженных изображениях (имя файла, дата загрузки, параметры обработки)

Возможность поиска и фильтрации по метаданным

REST API:

Эндпоинты для загрузки/скачивания изображений

Эндпоинты для запуска обработки изображений

Эндпоинты для получения информации о сохраненных изображениях

В чем заключается разработка такого модуля?

Создание моделей данных (например, таблиц в SQLite) для хранения информации об изображениях и их метаданных.

Реализация API-эндпоинтов с помощью FastAPI:

POST /images/ — загрузка нового изображения

GET /images/{id} — получение информации о конкретном изображении

POST /images/{id}/process — запуск обработки изображения

GET /images/ — список всех загруженных изображений

Обработка изображений с помощью библиотек Python (например, Pillow)

Хранение файлов изображений на сервере или в базе данных (обычно файлы хранятся отдельно от базы)

Это проект по созданию веб-сервиса на Python с помощью FastAPI и SQLite, который позволяет пользователям загружать изображения, выполнять их обработку и управлять ими через REST API. Такой модуль может быть частью более крупной системы или самостоятельным сервисом.

ЗАКЛЮЧЕНИЕ

Я закрепила и углубила теоретические знания, приобрела навыки по общепрофессиональным дисциплинам и профессиональным модулям, а также закрепила и совершенствовала общие и профессиональные компетенции.

Также я смогла научиться:

1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации.
2. Выполнять интеграцию модулей в программном обеспечении, отладку программного модуля (используя специализированные программные средства).
3. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.
4. Производить инспектирование компонента программного обеспечения на предмет соответствия стандартам кодирования.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официальный сайт ООО «Малленом Системс» [Электронный ресурс] – режим доступа: https://www.mallenom.ru/
2. АНО ДПО «Академия подготовки главных специалистов», курс «Программирование» [Электронный ресурс]/ Алексеев В.М. – режим доступа: https://specialitet.ru/lekcyi/progr/lekcyy\_modul\_3\_vopros\_2.pdf
3. Сайт AppMaster, блог «Типы системной интеграции» [Электронный ресурс] – режим доступа: https://appmaster.io/ru/blog/tipy-sistemnoi-integratsii
4. Сайт JavaRush, курс «Отладка программы» [Электронный ресурс] – режим доступа: https://javarush.com/quests/lectures/ru.javarush.web.core.lecture.level19.lecture08
5. АНО ДПО «Академия подготовки главных специалистов», курс «Программирование» [Электронный ресурс]/ Алексеев В.М. – режим доступа: https://specialitet.ru/lekcyi/progr/lekcyy\_modul\_2\_vopros\_2.pdf
6. Сайт Яндекс Практикум, лекция «Сценаристика в тестировании: что такое тестовый сценарий и зачем он нужен» [Электронный ресурс] – режим доступа: https://practicum.yandex.ru/blog/testovyi-scenarii-razrabotka-i-ispolzovanie/
7. Сайт Habr, блог «Внедрение инспекций кода в процесс разработки» - режим доступа: https://habr.com/ru/companies/pt/articles/148884/